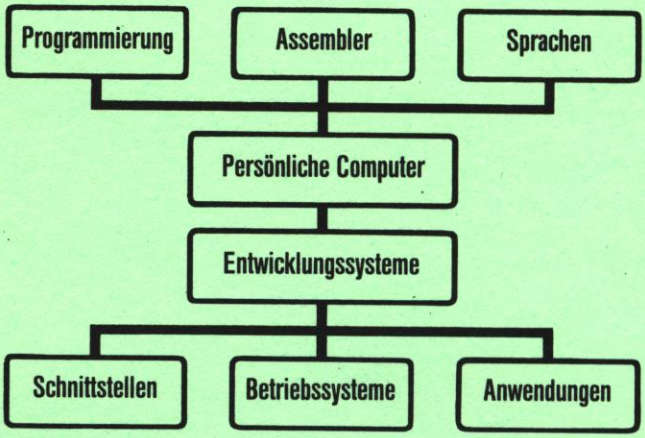
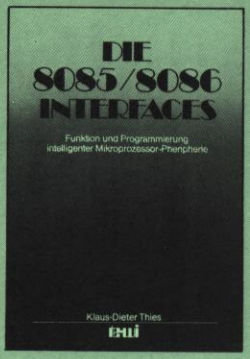
MICRO MAG

DM 9,50 Nr. 39 Oktober 1984

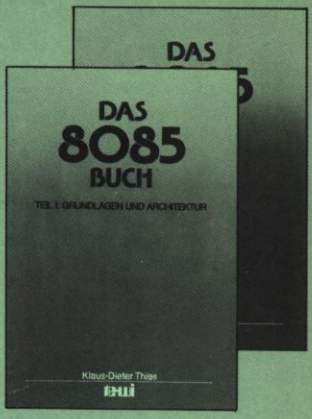


Inhaltsverzeichnis

AUSBILDUNG UND ENTWICKLUNG
in der Mikroprozessortechnik

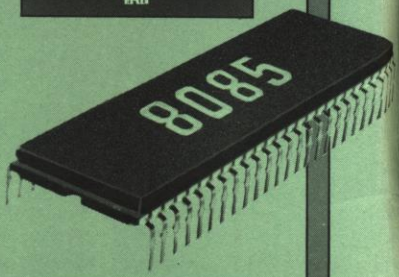


Die 8085/8086 Interfaces
K.-D. Thies
Beschreibung von Funktion und Anwendung der meisten Peripherie-Bausteine zu den CPUs 8085/8086.
656 Seiten, A 4, Softcover, DM 89,-.



Das 8085 Buch
K.-D. Thies
Teil 1: Grundlagen und Architektur
288 Seiten, A 4, Softcover, DM 49,-
Teil 2: Software
328 Seiten, A 4, Softcover, DM 49,-

Das 8086/8088 Buch
R. Rector/G. Alexy
Programmieren in Assembler und Systemarchitektur der hochaktuellen 16-Bit-CPUs 8086/8088.
560 Seiten, Softcover, DM 79,-



Diese Texte dienen Deutschlands Entwicklung in der Mikroelektronik – auf höchstem didaktischen und technologischen Niveau. Erprobt in industrieller Entwicklung, Werkausbildung und an Hochschulen.

te-wi

te-wi Verlag GmbH
Theo-Prosel-Weg 1
8000 München 40

In Vorbereitung:

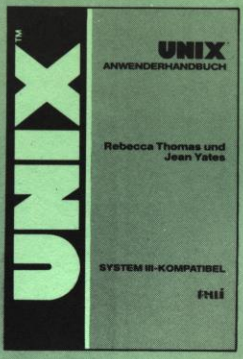
Der Arithmetik-Prozessor 8087
Architektur, Funktion und Programmierung des 8086/8087-Coprozessorsystems

Das Mikrocontroller-Buch
Hardware, Befehlssatz und Mehrrechnersysteme des 8051

Das 8086 Systembuch
I/O-Interfacing; Multiprocessing; Entwurf von Bus-Interfaceschaltungen und Multibus-Systemen mit dem Bus-Arbiter 8289

Der 8086/8088 Assembler
Aufbau von ASM86-Programmen, Segmentierung und modulare Programmierung

8080A/8085 – Programmieren in Assembler
L. A. Leventhal
Anschauliche, bildreiche Darstellung mit umfangreicher Beispielsammlung.
463 Seiten, Softcover, DM 49,-



UNIX™-Anwenderhandbuch
R. Thomas/J. Yates
Für alle, die mit diesem weitverbreiteten Betriebssystem arbeiten wollen. Von der international bekannten UNIX-Expertin Jean Yates.
550 Seiten, Softcover, DM 79,-

Roland Löhr

SPS für AIM 65

Software Preparation System

Heft 34 enthielt auf Seite 58 die Ankündigung des o.a. Rockwell-Produktes, das auch hier mittlerweile getestet werden konnte. Es handelt sich um ein gegenüber der Grundausstattung erweitertes Monitor-/Editor-/Assembler-/Disassemblerprogramm, um ein Arbeitsmittel zur Programmentwicklung also, speziell für die Erfordernisse der CMOS-CPU R65C02 mit ihrem erweiterten Befehlssatz.

Firmware

Zum SPS gehören eine CMOS-CPU und vier Festwertspeicher von je 4KB für die Sockel hex E000 und F000 (Monitor und Editor) sowie für B000 und C000 (Assembler, Disassembler). Optional sind zwei weitere ROMs, die auf die RM65-Module von Rockwell abgestimmt sind, nämlich das CRTC-ROM (Teil SPS-110) für Bildschirmausgabe und das Prommer-ROM (Teil SPS-120) für die Programmierung von EPROMs und EEROMs der Typen 2716, 2532, 2732, 2732A, 87C32, 2564, 2764, 68764, 27128 sowie 2816, 5213 und 48016.

Dazu gehört das Handbuch 'R6500 Software Preparation User's Manual' (Best. Nr. 2167) mit etwa 250 Seiten Inhalt. Es dokumentiert diese neuen Teile für den Einbau und für den Betrieb unter den gegebenen Befehlen. Angesichts der hier enthaltenen Assembler-Programmliste für den Bereich (nur !) E000 bis FFFF stellt das Handbuch somit eine ergänzende und aktualisierende Dokumentation sowohl für das AIM-Anwenderhandbuch dar, wie auch für das 'Monitor Program Listing'.

Allgemeine Hinweise

Die in den Festwertspeichern enthaltenen Programme sind für den Befehlssatz der herkömmlichen CPU 6502 geschrieben, so daß der Betrieb einer R65C02 für diese nicht zwingend ist.

Es sind zahlreiche in dieser Zeitschrift veröffentlichte Verbesserungen und Anregungen des Betriebssystems berücksichtigt worden, jedoch nicht diejenigen, die mit der Veröffentlichung von REMON zu einer durchgreifenden Überarbeitung führten. Enthalten sind weiterhin z.B. die meist entbehrlichen Routinen für KIM-1, deren Ausräumung ordentlich Platz für andere Implementierungen schaffte. Die mit REMON geschaffene beliebige Erweiterung der Befehlsebenen in Monitor und Editor ist ebenfalls nicht enthalten. Dagegen finden wir eine sehr breite Auffächerung der anwenderbestimmten Ein- und Ausgabe, auch die mögliche Einbindung eines Bildschirm-Monitors und eines Floppy-Systems. – Nachfolgend sollen vor allem die neuen Merkmale genannt werden, um dem Leser Bewertungskriterien zu geben.

Neue Speicherbenutzung

Es wurde bereits erwähnt, daß die ROMs mit Assembler/Disassembler und weiteren Utilities auf die Sockel B000 und C000 gehören. Hier werden normalerweise Sprachen wie BASIC und FORTH gesteckt. Wer also SPS und Sprachen abwechselnd betreiben möchte, sollte in diesem Bereich umschaltbare Sockel-Batterien haben oder dort RAM vorsehen, in das jeweils geladen wird. Es kommt eine weitere Besonderheit für das RAM: Der Bereich bis Adresse 7FF ist für verschiedene Systemvariable und Sprungleisten reserviert, so daß Sprachen und ggfs. existierende Anwenderprogramme 'höher gelegt' werden müssen. Für Sprachen enthält das Handbuch dazu Hinweise, das SPS ist jedoch nicht kompatibel mit den ROMs für PL/65.

Wenn nur die ROMs auf den Adressen E000 und F000 eingesetzt werden, entfallen die Funktionen N (Assembler-Aufruf), K (Disassembler) und I (zeilenweise Instruktionseingabe).

Innerhalb eines SPS-Systems mit RM65-Erweiterungsmodulen sind weiterhin folgende Speichernutzungen vorgesehen: Ab 8000 Floppy Disk Controller, 9000 mit Prommer und D000 für das ROM des CRT-Controllers.

Es ist darauf hinzuweisen, daß die SPS-ROMs praktisch keine unbenutzten Bereiche mehr haben, in die man eigene Routinen einbrennen könnte. Platz kann nur durch Verzicht z.B. auf KIM-Routinen geschaffen werden oder durch die Benutzung von Bereichen, die zwar für RM65-Module vorgesehen sind, die aber vom Anwender noch nicht belegt wurden, z.B. ab D000 usw.

Der Editor wird 'ersatzweise' (wenn man keine Parameter angibt) bei Adresse 800 gestartet. Aber auch die Assembler-Symboltafel wird 'ersatzweise' von 800-2000 initialisiert, was dann gelegentlich zu Überraschungen führt, wenn die Symboltafel den Quelltext zerstört hat.

Neue Befehle mit der Taste 'U'

Beim Reset des Systems sind die im Handbuch vermerkten Besonderheiten zu beachten, damit richtig initialisiert wird, insbesondere was das CRT-Format und die im RAM abzulegenden Anwender-Vektoren angeht. Die Taste 'U' im Monitor führt zu neuen Funktionen, die gedanklich von den Begriffen IN=U und OUT=U zu trennen sind (letztgenannte sind auch geändert). Nach 'U' finden wir ein Menue und können nun mit einer Zifferntaste eine Wahl treffen. Es sind dies:

0 Directory, verlangt ebenso wie 1, 2, 3 das DOS-ROM. Inhaltsverzeichnis der Diskette(n) lesen,
1 Aufruf von DOS Utilities,
2 Schließen von Disketten-Files,
3 Kopieren von Disk-Files,
4 Speicheranzeige in Hexbytes und als ASCII-Zeichen, soweit darstellbar. Analysefunktion in Ergänzung zum einfachen Memory-Befehl,
5 Abspeicherung der Assembler-Symboltafel auf verschiedene Medien,
6 Auflistung der Symboltafel,
7 Re-Enter Editor, mit der Möglichkeit in einen zweiten Textbereich zu gehen,
8 Funktionen zum PROM-Programmieren (Menue mit 15 Möglichkeiten).

Erweiterte Ein- und Ausgabe-Handler

Sofern man beim Reset richtig initialisiert hat führen Befehle wie IN=U und OUT=U zunächst auf ein Menue, das mit einer Ziffer zu beantworten ist. Es handelt sich damit um eine Aufsplittung der anwenderbestimmten Ein- und Ausgabe auf jeweils 10 Möglichkeiten, die zum Teil vom System vorbesetzt sind, die man um eigene Treiberprogramme ergänzen kann oder die man, weil im RAM, auch anders vektorieren kann. Für die **Ausgabe** haben wir folgende Vorbesetzung:

0 Diskette
1 Printer des AIM 65
2 Magnetband
3 Serieller Kanal
4 Memory (Ausgabe vom Editor in einen definierten Speicherbereich)
5 Ausgabe auf einen Drucker mit Centronics-Interface.

Ähnlich bei der **Eingabe:** Von einem Menue aus kann in 10 verschiedene Kanäle anwenderbestimmt eingegeben werden, fünf davon sind vom System schon vorbelegt, fünf können vom Anwender dazugelinkt werden:

0 Eingabe von Diskette
1 Eingabe von der Tastatur das AIM 65
2 Eingabe vom Tonband
3 Eingabe vom seriellen Port
4 Eingabe aus einem zu definierenden Speicherbereich in den Textpuffer des Editors. Zusammen mit der entspr. Ausgabe sind hiermit Um-Ordnungen in Texten möglich.

Verbesserte Disassemblierung

Das K-Kommando ist nicht nur um die neuen Befehle erweitert worden, man kann jetzt mit dem Ausgabe-Handler auch direkt in den Text-Editor disassemblieren. Dabei werden die bei einer späteren Wieder-Assemblierung des Textes störenden Adreßangaben am Zeilenbeginn unterdrückt, so daß keine Überarbeitung nötig ist. Es gibt sogar die Option, daß in der Symboltabelle enthaltene Adressen und Werte als Symbole in den disassemblierten Text übernommen werden.

Natürlich ist auch das I-Kommando für die direkte zeilenweise Instruktionseingabe in den Speicher dem neuen Befehlssatz und seiner Assembler-Schreibweise angepaßt worden. Das V-Kommando hat jetzt neue Funktionen, es schaltet die Durchsuchung einer Symboltafel ein, so daß es jetzt auf die Disassemblierung (s.o.) wirkt, aber auch auf den schrittweisen Befehlsdurchlauf (STEP mit eingeschaltetem Z-Kommando). Im letzteren Fall finden wir neben der Adresse und dem Opcode die Ausgabe ggfs. eines Labels, des menmonischen Befehles, der Operanden und auch, das ist neu, die Registerinhalte, das Statusregister dabei mit symbolisch gesetzten Flags.

Der Assembler

Die mnemonische Eingabe von Befehlen und Anweisungen ist ziemlich gleich geblieben. Änderungen traten durch neue Befehle, Adressierungsarten und Optionen ein, die sich auch auf das Listenbild und auf die weitere Ausgabe auswirken. Bei der in Pass 2 erstellten Liste finden wir zunächst Fehlerhinweise in Klartext (nicht nur Nummern), die wesentlich hilfreicher sind. Zweitens kann durch eine Option erreicht werden, daß Befehle, die vom Standard 6502 abweichen, besonders kenntlich gemacht werden. Am Ende des Pass 2 schreibt der Assembler ferner eine Symboltafel, Cross-References und die Zahl der Fehler und Warnungen. Zu diesem Zweck ist die Symboltafel des Assemblers neu organisiert worden. Brauchte sie vorher 8 Bytes je Eintrag, so sind es jetzt 13. Neu sind darin die Zahl der Benutzungen, ein Link zur Cross-Referenz und die Zeile, in der ein Symbol definiert wurde.

Die generierte Assembler-Liste wird im Format 80 Zeichen je Zeile ausgegeben. Sie hat damit eine'anständige' Formatierung, wie wir sie seit Jahren in den Programmen dieser Zeitschrift finden, das gilt auch für die alphabetisch geordnete Symboltafel und für die Cross-Referenz-Liste.

Auch bei den Direktiven des Assemblers sind Verbesserungen eingetreten. Die Ausgabe einer Symboltafel oder einer Cross-Referenz kann gefordert oder unterdrückt werden, ebenso die Ausgabe von Warnungen hinsichtlich des abweichenden Befehlssatzes und seiner Adressierungsarten. Auch bei der Verkettung von Eingabeströmen mit der .FILE- und .END-Anweisung sind neue Möglichkeiten gegeben, als neben dem Namen auch eine Einheiten-Nummer angegeben werden kann. Damit können Textabschnitte von verschiedenen Medien gekoppelt werden.

Zusammenfassung

Rockwell hat mit der Freigabe des SPS-Programmpakets lange gezögert. Mittlerweile hat eine große Zahl von AIM-Betreibern das System individuell für die Bedürfnisse eingerichtet. Dazu gab es in den vergangenen Jahren viele Vorschläge in dieser aber auch in anderen Zeitschriften. Man wird das SPS heute häufig unter dem Gesichtspunkt betrachten müssen, was es zusätzlich bringt und was es evtl. noch nicht hat, was aber wünschenswert wäre. Es hat zweifellos den Vorteil, eine Vielzahl von Diensten bereitzustellen, die man sonst nur in verstreuten Veröffentlichungen findet oder auf diversen Programm-Files, die man sich im Laufe der Zeit selber zugelegt hat.

Im SPS war man bemüht, die Haupt-Einsprungspunkte für Systemroutinen an die alten Adressen zu legen. Das betrifft vor allem die Ein- und Ausgabe, nicht jedoch viele untergeordnete Routinen, deren Platz sich verschoben hat. Hier mag sich der früher übliche Programmierstil rächen, in System- und Anwenderprogrammen möglichst viele vorhandene Unterprogramme mitzubenutzen, um möglichst Speicherplatz zu sparen. Unter heutigen Gesichtspunkten sollte man lieber möglichst eigenständige vollständige Module schreiben, die man leichter anpassen kann, wenn sich notwendige Änderungen ergeben. Verzahnungen mit Systemprogrammen stellt man übersichtlicher durch schneller re-definierbare Sprungleisten dar.

Eine Anpassung des SPS für die zwischenzeitlich für den AIM 65 vorgeschlagenen zusätzlichen Dienstprogramme ist in jedem Fall mit Arbeit verbunden, auch die Anpassung von ladbaren Programmen und Sprachen auf die neue Speicherbelegung. Es mag in vielen Fällen bequemer sein, mit umschaltbaren Speichermodulen zu arbeiten, aus denen man die jeweils benötigten Dienste heranzieht. - Gleichwohl enthält die Dokumentation zum SPS verschiedene Anregungen und Vorbilder für individuelle Anpassungen des eigenen Systems, die man auch berücksichtigen könnte.

Karl-Anton Dichtel, 7800 Freiburg

Vergleich zweier Speicherbereiche

mit Ausgabe der Abweichungen

Bei Arbeiten mit verschiedenen Monitor- und Programmversionen zeigte sich, daß die nachträgliche Kontrolle durchgeführter Änderungen durch das folgende Programm vereinfacht wird. Besonders bei Bedienungsfehlern läßt sich so ein einzelnes geändertes Byte sofort finden.

Das Programm wurde für die Monitor-Version REMON (s. Heft 28/83) mit Tastenrepeat nach Heft 31/83 geschrieben. Das Videoprogramm wurde ebenfalls übernommen, jedoch der RAM-Bereich nach 9000 bis 97FD und die Kontrollregister in die beiden folgenden Adressen verschoben.

Bei Verwendung mit dem Original-Monitor und ohne zusätzlichen Drucker entfallen die Programmteile, die den Drucker bedienen. - Das Programm erwartet das Referenz-Programm mit den Originaladressen und die geänderte Version mit beliebiger Adresse. Ausgegeben werden dann die geänderten Bytes. Das Programm wurde ausführlich kommentiert, so daß auch Anpassungen an die persönlichen Wünsche leicht möglich sind.

Das Beispiel zeigt den Anfang eines Vergleiches zwischen REMON und dem Original-Monitor des AIM 65.

```
0000          ;************************************
0000          ;*                                  *
0000          ;* VERGLEICH ZWEIER SPEICHERBEREICHE *
0000          ;* 30.04.1984                       *
0000          ;*                                  *
0000          ;************************************
0000          ;
0000          ;RAM-BEREICH IN PAGE 0
0000 RAM              =$EB              ;ANFANG RAM-BEREICH
0000 REFANF           =RAM              ;ANFANG REFERENZ-DATEN
0000 REFEND           =RAM+2            ;ENDE REFERENZ-DATEN
0000 VARANF           =RAM+4            ;ANFANG ZU VERGL.DATEN
0000 BLOZA            =RAM+6            ;ZAHL DER AUSGABE-BLOECKE
0000          ;MONITOR-ADRESSEN AIM65
0000 OUTALL           =$E9BC            ;AUSGABE ASCII-ZEICHEN
0000 FROM             =$E7A3            ;HOLE ADRESSE MIT FROM
0000 TO               =$E7A7            ;HOLE ADRESSE MIT TO
0000 CRLF             =$E9F0            ;AUSGABE CR AN D/P
0000 ADDR             =$A41C            ;*-ADRESSE
0000 BLANK            =$E83E            ;1*SPACE
0000 REDOUT           =$E973            ;ASCII-CHAR. VON KB
0000 PRIFLG           =$A408            ;BIT 1:FX-80 PRINTERFLAG
0000 NUMA             =$EA46            ;AKKU->2 ASCII-CHAR.
*************************************************************

0000          ;PROGRAMM-START
0000                  *=$7400
7400          D8      CLD
7401 MEMAD    20F0E9  JSR CRLF
7404          A000    LDY #0
7406          20FA74  JSR TEXT          ;DATEN-REF.
7409          20A3E7  JSR FROM          ;ANFANG
740C          B0F3    BCS MEMAD         ;FEHLER
740E          AD1CA4  LDA ADDR          ;UMSPEICHERN
7411          85EB    STA REFANF
7413          AD1DA4  LDA ADDR+1
7416          85EC    STA REFANF+1
```

```
7418          203EE8 JSR BLANK
741B          20A7E7 JSR TO            ;ENDE
741E          B0E1   BCS MEMAD         ;FEHLER
7420          AD1CA4 LDA ADDR          ;UMSPEICHERN
7423          85ED   STA REFEND
7425          AD1DA4 LDA ADDR+1
7428          85EE   STA REFEND+1
742A MAD1     20F0E9 JSR CRLF          ;NEUE ZEILE
742D          A010   LDY #TXT2-TXT1
742F          20FA74 JSR TEXT          ;DATEN-VARIANTE
7432          20A3E7 JSR FROM          ;ANFANG
7435          B0F3   BCS MAD1          ;FEHLER
7437          AD1CA4 LDA ADDR          ;UMSPEICHERN
743A          85EF   STA VARANF
743C          AD1DA4 LDA ADDR+1
743F          85F0   STA VARANF+1
7441          20F0E9 JSR CRLF          ;NEUE ZEILE
7444          A020   LDY #TXT3-TXT1    ;FX-80 DRUCKER EIN ?
7446          20FA74 JSR TEXT
7449          2073E9 JSR REDOUT        ;HOLE Y/N
744C          48     PHA
744D          20F0E9 JSR CRLF
7450          68     PLA
7451          C959   CMP #'Y'
7453          D008   BNE NOPR
7455          AD08A4 LDA PRIFLG        ;PRINTER EIN
7458          0902   ORA #2            ;SETZE BIT 1
745A          8D08A4 STA PRIFLG
745D NOPR     A042   LDY #TXT4-TXT1    ;UEBERSCHRIFT
745F          20FA74 JSR TEXT
7462          20AE74 JSR AUSPZ         ;AUSGABE:PROGRAMM-ZAEHLER
7465          A064   LDY #TXT5-TXT1    ;START-ADR.
7467          20FA74 JSR TEXT
746A NODIF    A900   LDA #0             ;LOESCHE ZAEHLER
746C          85F1   STA BLOZA
746E BVER     20BC74 JSR BYTVER        ;BYTE'S GLEICH ?
7471          F00D   BEQ GLEI          ;GLEICHHEIT-KEINE AUSGABE
7473          A5F1   LDA BLOZA
7475          D006   BNE BOUT          ;BYTE-AUSGABE
7477          20F0E9 JSR CRLF          ;NEUE ZEILE
747A          20AE74 JSR AUSPZ         ;ZUERST PROG.ZAEHLER
747D BOUT     20C974 JSR BYTOUT
7480 GLEI     20D474 JSR LASTAD        ;FERTIG ?
7483          F00B   BEQ LASTZE
7485          20DF74 JSR ADRERH        ;NAECHSTE ADR.
7488          A910   LDA #16            ;D.H.16 BYTE
748A          C5F1   CMP BLOZA
748C          F0DC   BEQ NODIF         ;NEUE ZEILE
748E          D0DE   BNE BVER
7490 LASTZE   20F0E9 JSR CRLF          ;LETZTE ZEILE
7493          20F0E9 JSR CRLF
7496          20AE74 JSR AUSPZ         ;LETZTE ADR.
7499          A087   LDY #TXT6-TXT1
749B          20FA74 JSR TEXT          ;ADR.LETZTES BYTE
749E          AD08A4 LDA PRIFLG        ;DRUCKER EIN ?
74A1          2902   AND #2
74A3          F006   BEQ AUSGA
74A5          4D08A4 EOR PRIFLG        ;DRUCKER WIEDER AUS !
74A8          8D08A4 STA PRIFLG
74AB AUSGA    4C82E1 JMP $E182         ;ZURUECK ZU MONITOR
```

```
74AE            ;AUSGABE PROGRAMMZAEHLER REF.DATEN
74AE AUSPZ      A5EC    LDA REFANF+1     ;HIGH-ADR.
74B0            2046EA  JSR NUMA
74B3            A5EB    LDA REFANF       ;LOW-ADR.
74B5            2046EA  JSR NUMA
74B8            203EE8  JSR BLANK
74BB            60      RTS
74BC            ;VERGLEICHE BYTES
74BC BYTVER     A200    LDX #0
74BE            A1EF    LDA (VARANF,X)
74C0            C1EB    CMP (REFANF,X)
74C2            D004    BNE BYTVE1
74C4            A900    LDA #0           ;NEUE ZEILE
74C6            85F1    STA BLOZA
74C8 BYTVE1     60      RTS
74C9            ;AUSGABE:AENDERUNG
74C9 BYTOUT     A1EF    LDA (VARANF,X)

74CB            2046EA  JSR NUMA
74CE            203EE8  JSR BLANK
74D1            E6F1    INC BLOZA
74D3            60      RTS
74D4            ;VERGLEICH FERTIG ?
74D4 LASTAD     A5EB    LDA REFANF
74D6            C5ED    CMP REFEND
74D8            D004    BNE LASTA1
74DA            A5EC    LDA REFANF+1
74DC            C5EE    CMP REFEND+1
74DE LASTA1     60      RTS              ;SIEHE Z-FLAG
74DF            ;ERHOEHE REFANF UM 1
74DF ADRERH     18      CLC
74E0            A5EB    LDA REFANF
74E2            6901    ADC #1
74E4            85EB    STA REFANF
74E6            A900    LDA #0
74E8            65EC    ADC REFANF+1
74EA            85EC    STA REFANF+1
74EC            ;ERHOEHE VARANF UM 1
74EC            18      CLC
74ED            A5EF    LDA VARANF
74EF            6901    ADC #1
74F1            85EF    STA VARANF
74F3            A900    LDA #0
74F5            65F0    ADC VARANF+1
74F7            85F0    STA VARANF+1
74F9            60      RTS
74FA            ;TEXT-AUSGABE AN AOD
74FA TEXT       B90875  LDA TXT1,Y
74FD            48      PHA
74FE            297F    AND #$7F
7500            20BCE9  JSR OUTALL
7503            C8      INY
7504            68      PLA
7505            10F3    BPL TEXT
7507            60      RTS
*************************************************************

7508 TXT1       4441    .BYT 'DATEN-REFERENZ ',$A0
7517            A0
```

```
7518 TXT2   4441    .BYT 'DATEN-VARIANTE  ',$A0
7527        A0
7528 TXT3   4155    .BYT 'AUSGABE AUF FX-80 DRUCKER ? (Y/N)',$A0
7549        A0
754A TXT4   5645    .BYT 'VERGLEICH ZWEIER SPEICHERBEREICHE',$8D
756B        8D
756C TXT5   2020    .BYT '  STARTADRESSE DES DATENVERGLEICHS',$8D
758E        8D
758F TXT6   2020    .BYT '  ENDADRESSE DES DATENVERLEICHS',$8D
75AE        8D
75AF        ;
75AF                .END
 ERRORS= 0000

VERGLEICH ZWEIER SPEICHERBEREICHE
E000   STARTADRESSE DES DATENVERGLEICHS

E13C B2 EF
E18B DA EF
E19F E4 EF
E322 02
E425 20 36 ED CA D0 09 A9 01 0D 08 A4 8D 08 A4 60 4C
E435 EA EF
E474 EA EA EA EA
E523 58
E529 EA EA EA EA EA
E75A 55 F2 00
E75E D4 E7 D4 E7
E765 00
E7E4 20 40 EE
E7F1 05
E860 EA EA EA EA EA EA EA EA
E889 EA EA EA EA EA
E94B CE EF
E9A1 EA EA EA
E9CC EA EA EA EA
EA91 41
ECEF 8A 48 A2 40 AD 08 A4 6A 90 02 A2 0A 20 EA EF 68
ECFF AA
EE3B A9 C7 EA
EE40 AD 15 A4 F0 0F CE 15 A4 09 C0 AA CA 8E
EE4E 97 A9 20 9D 00 93 60 5F 40 4A 07 1E 02 10 17 00
EE5E 09 40 09 00 00 03 C0
EE8E A9 C7 EA
EE94 C9 C7 EA
EE9A C9 C7 EA
EF6A F0 0A 2C 08 A4 30 09 48 8A 48 10 08
EF7B 4C 6C EF
EFB2 38 E9 2C 8D 18 A4 B0 03 CE 17 A4 60 48 8A D0 0A
EFC2 68 48 29 40 F0 04 68 09 20 48 68 60 20 40 EC 2C
EFD2 08 A4 50 03 20 BE EF 60
EFDB 96 FE C9 40 30 02 29 DF 60 6C 0A A4 6C 0C A4 AD
EFEB 82 A4 0D 7F A4 49 FF D0 08 A9 FE 2D 08 A4 4C 30
EFFB E4 4C 25 E4
F21D EA EA EA
F252 4C 90 F2 20 9E EB 48 48 A0 0F 8C FE 97 AD 15 A4
F262 09 C0 A8 68 29 7F C9 0D F0 0A C0 FC F0 0E 99 00
F272 93 C8 D0 05 20 C5 FF A0 C0 8C FF 97 68 20 AC EB
F282 4C 05 EF
```

Anwendung des Programmes:
Vergleich des REMON mit dem Original-Monitor

Peter Porbadnigk, 2070 Ahrensburg

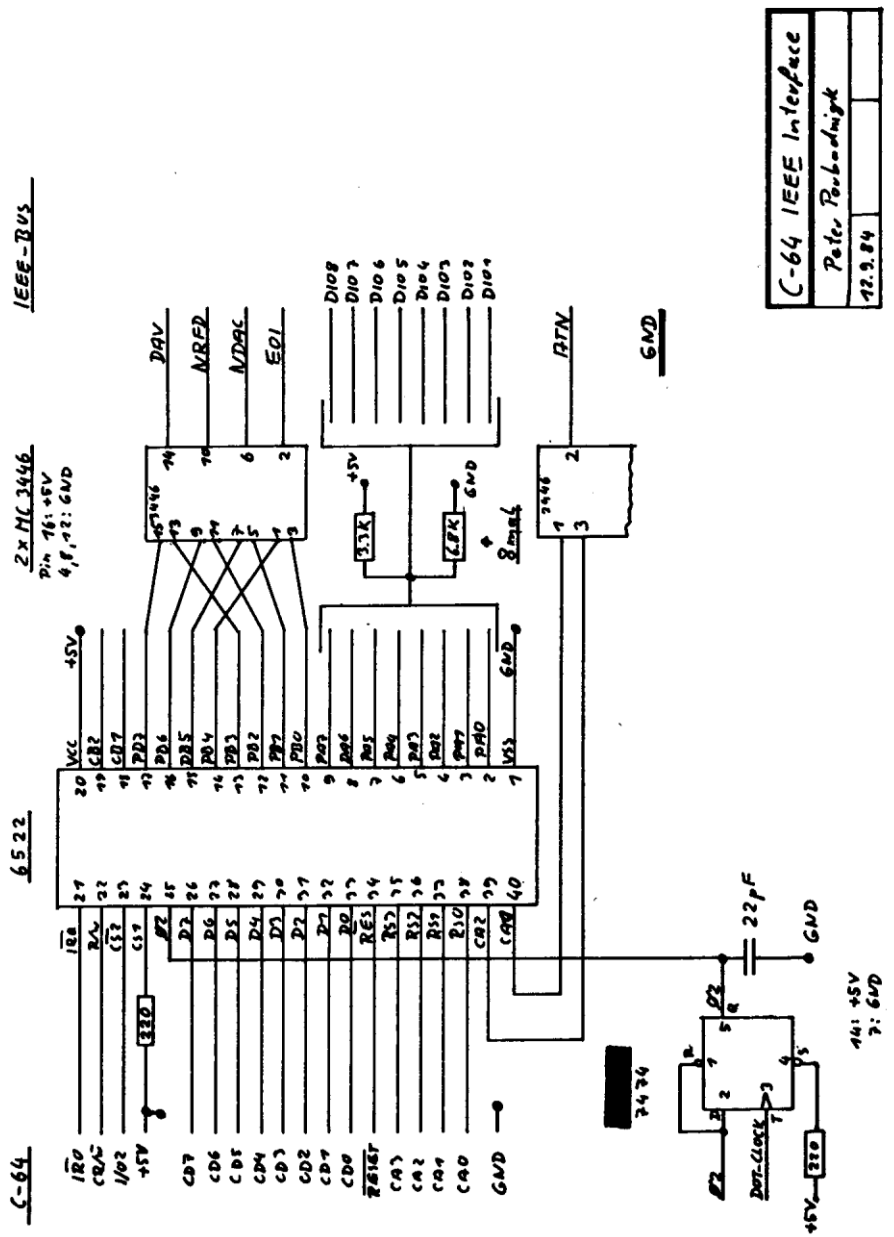
C-64 am IEEE 488-Bus

Mit der dazugehörenden einfachen Hardware realisiert das nachfolgende Programm am Commodore C-64 den parallelen IEEE 488-Bus, mit dem alle Geräte der großen CBM-Serie betrieben werden können. Die Hardware ist im Prinzip die schon seit langem in der MICRO MAG veröffentlichte Schaltung mit einer VIA 6522. Zur Anpassung an den C-64 muß die Taktleitung Phi2 verzögert werden (7474). Im allgemeinen reicht die Verzögerung durch diesen Chip. Beim Betrieb der CP/M-Karte wurde aber der Kondensator mit 22 Mikrofarad notwendig. Die in der Schaltung angegebenen IEEE-Treiber MC3446 und die Widerstände an den Datenleitungen können ggfs. entfallen. Bei mehreren Geräten am Bus wurden sie hier nötig. Sollen die Treiber entfallen, dann sind die Ports der Eingänge einfach mit denen der Ausgänge zu verbinden.

Die Software ersetzt die seriellen Routinen des Kernal-ROM. Der Betrieb der seriellen Schnittstelle ist anschließend nicht mehr möglich. Auch hier sind sicher andere Möglichkeiten realisierbar, z.B. Austausch gegen Kassetten-Routinen oder RS232. Das geänderte System benutzt keine Adressen zusätzlich, außer den I/O-Bereich DF00-DFFF für die VIA. Durch das Einhalten der Einsprungsadressen bleiben weitgehendst alle Programme funktionsfähig, wie Erweiterungen, Sprachen usw., soweit sie nicht eigene serielle Laderoutinen besitzen oder direkt mit dem DOS der 1541 zusammenarbeiten.

Soll die Software im RAM-Bereich E000-FFFF laufen, dann ist nach dem Umschalten vom ROM auf's RAM erst die Routine NEUTRL aufzurufen, um den Portbaustein zu initialisieren. Oder es ist das abgedruckte Ladeprogramm zu benutzen. - Hier wurde das geänderte Kernel-ROM in EPROMs fest im C-64 installiert. Für Anwender mit einfarbigen Monitoren ist es zu empfehlen, die am Ende des Listings stehenden Farbwerte für den Bildschirm nach eigenem Ermessen zu ändern.
Die Befehle für das Interface und den IEEE-Bus entsprechen denen des BASIC 3, wie sie in den Handbüchern für die Floppies und Drucker dokumentiert sind.

```
0001  0000                ;ladeprogramm für geändertes kernal-rom
0002  0000                ;mit dem namen: kernal.ieee     von disk
0003  0000                ;=======================================
0004  0000                        *=$801
0005  0801  10 08                 .wor basend
0006  0803  40                    .byt 64,0,0,'     ',$9e,'2080',0
0006  0804  00
0006  0805  00
0006  0806  20 20
0006  080a  9e
0006  080b  32 30
0006  080f  00
0007  0810  00              basend .byt 0,0
0007  0811  00
0008  0812                        *=2080
0009  0820  a2 00                  ldx #0          ;text ausgeben
0010  0822  bd 78 08        loop    lda text,x
0011  0825  f0 06                  beq notext
0012  0827  20 d2 ff               jsr $ffd2
0013  082a  e8                    inx
0014  082b  d0 f5                  bne loop
0015  082d  a2 08              notext ldx #8           ;dn 8
0016  082f  a0 00                  ldy #0          ;sa 0
0017  0831  20 ba ff               jsr $ffba       ;fileparameter setzen
0018  0834  a9 0b                  lda #texend-texanf ;prgm name
0019  0836  a2 6c                  ldx #<texanf
0020  0838  a0 08                  ldy #>texanf
0021  083a  20 bd ff               jsr $ffbd       ;filenamen setzen
0022  083d  a9 00                  lda #0          ;load-flag
0023  083f  a2 00                  ldx #0          ;laden nach $e000
0024  0841  a0 e0                  ldy ##e0
0025  0843  20 d5 ff               jsr $ffd5       ;load ausführen
0026  0846  a9 a0                  lda ##a0        ;basic in ram kopieren
```


C-64 IEEE Interface
Peter Rubenight
12.3.84
C-64
6522
IEEE-BUS
2x MC 3446
Pin 16: +5V
8, 12: GND
+5V
GND
DAV
NRFD
NDAC
EOI
ATN
DIO8
DIO7
DIO6
DIO5
DIO4
DIO3
DIO2
DIO1
3.3K
6.8K
8 mal
22pF
2434
DATA-CLOCK
14: +5V
7: GND

```
0027  0848  85 af          sta $af
0028  084a  a9 00          lda #0
0029  084c  85 ae          sta $ae
0030  084e  a0 00    floop1 ldy #0
0031  0850  b1 ae    floop2 lda ($ae),y
0032  0852  91 ae          sta ($ae),y
0033  0854  c8             iny
0034  0855  d0 f9          bne floop2
0035  0857  e6 af          inc $af
0036  0859  a5 af          lda $af
0037  085b  c9 c0          cmp #$c0        ;16k kopiert ?
0038  085d  d0 ef          bne floop1
0039  085f  a9 35          lda #$35
0040  0861  85 01          sta $01
0041  0863  20 6a ee       jsr $ee6a       ;neutrl
0042  0866  20 22 e4       jsr $e422       ;einschalttext
0043  0869  4c 7b e3       jmp $e37b       ;ready
0044  086c  4b 45    texanf .byt 'kernal.ieee'
0045  0877  00       texend .byt 0
0046  0878  93       text   .byt $93,$d,'loading kernal.ieee',0
0046  0879  0d
0046  087a  4c 4f
0046  088d  00
0047  088e                   .end

errors = 0000

line#  loc   code          line

0001  0000               ;*********************************
0002  0000               ;* c-64  ieee-488 bus interface *
0003  0000               ;*==============================*
0004  0000               ;* peter porbadnigk    02.09.84 *
0005  0000               ;*********************************
0006  0000
0008  0000
0009  0000               ;definition der portadressen
0010  0000               ;des zusätzlichen   via 6522
0011  0000
0012  0000                       *= $df00        ;i/o2
0013  df00
0014  df00               portb  =*               ;steuerport
0015  df00               porta  =*+1             ;datenport
0016  df00               ddrb   =*+2
0017  df00               ddra   =*+3
0018  df00               acr    =*+11
0019  df00               pcr    =acr+1
0020  df00
0021  df00               ;bitmuster des steuerports
0022  df00               ;=================================
0023  df00               ;bit   funktion      bit   funktion
0024  df00               ;---------------------------------
0025  df00               ; 0    eoi  out       4    eoi  in
0026  df00               ; 1    ndac out       5    ndac in
0027  df00               ; 2    nrfd out       6    nrfd in
0028  df00               ; 3    dav  out       7    dav  in
0029  df00
0030  df00               ;konstanten
0031  df00
0032  df00               unlisn =$3f
0033  df00               untalk =$5f
0034  df00
0035  df00               ;c-64 system adressen
0036  df00
0037  df00               status =$90             ;status byte
0038  df00               lbflag =$94             ;flag für last byte
0039  df00               outreg =$95             ;ausgabe register
```

```
0040  df00
0041  df00                ;die seriellen routinen im c-64 kernal (e000-ffff)
0042  df00                ;werden  unter  einhaltung  der  einsprungadressen
0043  df00                ;gegen  parallele  ieee 488 routinen  ausgetauscht
0045  df00
0046  df00                ;parallele ieee 488 routinen
0047  df00                ;===========================
0048  df00
0049  df00                       *= $ee13
0050  ee13                ;byt vom ieee-port holen
0051  ee13                ;-----------------------
0052  ee13  20 87 ee      iecin  jsr input+5      ;port auf input
0053  ee16  ad 00 df             lda portb
0054  ee19  09 04                ora #%00000100 ;nrfd-h
0055  ee1b  8d 00 df             sta portb
0056  ee1e  a9 ff                lda #255        ;64 ms (timeout)
0057  ee20  8d 05 df             sta portb+5
0058  ee23  2c 0d df      test   bit portb+$d
0059  ee26  70 3a                bvs error
0060  ee28  2c 00 df      w1     bit portb        ;warte dav-l
0061  ee2b  30 f6                bmi test
0062  ee2d  ad 01 df             lda porta
0063  ee30  48                   pha
0064  ee31  ad 00 df             lda portb
0065  ee34  29 0b                and #%00001011 ;nrfd-l
0066  ee36  8d 00 df             sta portb
0067  ee39  18                   clc
0068  ee3a  ad 00 df             lda portb
0069  ee3d  29 10                and #%00010000 ;eoi ???
0070  ee3f  f0 01                beq acc3
0071  ee41  38                   sec
0072  ee42  b0 04         acc3   bcs acc4
0073  ee44  a9 40                lda #%01000000 ;status 64
0074  ee46  85 90                sta status
0075  ee48  ad 00 df      acc4   lda portb
0076  ee4b  09 02                ora #%00000010 ;ndac-h
0077  ee4d  8d 00 df             sta portb
0078  ee50  2c 00 df      w2     bit portb        ;warte for dav-h
0079  ee53  10 fb                bpl w2
0080  ee55  ad 00 df             lda portb
0081  ee58  29 0d                and #%00001101 ;ndac-l
0082  ee5a  8d 00 df             sta portb
0083  ee5d  68                   pla
0084  ee5e  49 ff                eor #$ff
0085  ee60  18                   clc
0086  ee61  60                   rts
0087  ee62
0088  ee62  a9 02         error  lda #2           ;timeout
0089  ee64  85 90                sta status
0090  ee66  a9 0d                lda #13         ;cr
0091  ee68  18                   clc
0092  ee69  60                   rts
0093  ee6a
0094  ee6a                       *= $ed09
0095  ed09                ;zuweisung für talk oder listen
0096  ed09                ;------------------------------
0097  ed09                ;(unlisten und untalk) sowie
0098  ed09                ;kennzeichen des letzen bytes
0099  ed09                ;mit eoi
0100  ed09  09 40         talk   ora #%01000000  ;talk
0101  ed0b  2c                   .byt $2c        ;dummy-byte
0102  ed0c  09 20         listen ora #%00100000  ;listen
0103  ed0e  48            unli   pha
0104  ed0f  ad 00 df             lda portb
0105  ed12  09 06                ora #%00000110 ;nrfd ndac h
0106  ed14  8d 00 df             sta portb
0107  ed17  24 94                bit lbflag      ;bit 7 = 1 (last byte)
0108  ed19  f0 12                beq noeoi
```

```
0109  ed1b  20 a1 ee          jsr eoilo       ;eoi low
0110  ed1e  20 40 ed          jsr send        ;letztes byt senden
0111  ed21  a9 00             lda #0
0112  ed23  85 94             sta lbflag      ;reset last byte flag
0113  ed25  ad 00 df          lda portb
0114  ed28  09 01             ora #%00000001  ;restore eoi
0115  ed2a  8d 00 df          sta portb
0116  ed2d  68          noeoi  pla
0117  ed2e  85 95             sta outreg
0118  ed30  2c 00 df    w8     bit portb       ;warten dav-h
0119  ed33  10 fb             bpl w8
0120  ed35  20 8f ee          jsr atnlo
0121  ed38  4c 40 ed          jmp send
0122  ed3b
0123  ed3b                    *= $edb9
0124  edb9                ;sekundär adresse ausgeben
0125  edb9                ;-------------------------
0126  edb9  85 95       second sta outreg
0127  edbb  20 40 ed          jsr send
0128  edbe  4c 98 ee          jmp atnhi
0129  edc1
0130  edc1                    *= $ee6a
0131  ee6a                ;unterprogramme für bus handling
0132  ee6a                ;--------------------------------
0133  ee6a  a9 ee       neutrl lda ##$ee         ;bus neutral
0134  ee6c  8d 0c df          sta pcr
0135  ee6f  a9 0f             lda ##$0f
0136  ee71  8d 02 df          sta ddrb
0137  ee74  ea                .byt $ea,$ea
0137  ee75  ea
0138  ee76  8d 00 df          sta portb
0139  ee79  a9 ff       output lda ##$ff         ;alle ports ausgang
0140  ee7b  8d 03 df          sta ddra
0141  ee7e  8d 01 df          sta porta
0142  ee81  60                rts
0143  ee82
0144  ee82  a9 09       input  lda #%00001001  ;nrfd ndac-l
0145  ee84  8d 00 df          sta portb
0146  ee87  a9 00             lda #0
0147  ee89  8d 03 df          sta ddra        ;alle ports eingang
0148  ee8c  60                rts
0149  ee8d
0150  ee8d                    *= $ed40
0151  ed40                ;ausgabe eines bytes aus outreg ($95)
0152  ed40                ;-------------------------------------
0153  ed40  20 79 ee    send   jsr output       ;port als ausgang
0154  ed43  ad 00 df          lda portb
0155  ed46  09 08             ora #%00001000 ;dav high
0156  ed48  8d 00 df          sta portb
0157  ed4b  ad 00 df          lda portb
0158  ed4e  29 60             and #%01100000 ;nrfd ndac isolieren
0159  ed50  c9 60             cmp #%01100000 ;nrfd ndac high ??
0160  ed52  f0 31             beq dnperr
0161  ed54  a5 95             lda outreg      ;data byte auf bus
0162  ed56  49 ff             eor ##$ff
0163  ed58  8d 01 df          sta porta
0164  ed5b  ad 00 df    sample lda portb
0165  ed5e  29 60             and #%01100000 ;nrfd ndac
0166  ed60  c9 40             cmp #%01000000 ;nrfd-h???
0167  ed62  d0 f7             bne sample      ;nein dann warten
0168  ed64  ad 00 df          lda portb       ;flip dav
0169  ed67  29 01             and #%00000001 ;eoi nicht ändern
0170  ed69  09 06             ora #%00000110 ;dav low
0171  ed6b  8d 00 df          sta portb
0172  ed6e  ad 00 df    sampl  lda portb
0173  ed71  29 60             and #%01100000 ;nrfd ndac
0174  ed73  c9 20             cmp #%00100000 ;ndac-h ??
0175  ed75  d0 f7             bne sampl       ;nein dann warten
```

```
0176  ed77  ad 00 df          lda portb
0177  ed7a  09 08             ora #%00001000 ;dav high
0178  ed7c  8d 00 df          sta portb
0179  ed7f  a9 ff             lda #$ff       ;bus neutral
0180  ed81  8d 01 df          sta porta
0181  ed84  60                rts
0182  ed85
0183  ed85  a9 80       dmyepr lda #%10000000

0184  ed87  05 90             ora status     ;bit 7 setzen
0185  ed89  85 90             sta status
0186  ed8b  ad 00 df          lda portb
0187  ed8e  09 08             ora #%00001000 ;dav auf high
0188  ed90  8d 00 df          sta portb
0189  ed93  a9 ff             lda #$ff       ;bus neutral
0190  ed95  8d 01 df          sta porta
0191  ed98  60                rts
0192  ed99
0193  ed99                    *= $ee8e
0194  ee8e  60                rts
0195  ee8f  ad 0c df    atnlo lda pcr
0196  ee92  29 fd             and #%11111101
0197  ee94  8d 0c df          sta pcr
0198  ee97  60                rts
0199  ee98
0200  ee98  ad 0c df    atnhi lda pcr
0201  ee9b  09 02             ora #%10
0202  ee9d  8d 0c df          sta pcr
0203  eea0  60                rts
0204  eea1
0205  eea1  ad 00 df    eoilo lda portb      ;eoi  low
0206  eea4  29 fe             and #$fe
0207  eea6  8d 00 df          sta portb
0208  eea9  60                rts
0209  eeaa
0210  eeaa                    *= $edef
0211  edef  ea                .byt $ea,$ea,$ea,$ea,$ea,$ea
0211  edf0  ea
0211  edf1  ea
0211  edf2  ea
0211  edf3  ea
0211  edf4  ea
0212  edf5  ea                .byt $ea,$ea,$ea,$ea,$ea,$ea
0212  edf6  ea
0212  edf7  ea
0212  edf8  ea
0212  edf9  ea
0212  edfa  ea
0213  edfb  a9 5f       untal lda #untalk    ;sende untalk
0214  edfd  2c                .byt $2c
0215  edfe  a9 3f       unlis lda #unlisn    ;sende unlisn
0216  ee00  20 0e ed          jsr unli
0217  ee03  4c 98 ee          jmp atnhi
0218  ee06
0219  ee06                    *= $edc7
0220  edc7  85 95       sectlk sta outreg     ;sec-adr nach talk
0221  edc9  20 40 ed          jsr send       ;sa ausgeben
0222  edcc  20 82 ee          jsr input      ;nrfd und ndac low
0223  edcf  4c 98 ee          jmp atnhi
0224  edd2
0225  edd2              ;bsout
0226  edd2              ;-----
0227  edd2                    *= $ede7
0228  ede7  20 40 ed          jsr send
0229  edea
```

```
0230  edea                  ;basin
0231  edea                  ;------
0232  edea                        *= $f1b5
0233  f1b5  4c 13 ee              jmp iecin
0234  f1b8

0235  f1b8                  ;chkin
0236  f1b8                  ;------
0237  f1b8                        *= $f238
0238  f238  20 09 ed              jsr talk
0239  f23b                        *= $f23f
0240  f23f  20 bd e4              jsr le4bd
0241  f242                        *= $f245
0242  f245  20 c7 ed              jsr sectlk
0243  f248
0244  f248                  ;ckout
0245  f248                  ;------
0246  f248                        *= $f27a
0247  f27a  20 0c ed              jsr listen
0248  f27d                        *= $f281
0249  f281  20 98 ee              jsr atnhi
0250  f284                        *= $f286
0251  f286  20 b9 ed              jsr second
0252  f289
0253  f289                  ;clall
0254  f289                  ;------
0255  f289                        *= $f339
0256  f339  20 fe ed              jsr unlis
0257  f33c                        *= $f340
0258  f340  20 fb ed              jsr untal
0259  f343
0260  f343                  ;open
0261  f343                  ;----
0262  f343                        *= $f3e3
0263  f3e3  20 0c ed              jsr listen
0264  f3e6                        *= $f3ea
0265  f3ea  20 b9 ed              jsr second
0266  f3ed
0267  f3ed                  ;load
0268  f3ed                  ;----
0269  f3ed                        *= $f4cd
0270  f4cd  20 09 ed              jsr talk        ;dev wird talker
0271  f4d0                        *= $f4d2
0272  f4d2  20 c7 ed              jsr sectlk      ;sec adr senden
0273  f4d5  20 13 ee              jsr iecin       ;1. byte holen
0274  f4d8                        *= $f4e0
0275  f4e0  20 13 ee              jsr iecin       ;2. byte (addr)
0276  f4e3                        *= $f501
0277  f501  20 13 ee              jsr iecin       ;get data byte
0278  f504                        *= $f528
0279  f528  20 fb ed              jsr untal       ;ready? sende untalk
0280  f52b
0281  f52b                  ;save
0282  f52b                  ;----
0283  f52b                        *= $f60d
0284  f60d  20 0c ed              jsr listen      ;listen senden
0285  f610                        *= $f612
0286  f612  20 b9 ed              jsr second      ;sec adr senden
0287  f615                        *= $f63f
0288  f63f  20 fe ed              jsr unlis       ;unlisten zum bus
0289  f642                        *= $f648
0290  f648  20 0c ed              jsr listen      ;listen
0291  f64b                        *= $f651
0292  f651  20 b9 ed              jsr second      ;sec. adr. senden
0293  f654  20 fe ed              jsr unlis       ;unlisn - close
0294  f657
0295  f657                  ;reset und start des kernal.ieee
```

```
f657                          ;-------------------------------
f657                          ;erkennung des  rom $8000 wird erweitert
f657                          ;durch  initialisierung des  neuen ports
f657                                *= $fce7
fce7  20 b7 e4                      jsr  le4b7
fcea
fcea                                *= $e4b7
e4b7  20 5a ee        le4b7  jsr neutrl          ;port init
e4ba  4c 02 fd               jmp $fd02          ;test modul $8000
e4bd  20 82 ee        le4bd  jsr input           ;port input
e4c0  4c 98 ee               jmp atnhi
e4c3
e4c3                                *= $e461
e461  42 59                         .byt 'bytes free',$d,0
e46b  0d
e46c  00
e46d
e46d                                *= $e475
e475  43 4f                         .byt 'commodore c-64.ieee'
e488  20 2d                         .byt ' - basic v2.4     ',$d,$d
e499  0d
e49a  0d
e49b
e49b                                *= $e535
e535  00                            .byt 0             ;schwarz - zeichen
e536
e536                                *= $ecd9
ecd9  0e                            .byt 14            ;hellblau - rahmen
ecda  0f                            .byt 15            ;hellgrau - hintergrund
ecdb
ecdb                                .end

errors = 0000

end of assembly
```

□=

Bücher

McGilton, H.; Morgan,R.: Einführung in das UNIX-System. Bei McGraw-Hill, Hamburg 1984, 468 S., DM 58,--, ISBN 3-89028-003-X. Es handelt sich um die deutsche Übersetzung der amerikanischen Vorlage 'Introducing the UNIX System'. Nach zwei einführenden Abschnitten finden wir folgende Kapitel: Verzeichnisse und Dateien - Kommandos und Standard-Dateien - Benutzerkommunikation - Textbearbeitung - Der ED- und der VI-Editor - Gestalten von Texten - Weitere Formatierhilfen - Programmieren der UNIX-Shell - Werkzeuge für die Programmentwicklung - Das UNIX-System von Berkely - UNIX-Systemleitung - Literatur zu UNIX. Es ist ein leichtverständliches Lehrbuch, mit dem man sich einen kompletten Überblick über das UNIX-System verschaffen kann. Es soll vor allem einen Leitfaden für den Gebraucher geben, der sich in den umfangreichen Systemhandbüchern nicht so leicht zurechtfinden kann. Dementsprechend finden wir auf fast allen Seiten ein ein oder mehrere ausgetestete Beispiele für die Hantierung am Terminal, am System. Damit sind viele Tips für eine effektivere Anwendung der einzelnen UNIX-Programme gegeben, die in den Handbüchern nicht geboten werden. - Die Übersetzung hat einen klaren deutschen Text, und auch die druckmäßige Gestaltung ist sorgfältig, so daß man dieses Buch nicht nur zum Lernen, sondern auch zum Experimentieren und Nachschlagen gut gebrauchen kann. Als bisheriger Nicht-Benutzer von UNIX erhält man außerdem einen Eindruck von der Leistungsfähigkeit dieses Betriebssystems, so daß man hier die Kriterien findet, die eine Entscheidung für den Einsatz erleichtern. Mit den vielen beschriebenen Leistungen des Systems finden aber auch Systemprogrammierer einen Katalog wünschenswerter Eigenschaften eines Betriebssystems.

Roland Löhr

68008-Computer

Kurz vor Radaktionsschluß hatte der Autor Gelegenheit, einen Computer mit der CPU 68008 vorübergehend zu testen. Es handelt sich um den Single Board Computer SBC 68K8 der Firma MTC Paul & Scherer. Er verdient Interesse, nicht nur wegen der aktuellen CPU von Motorola mit ihren 32 Bit breiten Registern, sondern auch wegen des zum Computer lieferbaren Betriebssystems CP/M 68K nebst zahlreicher Utilities, wegen des enthaltenen 68000-Assemblers, wegen des Compilers für die Sprache C und wegen des günstigen Preises.

Hardware

Die Platine mißt ca. 232x188 mm. Sie enthält folgende Baugruppen (siehe auch Schemazeichnung): CPU MC68008, mit 8 MHz betrieben, den Floppy Disk Controller WD2793, der bis zu vier Laufwerke ansteuert, zwei serielle Bausteine und Kanäle mit MC6850, V24-Schnittstelle und einstellbarer Baudrate, einer VIA 6522 und zwei PIA 6821. Die VIA kann je nach Konfiguration als Centronics-Schnittstelle dienen oder für den Anschluß einer Tastatur. Eine PIA ist frei, die andere hat Systemfunktionen für die Bedienung der Real Time Clock und für die angekündigte Grafik-Erweiterungskarte. Die Real Time Clock ist vom Typ Seikosha RTC58321 (interessanterweise mit integriertem eigenen Quarz). Sie wird durch einen Akku gepuffert. Wir finden ferner 4 Sockel mit 28 Polen für Festwertspeicher (EPROMs), die maximal 64 Kilobyte aufnehmen können. Ein Sockel nimmt dabei den Urlader für das Betriebssystem auf, der auch Teile des BIOS (Basic Input/Output System) enthält. Interessant ist auch das dynamische RAM, das standardmäßig 256 KB umfaßt. Es ist in Stufen von 64KB und mit hochintegrierten Bausteinen auf maximal 832 KB on board (!) erweiterbar (die Bausteine sind gesockelt). Ein großes RAM hat den Vorteil, das es vom Betriebssystem als RAM-Disk angesprochen werden kann. Das erspart viele Diskettenzugriffe und beschleunigt die meisten Aktionen des Betriebssystems, des Assemblers und des Compilers.

Das Betriebssystem ist disketten-orientiert. Also muß man mindestens ein Laufwerk vorsehen. Standardmäßig wird ein EPSON-Laufwerk SMD 140 für Disketten mit 3 1/2" angeschlossen. Es wird vom BIOS unterstützt, schreibt und liest doppelseitig mit doppelter Dichte und nimmt derzeit etwa 800 KB an formierter Information auf. Die Schnittstelle entspricht 5 1/4". Wahlweise kann auch eine solche für 3 1/2" geschaffen werden für den Anschluß von Sony-Laufwerken. Die Disketten entsprechen damit denjenigen, die z.B. auf dem McIntosh von Apple benutzt werden. Auch eine Schnittstelle für Laufwerke mit 8" ist vorhanden.

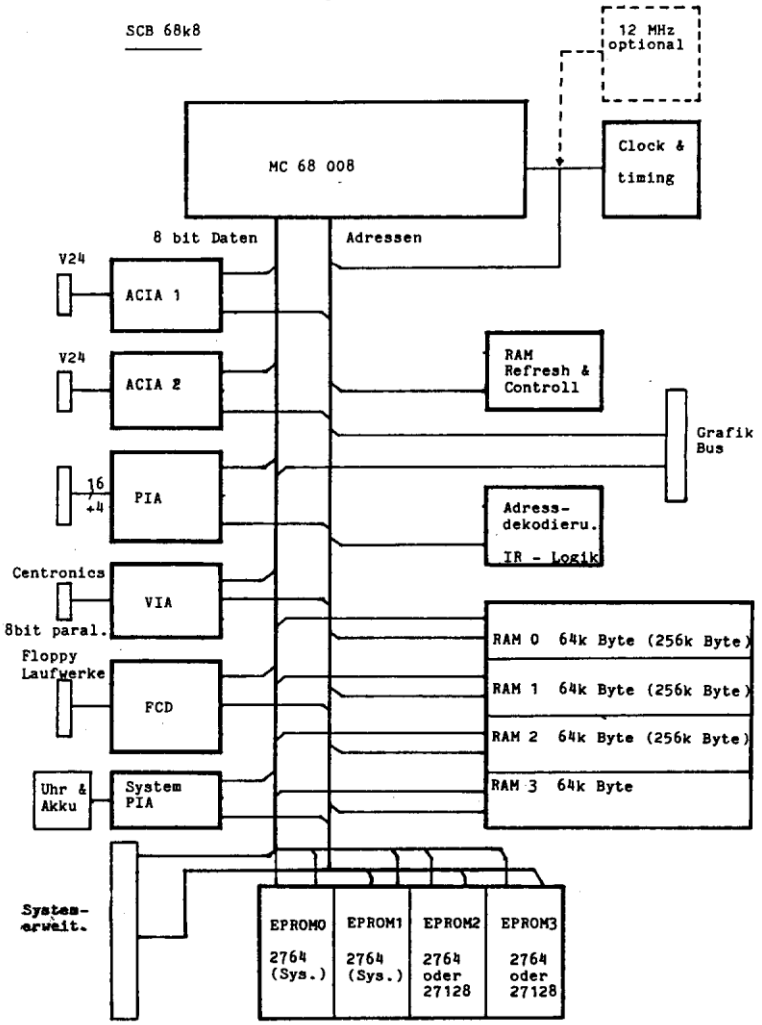
Der Anschluß der Platine geschieht im Nu: Man muß +5V, +12V und -12V vorhalten, ferner ein Terminal mit Bildschirm und Tastatur. Man kann dann mit dem Urlader sofort booten. Es sind jedoch auch andere Systemkonfigurationen möglich, z.B. mit Anschluß einer Tastatur mit Strobe, Anschluß der Grafikkarte 68k8 mit einfarbigem Monitor, Anschluß der Farbgrafikkarte 68k8F und Farb-Monitor. Hier wurde die Karte am Terminal betrieben. Zu den Grafik-Karten wird berichtet, daß sie den Controller NEC7220 enthalten sollen. 16 Graustufen sind möglich, bzw. 8 Farben und eine Textebene. Die Auflösung soll im Format definierbar sein, z.B. 512x512 Pixel oder 720x364. Durch maximale Bestückung des Video-RAMs sollen 1024x1024 Pixel erreichbar sein und insgesamt vier Darstellungsebenen.

Zum Aufbau der CPU-Karte ist zu bemerken, daß die Decodierung auf PALs verzichtet und somit durchschaubar ist. Der Prozessorbus kann für Erweiterungen über eine VG64-Verbindung herausgeführt werden. Auch die Interfacebausteine sind auf Kontaktleisten herausgeführt.

Dem Autor wurden folgende Preise für die Hardware genannt (ohne Gewähr), jeweils inkl. MWSt: Computerplatine mit 256K RAM und Urlader/Boot DM 2.500. EPSON Floppy, wie beschrieben DM 815, Grafik-Karte mit 1 Darstellungsebene DM 900.

Das Betriebssystem CP/M-68K

Das Betriebssystem- und Sprachpaket für die Karte wird gesondert berechnet. Es kostet ca. DM 700 (inkl. MWSt). Man erhält ein für die Kartenkonfiguration angepaßtes BIOS, ein Betriebsystem



mit residenten und transienten Kommandos, wie sie im CP/M für Z80 gebräuchlich sind (siehe dazu weiteren Beitrag in diesem Heft). Wenn man also auf einem anderen CP/M-System gearbeitet hat, braucht man sich nicht umzugewöhnen. Man kann hinsichtlich der Befehle und ihrer Wirkung auf vorhandene Handbücher zurückgreifen. Gleichwohl empfiehlt sich die Anschaffung oder der Besitz der vierteiligen etwa 600-seitigen Dokumentation von Digital Research, des Sammeldordners 'CP/M-68K Operating System'. Denn zum Softwarepaket gehören neben zahlreichen Utilities auch der Editor, der Linker, ein Archivprogramm, ein Assembler für MC68000 mit Motorola-Syntax, ein Compiler für die Sprache C und ein Debugging-Programm.

Mit dieser Software in der Hand wird der Computer SBC 68k8 auch im Grundausbau zu einem hervorragenden Entwicklungssystem für die Programmentwicklung und Erprobung in Assemblersprache und in der Sprache C, die meistens Computer in einer höheren Preisstufe voraussetzt. Diese

strukturierte Sprache hat zudem den Vorteil hoher Portabiltität, so daß auch in 'C' formulierte System- und Anwenderprogramme auf dieser Maschine zur Ausführung gebracht werden können, die für das UNIX-Betriebssystem geschrieben wurden. Ausgenommen sind solche, die unter UNIX die Funktionen fork/exec benutzen oder die File-Strukturen des UNIX lesen.

Dokumentation: Mit der genannten vierteiligen Dokumentation von Digital Research ist die Hantierung des CP/M-68K, des Editors und der Sprachen ausreichend beschrieben. Der Hersteller der Platine (MCT Paul & Scherer, Rostocker Str. 31, 1000 Berlin 21, T.: 030-392 30 11) beabsichtigt, für diesen Software-Teil noch verkürzte, besser überschaubare Übersichten bereitzustellen. Den Lieferungen sollen ferner Datenblätter für die wichtigen Bausteine sowie Anschlußbilder für Signale und Konfigurationen beigefügt werden.

Zusammenfassung

Die Beschäftigung mit den modernen 16 Bit-Systemen, insbesondere Motorola 680xx nimmt jetzt rasch zu. Einige große Zeitschriften veröffentlichen Schaltungen und Bauanleitungen für solche Computer. Hier, wie auch bei der besprochenen Platine, wird der 68008 als CPU eingesetzt, der den vollen Befehlssatz der MC68000 hat, jedoch einen Datenbus mit 8 Bit. Diese CPU stellt damit das Pendant des 8088 im Verhältnis zum 8086 dar. Sie erlaubt einen einfacheren Aufbau eines Computers, der dann entweder Anwendersystem ist oder Entwicklungssystem speziell für 68000-Boards oder noch andere Computer, wenn Cross-Assembler zur Verfügung stehen, wie es sie unter 'C' schon gibt. - In vielen Fällen wird man das mit einem eigenen Computeraufbau aus Bausätzen verbundene Risiko meiden wollen. Die Beherrschung einer Maschine mit 8 MHz Takt mag oft schwierig sein, vor allem wenn geeignete Meßmittel fehlen. In solchen Fällen ist eine fertige Computerkarte wie die SBC 68k8 die Lösung mit mehr Sicherheit. Wenn Netzteil und Terminal (oder ein Computer der als Terminal dienen kann) vorhanden sind, erhält man für ca. DM 3300 die arbeitsfähige Hardware (Computerboard nebst erstem Floppy Drive) sowie für weitere DM 700 das Betriebssystem nebst Assembler und C-Compiler. In der Summe erhält man einen betriebsfertigen Computer, nach gewerblichen Maßstäben gefertigt, mit einem günstigen Preis-/Leistungsverhältnis.

Erste Beobachtungen

Kurz vor Abschluß dieses Heftes stand leider nicht genug Zeit für eine tiefergehende Einarbeitung in die vielen Möglichkeiten des Betriebssystems, seiner Utilities und seiner Sprachen zur Verfügung. Es konnten lediglich einige Stichproben mit einfachen Beispielen durchgeführt werden. Dabei zeigte sich, daß das System ein zügiges Arbeiten ermöglicht, vor allem wenn eine RAM-Disk zur Verfügung steht. Besonderes Interesse fand zunächst einmal das DDT (Dynamic Debugging Tool), ein Hilfsprogramm mit Monitor-Eigenschaften. Es verleiht dem System hohe innere Transparenz zum Zwecke der Programmprüfung und -erprobung. Seine wichtigsten Dienstleistungen sind Anzeige und Setzen von Speicherstellen und Registern, Füllen und Versetzen von Bereichen, Disassembler in Motorola-Syntax, Laden und Starten zur Programmausführung, auch mit Trace und Breakpoints. - Es wurden keine eigenen Programmbeispiele editiert, jedoch auf Diskette vorhandene Quelltexte dem Assembler und dem C-Compiler zur Bearbeitung übergeben und zur Ausführung gebracht, wobei sich erwartungsgemäße Ergebnisse einstellten.

□○□

Dipl.-Ing. Wolfgang Krebs, 1000 Berlin 41

CP/M-68K

Ein leistungsfähiges Betriebssystem für 68000-Programmierer und Anwender

Mit CP/M-68K steht dem Benutzer ein unfangreiches Betriebssystem für die 68000-Familie zur Verfügung. Es ist ein echtes 68000-Betriebssystem, nicht etwa eine Systemergänzung mit einer Z80-Karte. Eine mit CP/M-68K ausgerüstete 68000-Karte bietet dem Programmierer alles, was er zur Programmentwicklung benötigt (Assembler, Editor und 'C'), der Anwender kann seine Pro-

gramme weitgehend hardwareunabhängig einsetzen. - Hier soll ein Überblick über das CP/M-68K gegeben werden, der die Möglichkeiten aufzeigt und die benötigte Systemkonstellation. Und es soll auf einzelne Utilities hingewiesen werden. Anwendern möchte ich die Informationen geben, die nötig sind, um eine Entscheidung über den Einsatz von CP/M-68K zu treffen. Kenntnisse des CP/M für 8080/Z80 sind nicht nötig, sie erleichtern aber sehr den Einstieg in das Betriebssystem.

CP/M-Versionen und Bezugsquellen

Digital Research entwickelte mit CP/M das am weitesten verbreitete Betriebssystem für 8-Bit-Mikrocomputer. Es ist für Achtziger-Prozessoren geschrieben worden und enthält ein umfangreiches Paket an Utilities, wie z.B. Debugger, Editor und Assembler. Es ist in der Maschinensprache des 8080 geschrieben worden. Vor etwa einem Jahr brachte Digital Research dann CP/M-86 (für den 8086) und CP/M-68K auf den Markt. Beide sind zum größten Teil in 'C' geschrieben, und nur prozessorspezifische Besonderheiten unterscheiden sich.

Da Digital Research ein weiteres Betriebssystem für den MC68000 auf den Markt bringen will, betreibt diese Firma keine besonderen Vertriebsanstrengungen, verkauft das CP/M-68K aber weiterhin ab 50 Stück zum Stückpreis von etwa 100 Dollar. Einzeln ist es z.Zt. für etwa 1100 bis 1600 DM zu erhalten. Ich habe es bei den Firmen Eltec, MicroSys, GWK, EMS und Valvo angeboten gesehen. Die Zeitschriften MC und c't entwickeln gerade 68000-Karten mit CP/M-68K und lassen noch preiswertere Angebote auf dem Markt erwarten. Das CP/M-68K wird meines Erachtens wegen der Kompatibilität mit dem Z80-CP/M, wegen der umfangreichen Software-Werkzeuge, des 68000-Assemblers und wegen des C-Compilers als Betriebssystem für den 68000 an Bedeutung gewinnen. Zukünftige Betriebssysteme müssen, um eine umfangreiche Einarbeitung zu ersparen, in weiten Teilen CP/M ähneln, wenn sie nicht gleich übernommen werden.

Das CP/M-68K wird auf fünf Disketten (8", single side, single density) ausgeliefert. Standardmäßig wird das BIOS für das Motorola EXORmacs-Entwicklungssystem mitgeliefert. Nur durch das BIOS (Basic Input/Output System) wird das CP/M-68K an die jeweilige Hardware angepaßt. Oben genannte Firmen bieten meist mit dem CP/M-68K ein BIOS für ihre eigenen 68000-Platinen an. Da CP/M-68K mit S-Records bei der Datenübertragung arbeitet, wird die Übertragung auf ein anderes Zielsystem vereinfacht.

Dokumentation

Mit einem mehr als 600-seitigen Handbuch in Englisch ist das CP/M-68K gut dokumentiert. Es enthält vier Teile:

CP/M-68K Operating System User's Guide
CP/M-68K Operating System Programmers Guide
CP/M-68K Operating System System Guide
The C Programming Guide for CP/M-68K

Das Handbuch erklärt zunächst das Arbeiten mit CP/M-68K auf der Benutzerebene. Mit Beispielen folgen die Anpassung von Benutzerprogrammen unter CP/M-68K, der Editor, der Assembler und Debugger, der C-Compiler und die Anpassung an eine spezielle Hardware. Da die Kommandostruktur mit dem des CP/M für Z80 mit kleinen Ausnahmen gleich ist, empfehle ich eine gute deutschsprachige Einführung in das CP/M für Z80 als Ergänzung. Ich habe mich so z.B. in den Editor eingearbeitet, der voll mit dem des Vorbildes kompatibel ist.

Durch die Arbeit mit CP/M kann das wachsende Problem, daß ein Programmierer sich für jeden neuen Mikroprozessor auf ein neues Betriebssystem einarbeiten muß, beschränkt werden auf den Assembler, spezielle Speicherbelegungen und Eigenarten des BIOS und DDT (Dynamic Debugging Tool). Das CP/M-68K ist weitgehend mit dem für Z80 (V2.2) kompatibel. Mit dem CP/M-86 existiert ein weiteres kompatibles Betriebssystem (ca. DM 300 kostet die billige IBM-PC-Version). Ich hoffe auf einen zukünftigen Erfahrungsaustausch über die Eigenarten der drei CP/M-Systeme an dieser Stelle.

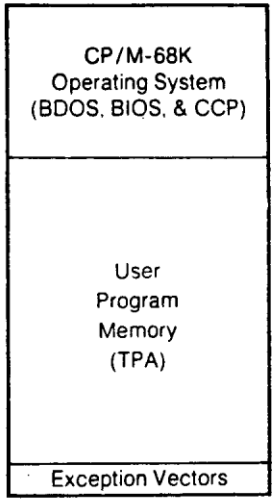
Hardware und Speicherbelegung

Zum Arbeiten mit CP/M-68 werden minimal 64 KByte RAM benötigt. Es werden maximal 512

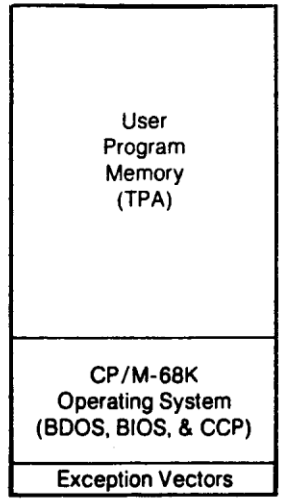
MByte RAM bedient. Zum vernünftigen Arbeiten sollte man jedoch mindestens 128 KB zur Verfügung haben. Ca. 24 KB des RAM werden vom CP/M-68K belegt, abhängig von der Größe des BIOS und damit der Anzahl der Peripheriegeräte. Es können 1 bis 16 Diskettenlaufwerke mit je maximal 512 MByte angeschlossen werden. Für Programmierer ist ein größerer RAM-Bereich zu empfehlen, ebenso das Arbeiten mit einer virtuellen Disk mit einem speziellen BIOS (ist nicht im CP/M enthalten, wird aber auf dem Markt angeboten). Die virtuelle Disk im RAM empfiehlt sich für größere Assembler- und C-Programme, um die Durchlaufzeiten zu verkürzen. Das CP/M-68K benötigt die Exception-Vektoren im RAM-Bereich, d.h. in den Adressen hex 0 bis 3FF. Der TRAP #4 (Adresse 90-93) wird für CP/M-Funktionsaufrufe verwendet. Im RAM-Bereich ab hex 400 wird während des Ladens ein kurzes Boot-Programm von der Diskette geladen.

Das CP/M-68K kann in jedem Adressenbereich über 400 arbeiten. Der Betrieb in der Mitte des RAM-Bereiches ist jedoch nicht sinnvoll, da dieses Betriebssystem einen zusammenhängenden Benutzer-RAM-Bereich (TPA) vorschreibt. Von Digital Research wird empfohlen, das CP/M-68K entweder direkt über die Exception-Vektoren zu legen oder an das obere Ende des Speichers. So ergeben sich typische Speicherbelegungen gem. Bild 1.

Typische Memory-Map mit dem CP/M-68K im oberem Bereich



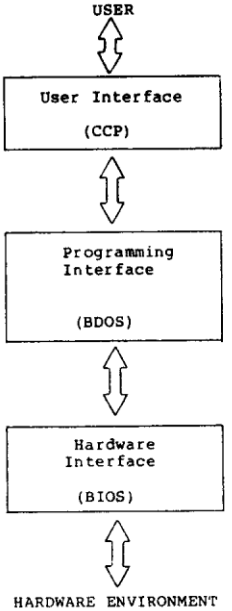
Typische Memory-Map mit dem CP/M-68K im unterem Bereich



Das CP/M-68K ist auf den Mikroprozessoren MC68008, MC68000 und MC68010 getestet worden. Auf dem 32-Bit-Prozessor MC68020 sollte es auch arbeiten, obwohl der Prozessor von Digital Research nicht extra angegeben wird. Die Benutzerprogramme können alle Eigenschaften der MC68000-Familie voll ausnutzen. So können Exceptions und der Supervisor-State in den Benutzerprogrammen verwendet werden. Auch Coprozessoren können in bestimmten Anwendungen eingesetzt werden.

Programm-Module des CP/M-68K

Jedes CP/M besteht aus drei Modulen, dem BIOS, dem BDOS und dem CCP. Der CCP (Console Command Processor) ist die Schnittstelle zum Anwender und enthält den Befehls-Interpreter. Der CCP greift zur Ausführung seiner Befehle auf das BDOS (Basic Disk Operating System) zu. Die gesamte Kommunikation mit der Außenwelt findet über das BDOS statt. Das BIOS (Basic Input/Output System) sorgt für die Anpassung an die jeweilige Hardware. Nur das BIOS unterscheidet sich bei unterschiedlichen Rechnern.



CP/M-68K Interfaces

Im BIOS erfolgt die Hardwareanpassung über mehrere Tabellen. Die Zuordnung der logischen Sektoren zu den physikalischen erfolgt zum Beispiel über die Sample Sector Translation Table, damit das CP/M sehr anpassungfähig an unterschiedlichste Diskettenlaufwerke ist. Will man ein eigenes BIOS implementieren, so ist je nach der Anzahl der I/O-Geräte ca. 8K Maschinenprogramm zu schreiben. Im Handbuch findet man eine gute Hilfe. Es sind ein BIOS in Assembler und ein BIOS in C als Programmierbeispiel erklärt.

Ein Benutzerprogramm muß alle I/O-Operationen über die BDOS-Funktionen abwickeln, wenn es auf Computern mit verschiedener Hardware lauffähig sein soll. Der Funktionscode wird im CPU-Register D0 abgelegt, in D1 und D2 können weitere Parameter übergeben werden. Dann erfolgt der Funktionsaufruf mit TRAP #4. Nach Rückkehr stehen empfangene Daten in D0.

Das BDOS verwendet mit wenigen Ausnahmen die gleichen Funktionscodes wie das Z80-CP/M. 28 = Write Protect Disk fehlt. Es sind jedoch acht weitere Funktionen zusätzlich implementiert:

F#	Function	Type
46	Get Disk Free Space	Drive
47	Chain To Program	System/Program Control
48	Flush Buffers	System/Program Control
50	Direct BIOS Call	System/Program Control
59	Program Load	System/Program Control
61	Set Exception Vector	Exception
62	Set Supervisor State	Exception
63	Get/Set TPA Limits	Exception

Der MC68000-Assembler

Das CP/M-68K enthält den Assembler AS68 für den MC68000. Er entspricht der Motorola-Syntax, wie sie im '16 Bit Microprocessor User's Manual' von Motorola beschrieben ist. Kleine Unter-

schiede bestehen nur bei den Assembler-Direktiven. Für die Arbeit mit dem Assembler werden minimal 128 KByte RAM benötigt. Der Assembler arbeitet schneller als der von Motorola auf dem Exormax (in PASCAL geschrieben). Er kann lauffähigen 68000-Maschinencode erzeugen, ebenso sind unvollständige Objectfiles möglich, denen der Linker erst die endgültigen Adressen zuweist. Aufgabe des Linkers ist das Verbinden mehrerer Objectfiles. Der linker benötigt allerdings viel Zeit.

Der C-Compiler

Der C-Compiler entspricht der Sprachbeschreibung von Kernighan und Ritchie /4/, die lange Zeit die einzige C-Beschreibung war. Mit C wird ein Tor zur UNIX-Welt geöffnet. Die Library ist kompatibel mit UNIX V7, die Ausnahmen sind genau im Handbuch beschrieben. Mit dem C-Compiler ist eine Möglichkeit gegeben, Programme von UNIX zum CP/M-68K zu übertragen. Der C-Compiler benötigt mehrere Durchläufe, die einzeln aufgerufen werden müssen, aber auch durch einen mitgelieferten SUBMIT-File in einem Aufruf gestartet werden können. Dieser Compiler benötigt einige Laufzeit..

Literatur

/1/ CP/M-68K Handbuch, Digital Research
/2/ 16 Bit Microprocessor User's Manual, Motorola MC68000UM(AD3)
/3/ CP/M Handbuch, R. Zaks, Sybex-Verlag
/4/ Programmieren in C, Kernighan/Ritchie, Hanser-Verlag

PS: Das CP/M-68k wird jetzt in der Version 1.2 ausgeliefert. – Auf den nachfolgenden Seiten findet sich eine Übersicht für die von Digital Research auf den Systemdisketten gelieferten Programme.

Table A-1. Distribution Disk Contents

File	Contents
AR68.REL	Relocatable version of the archiver/librarian.
AS68INIT	Initialization file for assembler--see AS68 documentation in the CP/M-68K Operating System Programmer's Guide.
AS68.REL	Relocatable version of the assembler.
ASM.SUB	Submit file to assemble an assembly program with file type .S, put the object code in filename.O, and a listing file in filename.PRN.
BIOS.O	Object file of BIOS for EXORmacs.
BIOS.C	C language source for the EXORmacs BIOS as distributed with CP/M-68K.
BIOSA.O	Object file for assembly portion of EXORmacs BIOS.
BIOSA.S	Source for the assembly language portion of the EXORmacs BIOS as distributed with CP/M-68K.
BIOSTYPS.H	Include file for use with BIOS.C.
BOOTER.O	Object for EXORmacs bootstrap.
BOOTER.S	Assembly boot code for the EXORmacs.
C.SUB	Submit file to do a C compilation. Invokes all three passes of the C compiler as well as the assembler. You can compile a C program with the line: A>C filename.
C068.REL	Relocatable version of the C parser.
C168.REL	Relocatable version of the C code generator.

Table A-1. (continued)

File	Contents
CLIB	The C run-time library.
CLINK.SUB	Submit file for linking C object programs with the C run-time library.
CP68.REL	Relocatable version of the C preprocessor.
CPM.H	Include file with C definitions for CP/M-68K. See the C Programming Guide for CP/M-68K for details.
CPM.REL	Relocatable version of CPM.SYS.
CPM.SYS	CP/M-68K operating system file for the EXORmacs.
CPMLIB	Library of object files for CP/M-68K. See Section 2.
CPMLDR.SYS	The bootstrap loader for the EXORmacs. A copy of this was written to the system tracks using PUTBOOT.
CTYPE.H	Same as above.
DDT.REL	Relocatable version of the preloader for DDT™. (Loads DDT1 into the high end of the TPA.)
DDT1.68K	This is the real DDT that gets loaded into the top of the TPA. It is relocatable even though the file type is .68K, because it must be relocated to the top of the TPA each time it is used.
DUMP.REL	Relocatable version of the DUMP utility.
ED.REL	Relocatable version of the ED utility.
ELDBIOS.S	Assembly language source for the ERG sample loader BIOS.
ERGBIOS.S	Assembly language source for the ERG sample BIOS.
ERRNO.H	Same as above.
FORMAT.REL	Relocatable disk formatter for the Motorola EXORmacs.

Table A-1. (continued)

File	Contents
FORMAT.S	Assembly language source for the FORMAT utility.
INIT.REL	Relocatable version of the INIT utility.
INIT.S	Assembly language source for the INIT utility.
LCPM.SUB	Submit file to create CPM.REL for EXORmacs.
LDBIOS.O	Object file of loader BIOS for EXORmacs.
LDBIOSA.O	Object file for assembly portion of EXORmacs loader BIOS.
LDBIOSA.S	Source for the assembly language portion of the EXORmacs loader BIOS as distributed with CP/M-68K.
LDRLIB	Library of object files for creating a Bootstrap Loader. See Section 3.
LO68.REL	Relocatable version of the linker.
LOADBIOS.H	Include file for use with BIOS.C, to make it into a loader BIOS.
LOADBIOS.SUB	Submit file to create loader BIOS for EXORmacs.
MAKELDR.SUB	Submit file to create CPMLDR.SYS on EXORmacs.
NORMBIOS.H	Include file for use with BIOS.C, to make it into a normal. BIOS
NORMBIOS.SUB	Submit file to create normal BIOS for EXORmacs.
NM68.REL	Relocatable version of the symbol table dump utility.
PIP.REL	Relocatable version of the PIP utility.
PORTAB.H	Same as above.
PUTBOOT.REL	Relocatable version of the PUTBOOT utility.

Table A-1. (continued)

File	Contents
PUTBOOT.S	Assembly language source for the PUTBOOT utility.
README.TXT	ASCII file containing information relevant to this shipment of CP/M-68K. This file might not be present.
RELCPM.SUB	Submit file to relocate CPM.REL into CPM.SYS.
RELOC.REL	Relocatable version of the command file relocation utility.
RELOCx.SUB b	This file is included on each disk that contains .REL command files. (x is the number of the distribution disk containing the files). It is a submit file which will relocate the .REL files for the target system.
S.O	Startup routine for use with C programs--must be first object file linked.
SENDC68.REL	Relocatable version of the S-record creation utility.
SETJMP.H	Same as above.
SIGNAL.H	Same as above.
SIZE68.REL	Relocatable version of the SIZE68 utility.
SR128K.SYS	S-record version of CP/M-68K. This version has no BIOS, and is provided for use in porting CP/M-68K to new hardware.
SR400.SYS	S-record version of CP/M-68K. This version has no BIOS, and is provided for use in porting CP/M-68K to new hardware.
STAT.REL	Relocatable version of the STAT utility.
STDIO.H	Include file with standard I/O definitions for use with C programs. See the C Programming Guide for CP/M-68K for details.

Die im Text enthaltenen Übersichten werden mit freundlicher Genehmigung von Digital Research abgedruckt.

Bücher

Hogan, T.: CP/M Anwenderhandbuch. Bei McGraw-Hill, Hamburg 1984, 304 S., DM 39,80, ISBN 3-89028-005-6. Das Buch hat folgende Gliederung: CP/M und Betriebssysteme - Eingebaute Kommandos - Transiente Kommandos - Assembler-Dienstprogramme - Transiente Programme und CP/M - MP/M, CP/NET und CP/M-Verwandte - Technische Aspekte - Systemkriterien - Anhänge mit der Zusammenfassung der Kommandos, Vergleichen zwischen CP/M-80 und CP/M-86, CP/M-Anforderungen, Disketten-Auswahl, Glossarium usw. In den einzelnen Kapiteln finden wir eine Fülle von Beispielen für die Hantierung und für die Antworten des Systems, auch bei Fehlern. Es werden die neuesten Entwicklungen in der Familie der ähnlichen Betriebssysteme berücksichtigt, einschließlich CP/M-86. Wir finden ferner zahlreiche Tabellen, Übersichten und Hinweise für die verschiedenen Versionen der Betriebssysteme sowie Systemkriterien, die man als möglicher Anwender bei der Auswahl studieren sollte, denn hier spielen stark Fragen der Kompatibilität hinein. Der Reiz von CP/M liegt ja darin, daß man im Prinzip eine große Auswahl an Sprachen und Anwenderprogrammen zur Verfügung hat, gleichwohl gibt es Anpassungsprobleme, nicht zuletzt auch in Bezug auf die Hardware und auf die Diskettenformate. So hat dieses Buch den Vorteil, nicht nur ein Anwenderhandbuch zu sein, mit dem man sich in die Befehle des Betriebssystems einarbeitet, es ist vielmehr auch ein Buch mit Auswahlkriterien, das auch die Geschichte des Betriebssystems und die Entwicklung bei wichtigen Anbieterfirmen aufzeigt.

Dipl.-Ing. Rudolf Kirchgäßner, 6835 Brühl

UNASS Version 3

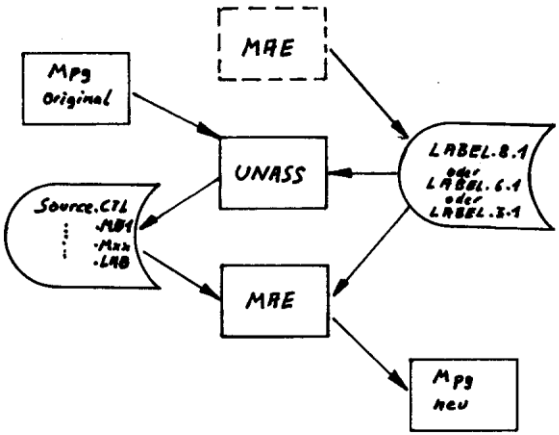
Vom Code zum Text – Ein komfortabler Unassembler für den MAE

Unter diesem Titel hat der Autor bereits in Heft 22 ein Programm vorgestellt. Zwischenzeitlich wurde diese Utility durch Einbindung von Maschinenprogrammen (Mpg) wesentlich beschleunigt (4K in ca. 12 Minuten) und bedienungsfreundlicher gestaltet. Neben Routinen zum Lesen und Schreiben von MAE-Files werden Tabellenroutinen sowie Hex-Dez-Wandelprogramme eingesetzt. Die Vielseitigkeit der Mpg ist durch die SYS-Aufrufe mit Variablenübergabe und durch die Portabiltität auf verschiedene CBM-Systeme gewährleistet.

Datenfluß

Der Datenflußplan gem. Abbildung zeigt die Wirkungsweise von UNASS: UNASS disassembliert ein beliebiges Mpg und generiert daraus Source-Files im MAE-Format. Sprungziele und Speicheradressen werden mit symbolischen Namen versehen. Für häufig benötigte System-Adressen wird ein Label-File angelegt. Als Beispiel sei auf das in den Maschinenroutinen enthaltene UNASS-SYS. LAB verwiesen. Eine bestimmte Betriebssystem-Adresse (Zero Page oder ROM) erhält einen 'sprechenden Namen' und die Angabe des Bezugsbereiches. Hat TOPRAM die Adresse hex 34, so soll 35 nach der Unassemblierung als TOPRAM+1 dargestellt werden; deshalb bezieht sich TOPRAM auf 2 Bytes. Für den SCREEN wäre der Bezugsbereich danach 2048 bzw. 1024. Das Label-File wird vom MAE erstellt und später wieder bei der Assemblierung verwendet. Daher muß das MAE-Format eingehalten werden. Aufsteigende Sortierung und Adressen im Hex-Code sind zwingend, der Bezugsbereich - als Kommentar geschrieben - kann entfallen (=1). Weitere Kommentare sind nicht erlaubt. (Die Adressen für andere Betriebssysteme im Beispiel dienen hier lediglich der Anpassung auf andere Rechnertypen!) Im Interesse einer einigermaßen schnellen Bearbeitung durch den MAE sollte die Zahl der Label 150 nicht überschreiten.

O.a. Labelfile ist spezifisch für das jeweilige Betriebssystem. Durch Bereitstellung mehrerer Files und deren Austausch bei der Assemblierung kann die Umstellung eines Programms auf ein anderes Betriebssystem mit erheblichem Zeitgewinn vorbereitet werden.



UNASS: Datenflußplan

Um bereits bekannte programminterne Adressen mit symbolischen Adressen zu versehen, kann UNASS mittels DATA-Zeilen dem Absolutprogramm angepaßt werden. Diese Adressen werden nach Pass 2 der Unassemblierung genauso wie alle nicht definierten Adressen in SOURCE.LAB festgehalten. Dort müssen sie besonders bei Umstellungen auf einen anderen Rechnertyp manuell weiterbearbeitet werden.

Bedienung

Die Bereichsgrenzen des Objektprogrammes werden UNASS vorgegeben (logic start & end), und zwar markiert nach Programm- und Datenbereichen. Da ein Mpg zum Schutz vorm Überschreiben häufig verschoben werden muß, können Adreßkorrekturen vorgenommen werden (real start). Entsprechend den Grenzen des Bereiches für das Text-File von MAE dürfen keine Module größer 1K gebildet werden.

Eine korrekte Beschreibung des belegten Textbereiches wird erst durch Editierung im MAE z.B. durch ED/$/$/ erreicht. Dies ist allerdings nur notwendig bei dem CTL-Modul, da dieses während einer Assemblerung im Speicher gehalten wird. Andererseits werden unassemblierte Files wohl immer nachgearbeitet werden (müssen).

Änderungen

Bei der Programmeingabe sollte aus Speicherplatzgründen auf alle Remarks verzichtet werden. Eben deshalb wurde das Programm auch so kompakt wie möglich aufgebaut. UNASS kann an verschiedene MAE-Versionen angepaßt werden: Ein großes mit dem MAE erstelltes Source-File wird mit einem Disk-Monitor untersucht. Die ersten 6 Bytes dieses Files ersetzen dann die CHR$-Werte in der Zeile 690. Eine Unassemblierung direkt von Diskette ist möglich, sofern alle PEEKs in Pass1 und Pass 2 durch GET# ersetzt werden. Allerdings ist dann die Untersuchung und Vorgabe der Modulgrenzen unübersichtlich. Am besten bewährt hat sich die Verschiebung des Mpgs in Soft-ROMs, so daß sich Programme bis ca. 12K verarbeiten lassen. Auch läßt sich der C-64 mit seinem durchgehenden RAM bestens einsetzen.

Eine Programmdiskette mit Labels der Betriebssysteme 8032, 3032 und C-64 kann beim Autor bezogen werden (Adresse: Nibelungenstr. 4, 6835 Brühl), und zwar gegen Überweisung von DM 45,-- auf das Postgirokonto Karlsruhe 94 108-752.

Quellen

Zur Historie von UNASS und zur grundsätzlichen Wirkungsweise: Kirchgäßner, R.: Vom Code zum Text, UNASS. 65xx MICRO MAG, Heft 28. – Seer, W.: Der UNASS, ein erster Test. 65xx MICRO MAG, Heft 29.

Maschinenroutinen: Kirchgäßner, R.: Dateiarbeit mit Tabellen. Computer Club Mannheim, Info 1/84. – Kellermann, H. J.: Instring-Funktion für VC 20. Computer Journal Heft 11/12 83.

```
******************                ***************
CBM 8032  08.04.84                UNASS  SEITE:01
******************                ***************

    0 ' SAVE"$0:UNASS",8:REM06.04.84
    1 A$=",8<5DOWN>":N$="NEW<3LEFT><2DOWN>":C$=CHR$(34):
      L$="LOAD"+C$
    2 PRINT"<CLEAR>"L$"UNASS.MPG"C$A$N$
    3 PRINT L$"UNASS.BAS"C$A$"RUN"
    4 POKE 623,19:FOR X=624 TO 628:POKE X,13:NEXT:POKE 158,5
    5 PRINT"<DOWN>(C) BY R.KIRCHGAESSNER 6835 BRUEHL
    6 POKE 144,PEEK(144)+3:POKE 52,176:POKE 53,122:CLR

259 BYTES
```

```
******************                                  ******************
CBM 8032  08.04.84                                  UNASS64  SEITE:01
******************                                  ******************

    0 ' SAVE"$0:UNASS64",8:REM06.04.84
    1 A$=",8<5DOWN>":N$="NEW<3LEFT><2DOWN>":C$=CHR$(34):
      L$="LOAD"+C$
    2 PRINT"<CLEAR>"L$"UNASS64.MPG"C$",8"A$N$
    3 PRINT L$"UNASS64.BAS"C$A$"RUN"
    4 POKE 631,19:FOR X=632 TO 636:POKE X,13:NEXT:POKE 198,5
    5 PRINT"<DOWN>(C) BY R.KIRCHGAESSNER 6835 BRUEHL
    6 POKE 788,PEEK(788)+3:POKE 55,176:POKE 56,122:CLR

 269 BYTES
******************                                  ******************
CBM 8032  08.04.84                                  UNASS.BAS  SEITE:01
******************                                  ******************

   10 GOTO 750:' SAVE"$0:UNASS.BAS",8:REM06.04.84

   20 '   (C) BY R. KIRCHGAESSNER    6835 BRUEHL  NIBELUNGENSTR.4
   30 '   *** BINARY SEARCH  ***
   40 W=0:X=V+I
   50 J=INT((W+X)/2)
   60 IF J=W OR A(J)=Q THEN RETURN:'  J=W -> NOT FOUND
   70 IF A(J)>Q THEN X=J:GOTO 50
   80 W=J:GOTO 50

   90 '   *** INSERT IN LABLE TABLE ***
  100 V=V+I:IF J<>V-I THEN SYS H+21,0,J+I,V,A(0),L%(0),A$(0),ZZ
  110 A(J+I)=Q:RETURN

  120 '   *** TEST FOR ROM-LABEL ***
  130 GOSUB 40:IF Q>A(J)+L%(J) THEN PRINT#8,K$;:SYS D,Q,X$:
      GOTO 160
  140 X$=A$(J):IF X$="" THEN SYS D,A(J),X$:X$=K$+X$
  150 IF J=W THEN PRINT#8,X$"+";:X$=MID$(STR$(Q-A(J)),2)+T$(T):
      RETURN
  160 X$=X$+T$(T):RETURN

  170 '   *** WRITE EMPTY LINE ***
  180 SYS N,Y$:PRINT#8,Y$CHR$(174);:RETURN

  190 '   *** MODULE VALUES ***
  200 SYS H,R$(F),R:SYS H,S$(F),S:SYS H,E$(F),E:RETURN

  210 '   *** BUILD .BY IN WORD FILE ***
  220 Q=R-S+E:FOR J=R TO Q:SYS N,Y$:PRINT#8,Y$" .BY";:W=8:
      IF J+W>Q THEN W=Q-J
  230 X$="":FOR X=J TO J+W:PRINT#8,X$;:SYS D,PEEK(X),X$:
      X$=" $"+X$:NEXT
  240 SYS Z,X$:PRINT#8,X$;:J=J+W:NEXT:RETURN

  250 '   *** PASS 1 ***
  260 PRINT"<RVS>PASS 1<OFF>: BUILD LABEL TABLE":FOR F=I TO G:
      IF RIGHT$(F$(F),2)=".W" THEN 370
  270 GOSUB 200:FOR Q=0 TO E-S:U=R+Q:SYS K,PEEK(U),B,M,P,T:
      IF B=0 THEN 360:'  1 BYTE
```

```
280 IF P=I THEN 350:' IMMEDIATE
290 Q=PEEK(U+I):IF B=2 THEN Q=PEEK(U+2)*Y+Q:GOTO 330
300 IF M>8 THEN 330:' NO BRANCH
310 IF Q>127 THEN Q=Q-Y:'     BRANCH BACK
320 Q=S+Q+2+Q
330 GOSUB 40:IF J=W THEN IF Q<=A(J)+L%(J) THEN 350
340 IF J=W THEN GOSUB 100
350 Q=Q+B
360 NEXT
370 NEXT
380 '  *** PASS 2 ***
390 PRINT"<RVS>PASS 2<OFF>: WRITE MODULE FILE":FOR F=I TO G:
    PRINT F$(F):GOSUB 200:GOSUB 680
400 PRINT#8,Y$" .BA $"S$(F)" ;- $"E$(F)"  "D$;:SYS N,Y$:
    PRINT#8,Y$" .MC $"R$(F)CHR$(160);
410 GOSUB 180:IF RIGHT$(F$(F),2)=".W" THEN GOSUB 220:GOTO 580
420 A=I:C=A(A):FOR Q=0 TO E-S:SYS N,Y$:PRINT#8,Y$;:L=S+Q
430 IF C>L THEN 460
440 IF C<L THEN A=A+I:C=A(A):GOTO 430
450 SYS D,C,X$:PRINT#8,K$X$;:S%(A)=-I
460 U=R+Q:SYS K,PEEK(U),B,M,P,T:IF M=0 THEN X$="":Q=PEEK(U):
    PRINT#8," .BY ";:GOTO 550
470 IF B=0 THEN X$=" "+C$(M):GOTO 560
480 Q=PEEK(U+I):IF B=2 THEN Q=PEEK(U+2)*Y+Q:GOTO 520
490 IF M>8 THEN 520
500 IF Q>127 THEN Q=Q-Y
510 Q=S+Q+2+Q
520 PRINT#8," "C$(M)" "P$(P);:IF P<>I THEN GOSUB 130:GOTO 560
530 IF Q<10 THEN X$=CHR$(Q+48):GOTO 560
540 IF Q>31 THEN IF Q<92 THEN PRINT#8,"'";:X$=CHR$(Q):GOTO 560
550 PRINT#8,"$";:SYS D,Q,X$
560 SYS Z,X$:PRINT#8,X$;:IF M>56 THEN GOSUB 180
570 Q=Q+B:NEXT
580 GOSUB 720:NEXT
590 '  *** WRITE LABEL FILE ***
600 PRINT"WRITE PROGRAM LABEL FILE":F=0:GOSUB 680:PRINT#8,Y$
    ";  "D$;:GOSUB 180
610 '  .DE = DEFINE EXTERNAL LABEL
620 FOR J=I TO V:IF A$(J)>"" OR S%(J) THEN 640
630 SYS N,Y$:SYS D,A(J),Q$:PRINT#8,Y$K$Q$" .DE $";:SYS Z,Q$:
    PRINT#8,Q$;
640 NEXT:GOSUB 180:SYS H+9,1300:' RESTORE TO EXPLICIT DEFINED
    LABELS
650 READ X$:IF X$="*" THEN GOSUB 720:PRINT CHR$(7)CHR$(7):
    RETURN
660 SYS N,Y$:READ Q$:Q$=Q$+"":SYS Z,Q$:PRINT#8,Y$X$" .DE $"Q$;:
    READ J:GOTO 650

670 '  *** MODUL START ***
680 OPEN 8,8,8,"$"+Q$+":"+F$(F)+",P,W
690 PRINT#8,CHR$(170)CHR$(0)CHR$(48)CHR$(135)CHR$(77)CHR$(170);
700 SYS N,Y$0:PRINT#8,Y$";PU "CHR$(34)"$0:"F$(F)CHR$(162);:
    GOSUB 180:SYS N,Y$:RETURN

710 '  *** MODUL END ***
720 SYS N,Y$:PRINT#8,Y$";        ================="CHR$(160);
730 X$=CHR$(0):SYS N,Y$:PRINT#8,Y$X$X$X$;:CLOSE 8:RETURN
```

```
740 '   *** INITIALIZATION ***
750 J=1000:X=0:Q=0:W=0:V=0:I=1:X$="":C=0:B=0:Y$="":Y=256
760 H=31408:D=H+3:N=H+6:K=H+18:Z=H+24:K$="
765 ' BEI C 64: POKE 785,188:POKE 786,122
770 POKE I,188:POKE 2,122:PRINT"<CLEAR><RVS>UNASSEMBLER FOR
    650X    VER 3<DOWN>
780 INPUT"<DOWN>DATUM:";D$:IF LEN(D$)=0 OR LEN(D$)>8 THEN 780
790 SYS Z,D$:INPUT"<DOWN>OUTPUT DISK IN DRIVE  1<3LEFT>";O$:
    IF O$<>"0" AND O$<>"1" THEN 790
800 INPUT"<DOWN>LABEL-FILE: <RVS>3.1<OFF>, <RVS>6.1<OFF>,  <RVS>
    8.1<OFF><5LEFT>";L$:IF LEN(L$)<>3 THEN 800
810 INPUT"PREFIX FOR LABELS: Z<3LEFT>";K$:IF LEN(K$)=
    0 OR LEN(K$)>3 THEN 810
820 INPUT"<CLEAR>PROGRAM-NAME";F$:IF LEN(F$)>13 THEN 820
830 INPUT"<DOWN># OF MODULES";G:IF G>20 THEN 830
840 PRINT"<DOWN>REAL    LOGIC     PROGRAM=P";PRINT"START START
    END  DATAFILE=.W<DOWN>
850 DIM A(J),L%(J),S%(J),A$(J),C$(60),P$(3),T$(5),F$(20),S$(20)
    ,E$(20),R$(20)
860 T$(0)="";T$(1)=",X)";T$(2)="),Y";T$(3)=",X";T$(4)=",Y":
    T$(5)=")
870 P$(0)="";P$(1)="#";P$(2)="(";P$(3)="*";F$(0)=F$+".LAB":
    FOR F=I TO G
880 INPUT R$(F),S$(F),E$(F),X$:F$(F)=F$+".M"+RIGHT$("0"
    +MID$(STR$(F),2),2)
890 IF X$=".W" THEN F$(F)=F$(F)+".W":GOTO 910
900 IF X$<>"P" OR E$(F)<S$(F) THEN PRINT"<UP>";:GOTO 880
910 IF LEN(R$(F))<>4 OR LEN(S$(F))<>4 OR LEN(E$(F))<>
    4 THEN PRINT"<UP>";:GOTO 880
920 IF F=G THEN 950
930 SYS H,R$(F),R:SYS H,S$(F),S:SYS H,E$(F),E:E=E+I:
    SYS D,E-S+R,X$:SYS D,E,Y$
940 PRINT"  "X$","Y$",,,,,P":PRINT"<UP>";
950 NEXT
960 '   *** READ MNEMONICS, CODES AND LABELS **
970 PRINT"<DOWN>READ DATAS";FOR J=0 TO 60:READ C$(J):NEXT
980 Q=-1:OPEN 2,8,2,"LABEL."+L$:FOR J=I TO 6:GET#2,X$:NEXT
990 SYS H+15,2,Y$:IF ASC(Y$)=0 THEN CLOSE 2:GOTO 1070
1000 X$=LEFT$(Y$,I):IF X$<"$" OR X$>"Z" THEN 990
1010 V=V+I:J=USR(" ")Y$:A$(V)=LEFT$(Y$,J-I):Y$=MID$(Y$,J+6)
1020 J=USR(";")Y$:IF J=0 THEN SYS H,Y$,X:GOTO 1040
1030 SYS H,LEFT$(Y$,J-2),X:L%(V)=VAL(MID$(Y$,J+I))-I
1040 IF X>Q THEN A(V)=X:Q=X:GOTO 990
1050 PRINT"ERROR IN LABEL  FILE":END

1060 '   *** NEW DEFINED LABELS ***
1070 READ X$:IF X$="*" THEN 1130
1080 READ Y$:SYS H,Y$,Q:READ C:GOSUB 40
1090 IF J=W THEN GOSUB 100:L%(J+I)=C-I:A$(J+I)=X$:GOTO 1070
1100 PRINT X$"! SYSTEM LABEL WILL BE OVERWRITTEN  N<3LEFT>";
1110 INPUT Y$:IF Y$="Y" THEN A$(J)=X$:L%(J)=C-I
1120 GOTO 1070

1130 PRINT"WRITE CONTROL FILE":Y$="..":F=G+I:F$(F)=F$+".CTL":
     GOSUB 680
1140 PRINT#8,Y$" .CT ;"D$;:'  CONTROL TABLE
1150 SYS N,Y$:PRINT#8,Y$" .O"CHR$(211);:' OBJECT MODULE STORE
1160 SYS N,Y$:PRINT#8,Y$" .C"CHR$(197);:GOSUB 180:' CONTINUE IF
     ERRORS
```

```
1170 FOR F=0 TO G:SYS N,Y$:PRINT#8,Y$" .FI "CHR$(34)F$(F)
     CHR$(162):NEXT
1180 SYS N,Y$:PRINT#8,Y$" .FI "CHR$(34)"LABEL."L$CHR$(162):
1190 GOSUB 180:SYS N,Y$:PRINT#8,Y$" .E"CHR$(206):GOSUB 730:
     GOSUB 260
1200 '   *** MNEMONICS ***
1210 DATA,BEQ,BNE,BCC,BCS,BMI,BPL,BVC:' 1.ELEMENT = "" !!
1220 DATA BVS,LDA,STA,LDX,STX,LDY,STY,CMP
1230 DATA CPX,CPY,INC,DEC,JSR,AND,ADC,SBC
1240 DATA ORA,EOR,BIT,ASL,LSR,ROL,ROR
1250 DATA INX,INY,DEX,DEY,CLC,CLD,SEC,SED
1260 DATA PLA,PHA,TAX,TXA,TAY,TYA,TSX,TXS,CLI
1270 DATA SEI,PHP,PLP,CLV,NOP,"ASL A","LSR A","ROL A","ROR A"
1280 DATA JMP,RTS,RTI,BRK
1290 '   *** EXPLICIT DEFINED LABELS ***
1300 '    LABEL,HEXADDRESS,LENGTH
1310 DATA*:' SIGN FOR LAST LABEL
```

```
4869 BYTES
```

```
                    0005            .LS
                    0010 ;PU "$0:UNASS64.CTL"
                    0020
                    0030            .BA $7AB0           ;06.04.84
                    0040            .OS
                    0050            .CT
                    0060            .CE
                    0070
                    0080 VERBASIC   .DE 8
                    0090
7AB0- 4C 92 7C      0100            JMP HEXDEC
7AB3- 4C F1 7C      0110            JMP DECHEX
7AB6- 4C DA 7A      0120            JMP ZEILENNR
7AB9- 4C 4F 7D      0130            JMP RESTOREZ
7ABC- 4C 6D 7D      0140            JMP INSTRING
7ABF- 4C 0D 7B      0150            JMP READMAE
7AC2- 4C B4 7D      0160            JMP CODEANA
7AC5- 4C 5E 7B      0170            JMP INSDELARY
                    0180
                    0190            .FI "UNASS-SYS.LAB"
```

```
0786  2DDB-3561  UNASS-SYS.LAB

                    0010 ;PU "$0:UNASS-SYS.LAB"
                    0020
                    0030            ;06.04.84
                    0040
                    0050 ;ADRESSEN: BASIC 4          BEREICH  BASIC 3   C 64
                    0060 ;===================================  ==========
                    0070 TYPFLG     .DE $08             ;      $08       $0E
                    0080 LINUM      .DE $11             ;2     $11       $14
                    0090 STRADR     .DE $1F             ;2     $1F       $22
                    0100 VARTAB     .DE $2A             ;      $2A       $2D
                    0110 DATPTR     .DE $3E             ;2     $3E       $41
                    0120 VARNAM     .DE $42             ;2     $42       $45
                    0130 VARADR     .DE $44             ;2     $44       $47
                    0140 FORPNT     .DE $46             ;2     $46       $49
                    0150 FAC3       .DE $54             ;1     $54       $57
                    0160 LINADR     .DE $5C             ;2     $5C       $5F
                    0170 FAC1       .DE $5E             ;6     $5E       $61
                    0180 FAC2       .DE $66             ;6     $66       $69
                    0190 ARISGN     .DE $6C             ;      $6C       $6F
                    0200 CHRGET     .DE $70             ;6     $70       $73
                    0210 CHRGOT     .DE $76             ;      $76       $79
                    0220 STATUS     .DE $96             ;      $96       $90
                    0230 PRTY       .DE $B1             ;      $B1       $9B
```

```
                0240 FNLEN      .DE $D1                 ;       $D1     $B7
                0250 LF         .DE $D2                 ;       $D2     $B8
                0260 DN         .DE $D4                 ;       $D4     $BA
                0270 FNADR      .DE $DA                 ;2      $DA     $BB
                0280 INBUF      .DE $0200               ;80     $0200   $0200
                0290 ERROC      .DE $B3CF               ;       $C357   $A437
                0300 SBAL       .DE $B5A3               ;       $C52C   $A613
                0310 UNDFERR    .DE $B86E               ;       $C7EB   $A8E3
                0320 LET        .DE $B930               ;       $C8AD   $A9A5
                0330 ARGREA     .DE $BD84               ;       $CC8B   $AD8A
                0340 STRTYP     .DE $BD89               ;       $CC90   $AD8F
                0350 ARGUM      .DE $BD98               ;       $CC9F   $AD9E
                0360 KOMMA      .DE $BEF5               ;       $CDF8   $AEFD
                0370 SNERR      .DE $BF00               ;       $CE03   $AF08
                0380 FVAR       .DE $C12B               ;       $CF6D   $B08B
                0390 QUANTERR   .DE $C373               ;       $D123   $B24B
                0400 INTFP      .DE $C4BC               ;       $D26D   $B391
                0410 POS        .DE $C4C9               ;       $D27A   $B3A0
                0420 STRINI     .DE $C59E               ;       $D34F   $B475
                0430 STRLIT     .DE $C5B0               ;       $D361   $B487
                0440 ARGSTR     .DE $C7B5               ;       $D57D   $B6A3
                0450 LEN        .DE $C8B2               ;       $D656   $B77C
                0460 ARGBYT     .DE $C8D4               ;       $D678   $B79E
                0470 GETBYTK    .DE $C927               ;       $D6CC   $B7F1
                0480 FPINT      .DE $C92D               ;       $D6D2   $B7F7
                0490 MEMADD     .DE $C99D               ;       $D773   $B867
                0500 FAC1MEM    .DE $CD0D               ;       $DAE3   $B8D7
                0510 SGNFAC1    .DE $CD61               ;       $DB37   $B82B
                0520 ADRFP      .DE $CD7F               ;       $DB55   $B849
                0530 CHKIN      .DE $FFC6
                0540 CLRCH      .DE $FFCC
                0550 GETIN      .DE $FFE4
                0560
                0200            .FI "WORKMAE.ASM"


0612  2DDB-33ED  WORKMAE.ASM

                0010 ;PU "$0:WORKMAE.ASM"
                0020
                0030 ; SETZT BIT 7 IM LETZTEN BYTE VON A$     06.04.84
                0040 ; AUFRUF: SYS SETBIT7 ,A$
                0050
7AC8- 20 F5 BE  0060 SETBIT7    JSR KOMMA
7ACB- 20 98 BD  0070            JSR ARGUM
7ACE- 20 B5 C7  0080            JSR ARGSTR
7AD1- A8        0090            TAY                     ;LAENGE
7AD2- 88        0100            DEY
7AD3- B1 1F     0110            LDA (STRADR),Y
7AD5- 09 80     0120            ORA #$80
7AD7- 91 1F     0130            STA (STRADR),Y
7AD9- 60        0140            RTS
                0150            ;==========
                0160 ; ERZEUGT ZEILENNUMMERN FUER MAE
                0170 ; AUFRUF: SYS ZEILENNR ,X$
                0180 ; X$ MUSS MIT 2 BYTE LAENGE DEFINIERT SEIN
                0190
7ADA- 20 F5 BE  0200 ZEILENNR   JSR KOMMA
7ADD- 20 98 BD  0210            JSR ARGUM
7AE0- 20 B5 C7  0220            JSR ARGSTR
7AE3- C9 02     0230            CMP #2                  ;LAENGE 2 BYTE ?
7AE5- F0 03     0240            BEQ INITIAL?
7AE7- 4C 00 BF  0250            JMP SNERR
                0260
7AEA- A0 00     0270 INITIAL?   LDY #0
7AEC- 20 76 00  0280            JSR CHRGOT
7AEF- C9 30     0290            CMP #'0
7AF1- D0 09     0300            BNE ADD10
7AF3- 20 70 00  0310            JSR CHRGET              ;BEI 0 WIRD ZEILEN# ..
7AF6- 98        0320            TYA                     ;.. AUF 0 GESETZT
7AF7- 91 1F     0330            STA (STRADR),Y
7AF9- C8        0340            INY
7AFA- D0 0D     0350            BNE STORE               ;IMMER
```

```
                    0360
7AFC- B1 1F         0370 ADD10      LDA (STRADR),Y
7AFE- 18            0380            CLC
7AFF- F8            0390            SED
7B00- 69 0A         0400            ADC #10             ;INCREMENT
7B02- 91 1F         0410            STA (STRADR),Y
7B04- C8            0420            INY
7B05- B1 1F         0430            LDA (STRADR),Y
7B07- 69 00         0440            ADC #0
7B09- 91 1F         0450 STORE      STA (STRADR),Y
7B0B- D8            0460            CLD
7B0C- 60            0470            RTS
                    0480            ;============
                    0490 ;   LIEST STRING AUS MAE-FILE
                    0500 ;   AUFRUF: SYS READMAE, A,B$
                    0510 ;         A : LOGISCHE FILENUMMER
                    0520 ;        B$ : STRINGVARIABLE
                    0530
7B0D- 20 F5 BE      0540 READMAE    JSR KOMMA
7B10- 20 D4 C8      0550            JSR ARGBYT          ;LOGISCHE FILE#
7B13- 20 C6 FF      0560            JSR CHKIN
7B16- 20 F5 BE      0570            JSR KOMMA
7B19- 20 2B C1      0580            JSR FVAR            ;A$
7B1C- 20 89 BD      0590            JSR STRTYP
7B1F- 85 46         0600            STA *FORPNT
7B21- 84 47         0610            STY *FORPNT+1
7B23- 20 E4 FF      0620            JSR GETIN           ;ZEILEN# UEBERLESEN
7B26- 20 E4 FF      0630            JSR GETIN           ;    "          "
7B29- A0 00         0640            LDY #0
7B2B- 20 E4 FF      0650 RMLOOP1    JSR GETIN           ;STRING EINLESEN
7B2E- A6 96         0660            LDX *STATUS
7B30- F0 05         0670            BEQ RMLOOP11
7B32- A9 00         0680            LDA #0
7B34- A8            0690            TAY
7B35- F0 09         0700            BEQ RMEND1          ;FILE-ENDE
                    0710
7B37- AA            0720 RMLOOP11   TAX                 ;LESEN BIS BIT 7 GESETZT
7B38- 30 06         0730            BMI RMEND1          ;SATZ-ENDE
7B3A- 99 00 02      0740            STA INBUF,Y         ;ZWISCHENSPEICHERN
7B3D- C8            0750            INY
7B3E- D0 EB         0760            BNE RMLOOP1
                    0770
7B40- 29 7F         0780 RMEND1     AND #$7F
7B42- 99 00 02      0790            STA INBUF,Y
7B45- C8            0800            INY
7B46- 98            0810            TYA                 ;STRINGLAENGE
7B47- 48            0820            PHA
7B48- 20 9E C5      0830            JSR STRINI          ;PLATZ BEREITSTELLEN
7B4B- 68            0840            PLA
7B4C- A8            0850            TAY
7B4D- B9 00 02      0860 RMLOOP2    LDA INBUF,Y         ;STRING UMKOPIEREN
7B50- 91 5F         0870            STA (FAC1+1),Y
7B52- 88            0880            DEY
7B53- 10 F8         0890            BPL RMLOOP2
7B55- 20 F3 C5      0900            JSR STRLIT+67       ;POINTER SETZEN
7B58- 20 65 B9      0910            JSR LET+53
7B5B- 4C CC FF      0920            JMP CLRCH
                    0930
                    0210            .FI "INS/DEL.ASM"
101C  2DDB-3DF7   INS/DEL.ASM

                    0010 ;PU "$0:INS/DEL.ASM"
                    0020
                    0030 ;LOESCHEN BZW PLATZ SCHAFFEN FUER EIN ELEMENT
                    0040 ;IN ARRAYS      VERSION 1    06.11.83
                    0050 ; VGL DEMO-PROGRAMM VON TORONTO PET USER GROUP
                    0060
                    0070
                    0080 ; SYS INSDELARY,SW,PO,EN,A(0)<,...>,ZZ
                    0090
                    0100 ; SW  0=PLATZ FUER EIN ELEMENT SCHAFFEN
                    0110 ;     1=EIN ELEMENT LOESCHEN
```

```
                    0120 ; PO  POSITION DES ELEMENTES
                    0130 ; EN  LETZTES BESETZTES ELEMENT IN DER TABELLE BEI SW=0
                    0140 ;          "          "          "     "     " +1   "  SW=1
                    0150 ; A(0),B$(0),CZ(0),.... LISTE VON ARRAYS
                    0160 ; ZZ  ENDE-KENNZEICHEN
                    0170
                    0180 ; ELEMENT 0 DER ARRAYS DARF NICHT BESETZT WERDEN !
                    0190
                    0200 ;PROGRAMMADRESSEN:
                    0210 ;=================
                    0220 POSITION1. .DE FAC3
                    0230 LASTELEM1  .DE FAC3+2
                    0240 HILFPT     .DE FAC3+4
                    0250 LENARYELEM .DE LINADR
                    0260 LENHIFE    .DE LINADR+1
                    0270 STRINGPT   .DE FAC2
                    0280 POSITION   .DE FNLEN
                    0290 AKTUELEM   .DE FNADR
                    0300 SWITCH     .DE DN
                    0310 LASTELEM   .DE PRTY
                    0320
7B5E- 20 F5 BE      0330 INSDELARY  JSR KOMMA
7B61- 20 D4 C8      0340            JSR ARGBYT          ;BETRIEBSART
7B64- 86 D4         0350            STX #SWITCH
7B66- 20 57 7C      0360            JSR GETPARAM        ;ELEMENTPOSITION
7B69- A5 11         0370            LDA #LINUM
7B6B- 85 D1         0380            STA #POSITION
7B6D- A5 12         0390            LDA #LINUM+1
7B6F- 85 D2         0400            STA #POSITION+1
7B71- 20 57 7C      0410            JSR GETPARAM        ;TABELLENENDE
7B74- A5 11         0420            LDA #LINUM
7B76- 85 B1         0430            STA #LASTELEM
7B78- A5 12         0440            LDA #LINUM+1
7B7A- 85 B2         0450            STA #LASTELEM+1
7B7C- 20 57 7C      0460 NEXTARRAY  JSR GETPARAM        ;ARRAY-NAME 1.ELEMENT
7B7F- 38            0470            SEC
7B80- A5 42         0480            LDA #VARNAM
7B82- E9 5A         0490            SBC #"Z             ;ENDE-KENNZEICHEN
7B84- A5 43         0500            LDA #VARNAM+1
7B86- E9 5A         0510            SBC #"Z
7B88- D0 01         0520            BNE MAINPROC
7B8A- 60            0530            RTS                 ;ZUM INTERPRETER
                    0540
7B8B- A2 04         0550 MAINPROC   LDX #4               ;REAL-ARRAY
7B8D- A9 80         0560            LDA #$80
7B8F- 24 43         0570            BIT #VARNAM+1
7B91- F0 02         0580            BEQ MAINPROC1
7B93- A2 02         0590            LDX #2              ;STRING
7B95- 24 42         0600 MAINPROC1  BIT #VARNAM
7B97- F0 02         0610            BEQ MAINPROC2
7B99- A2 01         0620            LDX #1              ;INTEGER
7B9B- 86 5C         0630 MAINPROC2  STX #LENARYELEM
7B9D- A5 44         0640            LDA #VARADR
7B9F- 85 DA         0650            STA #AKTUELEM
7BA1- A5 45         0660            LDA #VARADR+1
7BA3- 85 DB         0670            STA #AKTUELEM+1
7BA5- A5 D1         0680            LDA #POSITION
7BA7- 85 54         0690            STA #POSITION1
7BA9- A5 D2         0700            LDA #POSITION+1
7BAB- 85 55         0710            STA #POSITION1+1
7BAD- A5 B1         0720            LDA #LASTELEM
7BAF- 85 56         0730            STA #LASTELEM1
7BB1- A5 B2         0740            LDA #LASTELEM+1
7BB3- 85 57         0750            STA #LASTELEM1+1
7BB5- A5 D4         0760            LDA #SWITCH
7BB7- D0 4A         0770            BNE DELET           ;LOESCHEN
7BB9- 18            0780 INSERT     CLC                 ;PLATZ SCHAFFEN
7BBA- A5 DA         0790            LDA #AKTUELEM       ;LETZTES ELEMENT ...
7BBC- 65 56         0800            ADC #LASTELEM1      ;... ADRESSIEREN
```

```
7BBE- 85 DA      0810             STA #AKTUELEM
7BC0- A5 DB      0820             LDA #AKTUELEM+1
7BC2- 65 57      0830             ADC #LASTELEM1+1
7BC4- 85 DB      0840             STA #AKTUELEM+1
7BC6- CA         0850             DEX
7BC7- 10 F0      0860             BPL INSERT
7BC9- 18         0870 INSLOOP     CLC                    ;1 ELEMENT ZURUECK

7BCA- A5 DA      0880             LDA #AKTUELEM
7BCC- E5 5C      0890             SBC #LENARYELEM        ;- CARRY
7BCE- 85 58      0900             STA #HILFPT
7BD0- A5 DB      0910             LDA #AKTUELEM+1
7BD2- E9 00      0920             SBC #0
7BD4- 85 59      0930             STA #HILFPT+1
7BD6- A4 5C      0940             LDY #LENARYELEM        ;A(N+1)=A(N)
7BD8- B1 58      0950 INSLOOP1    LDA (HILFPT),Y
7BDA- 91 DA      0960             STA (AKTUELEM),Y
7BDC- 88         0970             DEY
7BDD- 10 F9      0980             BPL INSLOOP1
                 0990             IFE VERBASIC-8         ;NUR BEI BASIC 4
7BDF- 20 60 7C   1000             JSR PTDESC             ;POINTER UMHAENGEN
                 1010             ***
7BE2- A5 56      1020             LDA #LASTELEM1         ;LASTELEM1-1
7BE4- D0 02      1030             BNE INSLOOP2
7BE6- C6 57      1040             DEC #LASTELEM1+1
7BE8- C6 56      1050 INSLOOP2    DEC #LASTELEM1
7BEA- A5 56      1060             LDA #LASTELEM1         ;POSITION ERREICHT?
7BEC- C5 54      1070             CMP #POSITION1
7BEE- D0 06      1080             BNE INSLOOP3
7BF0- A5 57      1090             LDA #LASTELEM1+1
7BF2- C5 55      1100             CMP #POSITION1+1
7BF4- F0 55      1110             BEQ NULLSETZEN         ;TAB ABGEARBEITET
7BF6- 18         1120 INSLOOP3    CLC
7BF7- A5 DA      1130             LDA #AKTUELEM          ;N=N-1
7BF9- E5 5C      1140             SBC #LENARYELEM        ;CARRY GESETZT
7BFB- 85 DA      1150             STA #AKTUELEM
7BFD- B0 CA      1160             BCS INSLOOP
7BFF- C6 DB      1170             DEC #AKTUELEM+1
7C01- D0 C6      1180             BNE INSLOOP            ;IMMER
                 1190
7C03- 18         1200 DELET       CLC                    ;ZU LOESCHENDES ELEMENT ..
7C04- A5 DA      1210             LDA #AKTUELEM          ;.. ADRESSIEREN
7C06- 65 54      1220             ADC #POSITION1
7C08- 85 DA      1230             STA #AKTUELEM
7C0A- A5 DB      1240             LDA #AKTUELEM+1
7C0C- 65 55      1250             ADC #POSITION1+1
7C0E- 85 DB      1260             STA #AKTUELEM+1
7C10- CA         1270             DEX
7C11- 10 F0      1280             BPL DELET
7C13- 38         1290 DELLOOP     SEC                    ;HILFPT AUF NEXT EL
7C14- A5 DA      1300             LDA #AKTUELEM
7C16- 65 5C      1310             ADC #LENARYELEM        ;+CARRY
7C18- 85 58      1320             STA #HILFPT
7C1A- A5 DB      1330             LDA #AKTUELEM+1
7C1C- 69 00      1340             ADC #0
7C1E- 85 59      1350             STA #HILFPT+1
7C20- A4 5C      1360             LDY #LENARYELEM
7C22- B1 58      1370 DELLOOP1    LDA (HILFPT),Y         ;A(N)=A(N+1)
7C24- 91 DA      1380             STA (AKTUELEM),Y
7C26- 88         1390             DEY
7C27- 10 F9      1400             BPL DELLOOP1
                 1410             IFE VERBASIC-8         ;NUR BEI BASIC 4
7C29- 20 60 7C   1420             JSR PTDESC             ;POINTER UMHAENGEN
                 1430             ***
7C2C- E6 54      1440             INC #POSITION1
7C2E- D0 02      1450             BNE DELLOOP2
7C30- E6 55      1460             INC #POSITION1+1
7C32- A5 56      1470 DELLOOP2    LDA #LASTELEM1
7C34- C5 54      1480             CMP #POSITION1         ;ENDE ERREICHT ?
7C36- D0 06      1490             BNE DELLOOP3
7C38- A5 57      1500             LDA #LASTELEM1+1
7C3A- C5 55      1510             CMP #POSITION1+1
7C3C- F0 0D      1520             BEQ NULLSETZEN         ;TAB ABGEARBEITET
```

```
7C3E- A5 DA     1530 DELLOOP3   LDA #AKTUELEM          ;N=N+1
7C40- 38        1540            SEC
7C41- 65 5C     1550            ADC #LENARYELEM        ;+ CARRY
7C43- 85 DA     1560            STA #AKTUELEM
7C45- 90 CC     1570            BCC DELLOOP
7C47- E6 DB     1580            INC #AKTUELEM+1
7C49- D0 C8     1590            BNE DELLOOP            ;IMMER
                1600
7C4B- A9 00     1610 NULLSETZEN LDA #0                 ;ELEMENT NULLSETZEN
7C4D- A4 5C     1620            LDY #LENARYELEM
7C4F- 91 58     1630 NULLSETZ1  STA (HILFPT),Y
7C51- 88        1640            DEY
7C52- 10 FB     1650            BPL NULLSETZ1
7C54- 4C 7C 7B  1660 NULLRTS    JMP NEXTARRAY
                1670
7C57- 20 F5 BE  1680 GETPARAM   JSR KOMMA              ;KOMMA UND ..
7C5A- 20 98 BD  1690 GETPARAM1  JSR ARGUM              ;PARAMETER HOLEN
7C5D- 4C 2D C9  1700            JMP FPINT              ;WANDELN UND RTS
                1710
                1720            IFE VERBASIC-B         ;NUR BEI BASIC 4
7C60- A5 5C     1730 PTDESC     LDA #LENARYELEM        ;RUECKWAERTSPOINTER
7C62- C9 02     1740            CMP #2                 ;.. BERICHTIGEN
7C64- D0 2B     1750            BNE PTDESCRTS
7C66- A0 00     1760            LDY #0                 ;ALSO STRINGELEMENT
7C68- B1 DA     1770            LDA (AKTUELEM),Y
7C6A- F0 25     1780            BEQ PTDESCRTS          ;LAENGE = 0
7C6C- 85 5D     1790            STA #LENHIFE
7C6E- C8        1800            INY
7C6F- B1 DA     1810            LDA (AKTUELEM),Y
7C71- 85 66     1820            STA #STRINGPT
7C73- C8        1830            INY
7C74- B1 DA     1840            LDA (AKTUELEM),Y
7C76- 85 67     1850            STA #STRINGPT+1
7C78- C5 2B     1860            CMP #VARTAB+1          ;STRING IM BASICTEXT ?
7C7A- 90 15     1870            BCC PTDESCRTS          ;JA
7C7C- F0 02     1880            BEQ PTDESC1            ;VIELLEICHT
7C7E- B0 06     1890            BCS PTDESC2            ;NEIN
                1900
7C80- A5 66     1910 PTDESC1    LDA #STRINGPT
7C82- C5 2A     1920            CMP #VARTAB
7C84- 90 0B     1930            BCC PTDESCRTS          ;ALSO DOCH
7C86- A4 5D     1940 PTDESC2    LDY #LENHIFE           ;POINTER BERICHTIGEN
7C88- A5 DA     1950            LDA #AKTUELEM
7C8A- 91 66     1960            STA (STRINGPT),Y
7C8C- C8        1970            INY
7C8D- A5 DB     1980            LDA #AKTUELEM+1
7C8F- 91 66     1990            STA (STRINGPT),Y
7C91- 60        2000 PTDESCRTS  RTS
                2010            ***
                0220            .FI "HEXDEC.ASM"


05BD  2DDB-3398  HEXDEC.ASM

                0010 ;PU "$0:HEXDEC.ASM"
                0020
                0030 ; AUFRUF:  SYS HEXDEC ,A$,X    06.04.84
                0040
                0050 HXLOWINT   .DE LINADR
                0060 HXHIGHINT  .DE LINADR+1
                0070
7C92- 20 F5 BE  0080 HEXDEC     JSR KOMMA
7C95- 20 98 BD  0090            JSR ARGUM              ;A$
7C98- 20 B5 C7  0100            JSR ARGSTR
7C9B- 86 1F     0110            STX #STRADR
7C9D- 84 20     0120            STY #STRADR+1
7C9F- A0 00     0130            LDY #0
7CA1- 84 5D     0140            STY #HXHIGHINT         ;VORBESETZEN
7CA3- AA        0150            TAX
7CA4- D0 03     0160            BNE HX2                ;LAENGE = 0
7CA6- 4C 73 C3  0170 HX1        JMP QUANTERR
                0180
```

```
7CA9- 4A          0190 HX2        LSR A
7CAA- B0 FA       0200            BCS HX1             ;UNGERADE
7CAC- C9 03       0210            CMP #3
7CAE- B0 F6       0220            BCS HX1             ;> 2
7CB0- 4A          0230            LSR A
7CB1- B0 05       0240            BCS HX3             ;2 STELLEN
7CB3- 20 D7 7C    0250            JSR HX5
7CB6- 85 5D       0260            STA $HXHIGHINT
7CB8- 20 D7 7C    0270 HX3        JSR HX5
7CBB- A8          0280            TAY                 ;LOW
7CBC- A5 5D       0290            LDA $HXHIGHINT      ;HIGH
7CBE- 20 BC C4    0300            JSR INTFP
7CC1- 20 61 CD    0310            JSR $GNFAC1
7CC4- 10 07       0320            BPL HX4
7CC6- A9 EC       0330            LDA #L,HX6
7CC8- A0 7C       0340            LDY #H,HX6
7CCA- 20 9D C9    0350            JSR MEMADD
7CCD- 20 F5 BE    0360 HX4        JSR KOMMA
7CD0- 20 2B C1    0370            JSR FVAR            ;X
7CD3- AA          0380            TAX
7CD4- 4C 0D CD    0390            JMP FAC1MEM
                  0400
7CD7- B1 1F       0410 HX5        LDA (STRADR),Y      ;LINKES HALBBYTE
7CD9- 20 3A 7D    0420            JSR HEXIT1
7CDC- 0A          0430            ASL A
7CDD- 0A          0440            ASL A
7CDE- 0A          0450            ASL A
7CDF- 0A          0460            ASL A
7CE0- 85 5C       0470            STA $HXLOWINT
7CE2- C8          0480            INY
7CE3- B1 1F       0490            LDA (STRADR),Y      ;RECHTES ..
7CE5- 20 3A 7D    0500            JSR HEXIT1
7CE8- 05 5C       0510            ORA $HXLOWINT
7CEA- C8          0520            INY
7CEB- 60          0530            RTS
                  0540
7CEC- 91 00 00    0550 HX6        .BY $91 0 0 0 0
7CEF- 00 00
                  0560            ;==========
                  0570 ; AUFRUF:  SYS DECHEX ,X,A$    A$ = 2 ODER 4 STELLEN
7CF1- 20 F5 BE    0580 DECHEX     JSR KOMMA
7CF4- 20 84 BD    0590            JSR ARGREA          ;X
7CF7- 20 2D C9    0600            JSR FPINT
7CFA- 20 F5 BE    0610            JSR KOMMA
7CFD- 20 2B C1    0620            JSR FVAR            ;A$
7D00- 20 89 BD    0630            JSR STRTYP
7D03- 85 46       0640            STA $FORPNT
7D05- 84 47       0650            STY $FORPNT+1
7D07- A9 02       0660            LDA #2              ;LEN = 2 VORBESETZEN
7D09- A6 12       0670            LDX $LINUM+1
7D0B- F0 01       0680            BEQ DC1             ;< 256
7D0D- 0A          0690            ASL A               ;* 2 => LEN 4
7D0E- 20 9E C5    0700 DC1        JSR STRINI
7D11- A0 00       0710            LDY #0
7D13- A5 12       0720            LDA $LINUM+1
7D15- F0 03       0730            BEQ DC2             ;< 256
7D17- 20 25 7D    0740            JSR DC3
7D1A- A5 11       0750 DC2        LDA $LINUM
7D1C- 20 25 7D    0760            JSR DC3
7D1F- 20 F3 C5    0770            JSR STRLIT+67
7D22- 4C 65 B9    0780            JMP LET+53
                  0790
7D25- 48          0800 DC3        PHA                 ;WANDELT BYTE IN HEX
7D26- 4A          0810            LSR A
7D27- 4A          0820            LSR A
7D28- 4A          0830            LSR A
7D29- 4A          0840            LSR A
7D2A- 20 45 7D    0850            JSR ASCII1
7D2D- 91 5F       0860            STA (FAC1+1),Y
7D2F- C8          0870            INY
7D30- 68          0880            PLA
7D31- 29 0F       0890            AND #$0F
```

```
7D33- 20 45 7D  0900            JSR ASCII1
7D36- 91 5F     0910            STA (FAC1+1),Y
7D38- C8        0920            INY
7D39- 60        0930            RTS
                0940
7D3A- C9 3A     0950 HEXIT1     CMP #$3A
7D3C- 08        0960            PHP
7D3D- 29 0F     0970            AND #$0F
7D3F- 28        0980            PLP
7D40- 90 02     0990            BCC HEXIT11
7D42- 69 08     1000            ADC #8
7D44- 60        1010 HEXIT11    RTS
                1020
7D45- 18        1030 ASCII1     CLC
7D46- 69 F6     1040            ADC #$F6
7D48- 90 02     1050            BCC ASCII11
7D4A- 69 06     1060            ADC #6
7D4C- 69 3A     1070 ASCII11    ADC #$3A
7D4E- 60        1080            RTS
                1090
                0230            .FI "RESTOREZ.ASM"

00F8  2DDB-2ED3  RESTOREZ.ASM

                0010 ;PU "$0:RESTOREZ.ASM"
                0020
                0030 ;AUFRUF:  SYS RESTOREZ, ZEILENNR     06.04.84
                0040
7D4F- 20 F5 BE  0050 RESTOREZ   JSR KOMMA
7D52- 20 84 BD  0060            JSR ARGREA
7D55- 20 2D C9  0070            JSR FPINT
7D58- 20 A3 B5  0080            JSR SBAL
7D5B- B0 03     0090            BCS RE1
7D5D- 4C 6E B8  0100            JMP UNDFERR
                0110
7D60- A5 5C     0120 RE1        LDA *LINADR
7D62- E9 01     0130            SBC #1
7D64- 85 3E     0140            STA *DATPTR
7D66- A5 5D     0150            LDA *LINADR+1
7D68- E9 00     0160            SBC #0
7D6A- 85 3F     0170            STA *DATPTR+1
7D6C- 60        0180            RTS
                0190
                0240            .FI "INSTRING.ASM"


024E  2DDB-3029  INSTRING.ASM

                0010 ;PU "$0:INSTRING.ASM"
                0020
                0030 ; AUFRUF: POSITION = USR (SUCH$) QUELL$   06.04.84
                0040 ; NACH COMPUTER JOURNAL 11/12 1983
                0050
7D6D- 20 B5 C7  0060 INSTRING   JSR ARGSTR
7D70- 85 D1     0070            STA *FNLEN
7D72- 86 DA     0080            STX *FNADR
7D74- 84 DB     0090            STY *FNADR+1
7D76- 20 76 00  0100            JSR CHRGOT
7D79- 20 98 BD  0110            JSR ARGUM
7D7C- 20 B8 C8  0120            JSR LEN+6
7D7F- F0 30     0130            BEQ Z0388
7D81- C8        0140            INY
7D82- 84 61     0150            STY *FAC1+3
7D84- A4 D1     0160            LDY *FNLEN
7D86- F0 29     0170            BEQ Z0388
7D88- A5 1F     0180            LDA *STRADR
7D8A- D0 02     0190            BNE Z0365
7D8C- C6 20     0200            DEC *STRADR+1
7D8E- C6 1F     0210 Z0365      DEC *STRADR
7D90- C6 61     0220 Z0367      DEC *FAC1+3
7D92- A5 61     0230            LDA *FAC1+3
7D94- C5 D1     0240            CMP *FNLEN
```

```
7D96- B0 04     0250            BCS Z0373
7D98- A0 00     0260            LDY #0
7D9A- F0 15     0270            BEQ Z0388
7D9C- E6 1F     0280 Z0373      INC *STRADR
7D9E- D0 02     0290            BNE Z0379
7DA0- E6 20     0300            INC *STRADR+1
7DA2- E8        0310 Z0379      INX
7DA3- A4 D1     0320            LDY *FNLEN
7DA5- 88        0330            DEY
7DA6- B1 DA     0340 Z037D      LDA (FNADR),Y
7DA8- D1 1F     0350            CMP (STRADR),Y
7DAA- D0 E4     0360            BNE Z0367
7DAC- 88        0370            DEY
7DAD- 10 F7     0380            BPL Z037D
7DAF- 8A        0390            TXA
7DB0- A8        0400            TAY
7DB1- 4C CB C4  0410 Z0388      JMP POS+2
                0420
                0250            .FI "CODEANALYSE.ASM"


0C3E  2DDB-3A19  CODEANALYSE.ASM

                0010 ;PU "$0:CODEANALYSE.ASM"
                0020
                0030 ; SYS CODEANA,CODE,LAENGE,MNEMO,PREFIX,SUFFIX  06.04.84
                0040
7DB4- 20 27 C9  0050 CODEANA    JSR GETBYTK         ;BEFEHLSCODE
7DB7- 86 D1     0060            STX *FNLEN
7DB9- BD FA 7D  0070            LDA TAB1,X
7DBC- 48        0080            PHA
7DBD- 29 C0     0085            AND #$C0
7DBF- F0 0A     0090            BEQ LEN0
7DC1- 29 40     0100            AND #$40
7DC3- D0 03     0110            BNE LEN2
7DC5- A9 01     0120            LDA #1
7DC7- 2C        0130            .BY $2C             ;BIT ABSOLUT
7DC8- A9 02     0140 LEN2       LDA #2
7DCA- 2C        0150            .BY $2C             ;BIT ABSOLUT
7DCB- A9 00     0160 LEN0       LDA #0
7DCD- 20 E4 7D  0170            JSR WERT            ;LAENGE
7DD0- 68        0180            PLA
7DD1- 29 3F     0190            AND #$3F
7DD3- 20 E4 7D  0200            JSR WERT            ;MNEMOCODE
7DD6- A6 D1     0210            LDX *FNLEN
7DD8- BD FA 7E  0220            LDA TAB2,X
7DDB- 48        0230            PHA
7DDC- 29 03     0240            AND #$03
7DDE- 20 E4 7D  0250            JSR WERT            ;PREFIX
7DE1- 68        0260            PLA
7DE2- 4A        0270            LSR A
7DE3- 4A        0280            LSR A               ;SUFFIX
7DE4- 85 60     0290 WERT       STA *FAC1+2         ;LOW
7DE6- A9 00     0300            LDA #0
7DE8- 85 5F     0310            STA *FAC1+1         ;HIGH
7DEA- A2 90     0320            LDX #$90
7DEC- 38        0330            SEC
7DED- 20 7F CD  0340            JSR ADRFP
7DF0- 20 F5 BE  0350            JSR KOMMA
7DF3- 20 2B C1  0360            JSR FVAR
7DF6- AA        0370            TAX
7DF7- 4C 0D CD  0380            JMP FAC1MEM         ;WERT ZUWEISEN
                0390
                0400 ; BIT :  7 6      5 4 3 2 1 0
                0410 ;        LAENGE     MNEMOCODE
                0420
7DFA- 3C 98 00  0430 TAB1       .BY $3C $98 $00 $00 $00 $98 $98 $00
7DFD- 00 00 98
7E00- 98 00
7E02- 31 98 35  0440            .BY $31 $98 $35 $00 $00 $D8 $DB $00
7E05- 00 00 D8
7E08- DB 00
```

```
7E0A- 86 98 00  0450          .BY $86 $98 $00 $00 $00 $98 $9B $00
7E0D- 00 00 98
7E10- 9B 00
7E12- 23 D8 00  0460          .BY $23 $D8 $00 $00 $00 $D8 $DB $00
7E15- 00 00 D8
7E18- DB 00
7E1A- D4 95 00  0470          .BY $D4 $95 $00 $00 $9A $95 $9D $00
7E1D- 00 9A 95
7E20- 9D 00
7E22- 32 95 37  0480          .BY $32 $95 $37 $00 $DA $D5 $DD $00
7E25- 00 DA D5
7E28- DD 00
7E2A- 85 95 00  0490          .BY $85 $95 $00 $00 $00 $95 $9D $00
7E2D- 00 00 95
7E30- 9D 00
7E32- 25 D5 00  0500          .BY $25 $D5 $00 $00 $00 $D5 $DD $00
7E35- 00 00 D5
7E38- DD 00
7E3A- 3B 99 00  0510          .BY $3B $99 $00 $00 $00 $99 $9C $00
7E3D- 00 00 99
7E40- 9C 00
7E42- 28 99 36  0520          .BY $28 $99 $36 $00 $F9 $D9 $DC $00
7E45- 00 F9 D9
7E48- DC 00
7E4A- 87 99 00  0530          .BY $87 $99 $00 $00 $00 $99 $9C $00
7E4D- 00 00 99
7E50- 9C 00
7E52- 2F D9 00  0540          .BY $2F $D9 $00 $00 $00 $D9 $DC $00
7E55- 00 00 D9
7E58- DC 00
7E5A- 3A 96 00  0550          .BY $3A $96 $00 $00 $00 $96 $9E $00
7E5D- 00 00 96
7E60- 9E 00
7E62- 27 96 38  0560          .BY $27 $96 $38 $00 $F9 $D6 $DE $00
7E65- 00 F9 D6
7E68- DE 00
7E6A- 88 96 00  0570          .BY $88 $96 $00 $00 $00 $96 $9E $00
7E6D- 00 00 96
7E70- 9E 00
7E72- 30 D6 00  0580          .BY $30 $D6 $00 $00 $00 $D6 $DE $00
7E75- 00 00 D6
7E78- DE 00
7E7A- 00 8A 00  0590          .BY $00 $8A $00 $00 $8E $8A $8C $00
7E7D- 00 8E 8A
7E80- 8C 00
7E82- 22 00 2A  0600          .BY $22 $00 $2A $00 $CE $CA $CC $00
7E85- 00 CE CA
7E88- CC 00
7E8A- 83 8A 00  0610          .BY $83 $8A $00 $00 $8E $8A $8C $00
7E8D- 00 8E 8A
7E90- 8C 00
7E92- 2C CA 2E  0620          .BY $2C $CA $2E $00 $00 $CA $00 $00
7E95- 00 00 CA
7E98- 00 00
7E9A- 8D 89 8B  0630          .BY $8D $89 $8B $00 $8D $89 $8B $00
7E9D- 00 8D 89
7EA0- 8B 00
7EA2- 2B 89 29  0640          .BY $2B $89 $29 $00 $CD $C9 $CB $00
7EA5- 00 CD C9
7EA8- CB 00
7EAA- 84 89 00  0650          .BY $84 $89 $00 $00 $8D $89 $8B $00
7EAD- 00 8D 89
7EB0- 8B 00
7EB2- 33 C9 2D  0660          .BY $33 $C9 $2D $00 $CD $C9 $CB $00
7EB5- 00 CD C9
7EB8- CB 00
7EBA- 91 8F 00  0670          .BY $91 $8F $00 $00 $91 $8F $93 $00
7EBD- 00 91 8F
7EC0- 93 00
7EC2- 20 8F 21  0680          .BY $20 $8F $21 $00 $D1 $CF $D3 $00
7EC5- 00 D1 CF
7EC8- D3 00
```

```
7ECA- B2 8F 00   0690            .BY $B2 $8F $00 $00 $00 $8F $93 $00
7ECD- 00 00 8F
7ED0- 93 00
7ED2- 24 CF 00   0700            .BY $24 $CF $00 $00 $00 $CF $D3 $00
7ED5- 00 00 CF
7ED8- D3 00
7EDA- 90 97 00   0710            .BY $90 $97 $00 $00 $90 $97 $92 $00
7EDD- 00 90 97
7EE0- 92 00
7EE2- 1F 97 34   0720            .BY $1F $97 $34 $00 $D0 $D7 $D2 $00
7EE5- 00 D0 D7
7EE8- D2 00
7EEA- 81 97 00   0730            .BY $81 $97 $00 $00 $00 $97 $92 $00
7EED- 00 00 97
7EF0- 92 00
7EF2- 26 D7 00   0740            .BY $26 $D7 $00 $00 $00 $D7 $D2 $00
7EF5- 00 00 D7
7EF8- D2 00
                 0750
                 0751 ;  BIT:   4 3 2    1 0
                 0752 ;         SUFFIX  PREFIX
                 0753
7EFA- 00 06 00   0760 TAB2       .BY $00 $06 $00 $00 $00 $03 $03 $00
7EFD- 00 00 03
7F00- 03 00
7F02- 00 01 00   0770            .BY $00 $01 $00 $00 $00 $00 $00 $00
7F05- 00 00 00
7F08- 00 00
7F0A- 00 0A 00   0775            .BY $00 $0A $00 $00 $00 $0F $0F $00
7F0D- 00 00 0F
7F10- 0F 00
7F12- 00 10 00   0780            .BY $00 $10 $00 $00 $00 $0C $0C $00
7F15- 00 00 0C
7F18- 0C 00
7F1A- 00 06 00   0790            .BY $00 $06 $00 $00 $03 $03 $03 $00
7F1D- 00 03 03
7F20- 03 00
7F22- 00 01 00   0800            .BY $00 $01 $00 $00 $00 $00 $00 $00
7F25- 00 00 00
7F28- 00 00
7F2A- 00 0A 00   0810            .BY $00 $0A $00 $00 $00 $0F $0F $00
7F2D- 00 00 0F
7F30- 0F 00
7F32- 00 10 00   0820            .BY $00 $10 $00 $00 $00 $0C $0C $00
7F35- 00 00 0C
7F38- 0C 00
7F3A- 00 06 00   0830            .BY $00 $06 $00 $00 $00 $03 $03 $00
7F3D- 00 00 03
7F40- 03 00
7F42- 00 01 00   0840            .BY $00 $01 $00 $00 $00 $00 $00 $00
7F45- 00 00 00
7F48- 00 00
7F4A- 00 0A 00   0850            .BY $00 $0A $00 $00 $00 $0F $0F $00
7F4D- 00 00 0F
7F50- 0F 00
7F52- 00 10 00   0860            .BY $00 $10 $00 $00 $00 $0C $0C $00
7F55- 00 00 0C
7F58- 0C 00
7F5A- 00 06 00   0870            .BY $00 $06 $00 $00 $00 $03 $03 $00
7F5D- 00 00 03
7F60- 03 00
7F62- 00 01 00   0880            .BY $00 $01 $00 $00 $16 $00 $00 $00
7F65- 00 16 00
7F68- 00 00
7F6A- 00 0A 00   0890            .BY $00 $0A $00 $00 $00 $0F $0F $00
7F6D- 00 00 0F
7F70- 0F 00
7F72- 00 10 00   0900            .BY $00 $10 $00 $00 $00 $0C $0C $00
7F75- 00 00 0C
7F78- 0C 00
7F7A- 00 06 00   0910            .BY $00 $06 $00 $00 $03 $03 $03 $00
7F7D- 00 03 03
```

```
7FB0- 03 00
7FB2- 00 00 00   0920          .BY $00 $00 $00 $00 $00 $00 $00 $00
7FB5- 00 00 00
7FB8- 00 00
7FBA- 00 0A 00   0930          .BY $00 $0A $00 $00 $0F $0F $13 $00
7FBD- 00 0F 0F
7F90- 13 00
7F92- 00 10 00   0940          .BY $00 $10 $00 $00 $00 $0C $00 $00
7F95- 00 00 0C
7F98- 00 00
7F9A- 01 06 01   0950          .BY $01 $06 $01 $00 $03 $03 $03 $00
7F9D- 00 03 03
7FA0- 03 00
7FA2- 00 01 00   0960          .BY $00 $01 $00 $00 $00 $00 $00 $00
7FA5- 00 00 00
7FA8- 00 00
7FAA- 00 0A 00   0970          .BY $00 $0A $00 $00 $0F $0F $13 $00
7FAD- 00 0F 0F
7FB0- 13 00
7FB2- 00 10 00   0980          .BY $00 $10 $00 $00 $0C $0C $10 $00
7FB5- 00 0C 0C
7FB8- 10 00
7FBA- 01 06 00   0990          .BY $01 $06 $00 $00 $03 $03 $03 $00
7FBD- 00 03 03
7FC0- 03 00
7FC2- 00 01 00   1000          .BY $00 $01 $00 $00 $00 $00 $00 $00
7FC5- 00 00 00
7FC8- 00 00
7FCA- 00 0A 00   1010          .BY $00 $0A $00 $00 $00 $0F $0F $00
7FCD- 00 00 0F
7FD0- 0F 00
7FD2- 00 10 00   1020          .BY $00 $10 $00 $00 $00 $0C $0C $00
7FD5- 00 00 0C
7FD8- 0C 00
7FDA- 01 06 00   1030          .BY $01 $06 $00 $00 $03 $03 $03 $00
7FDD- 00 03 03
7FE0- 03 00
7FE2- 00 01 00   1040          .BY $00 $01 $00 $00 $00 $00 $00 $00
7FE5- 00 00 00
7FE8- 00 00
7FEA- 00 0A 00   1050          .BY $00 $0A $00 $00 $00 $0F $0F $00
7FED- 00 00 0F
7FF0- 0F 00
7FF2- 00 10 00   1060          .BY $00 $10 $00 $00 $00 $0C $0C $00
7FF5- 00 00 0C
7FF8- 0C 00
                 1070
                 0260
                 0270          .EN

ENDE VON MAE PASS !
```

Karl-Anton Dichtel, 7800 Freiburg

Disassembler für 6809

in Microsoft-BASIC, speziell für AIM 65

Das folgende Programm wurde für den AIM 65 mit FX-80 Drucker umgeschrieben und korrigiert. Es benutzt soweit als möglich die Zeilen-Nummern des Originals, so daß die dort vorhandenen Kommentare weiterhin gültig sind (siehe 65xx MICRO MAG Heft 24/1982).

Wer nur den AIM-Thermodrucker zur Verfügung hat, kann Zeile 1220 aufteilen und durch zweimaligen Durchlauf des Programms zwei Papierstreifen zum Nebeneinanderkleben erhalten.

1.Fehlerkorrektur:
Zeile 350: $10 muß $11 heißen
Zeile 510,570: Zahl für Operationstypen=0 falls diese Operationscodes nicht verwendet
Zeile 670 Es fehlt die Zahl 6 für den Operationstyp 6STS
Zeile 930 OE<48 statt OE<49
Zeile 940 Prebyte statt Postbyte
Zeile 1990 muß 2100 heißen, gleichzeitig Zeile 2100 in 2105 umbenennen, sonst falsche Berechnung bei 5 Bit Programmzähler-Offset bei Offset C,D

2.Änderungen für Anpassung an AIM-65 Basic
-Zeilenlänge maximal 60 Zeichen, Aufspaltung von Zeilen ist erforderlich
-Befehl HEX$( ) fehlt, Simmulation durch H$=.. in Zeile 700
-OPEN und CLOSE entfallen
-INKEY$ wird durch GET T$ ersetzt
-REM Zeilen entfallen um Platz zu sparen.Bei REM Befehl als Sprungziel GOTO .. ändern
-Ausgabeprogramm zwischen Zeile 1090 und 1280 stark verkürzen auf Zeile 1220
-Ersetzen von THEN GOTO durch GOTO in Zeile 2020 und 2105

3.Ergänzung:
Nicht erprobt wurde die Möglichkeit, in den PEEK-Befehlen A durch A + einem Versatz V zu ersetzen .Dadurch sollte es möglich sein, verschiedene Speicheradressen im AIM-65 und in einem 6809-System zu verwenden.

```
10 REM   ---- 6809 -
15 REM  DISASSEMBLER
20 REM   IN BASIC
30 REM TAPE-CODE:BDISN
40 REM REV.22.01.1984
50 REM
490 DIM OC$(15)
500 OC$(0)="2NEG 0????0????2COM 2LSR 0????2ROR 2ASR "
502 OC$(0)=OC$(0)+"2ASL 2ROL 2DEC 0????2INC 2TST 2JMP 2CLR
510 OC$(1)="0????0????0NOP 0SYNC0????0????5LBRA5LBSR"
512 OC$(1)=OC$(1)+"0????0DAA 0ORCC0????3ANDC0SEX 3EXG 3TFR
520 OC$(2)="4BRA 4BRN 4BHI 4BLS 4BHS 4BLO 4BNE 4BEQ "
522 OC$(2)=OC$(2)+"4DVC 4BVS 4BPL 4BMI 4BGE 4BLT 4BGT 4BLE
530 OC$(3)="7LEAX7LEAY7LEAS7LEAU3PSHS3PULS3PSHU3PULU"
532 OC$(3)=OC$(3)+"0????0RTS 0ABX 0RTI 3CWAI0MUL 0????0SWI
540 OC$(4)="0NEGA0????0????0COMA0LSRA0????0RORA0ASRA"
542 OC$(4)=OC$(4)+"0ASLA0ROLA0DECA0????0INCA0TSTA0????0CLRA
550 OC$(5)="0NEGB0????0????0COMB0LSRB0????0RORB0ASRB"
552 OC$(5)=OC$(5)+"0ASLB0ROLB0DECB0????0INCB0TSTB0????0CLRB
560 OC$(6)="7NEG 0????0????7COM 7LSR 0????7ROR 7ASR "
562 OC$(6)=OC$(6)+"7ASL 7ROL 7DEC 0????7INC 7TST 7JMP 7CLR
570 OC$(7)="6NEG 0????0????6COM 6LSR 0????6ROR 6ASR "
572 OC$(7)=OC$(7)+"6ASL 5ROL 6DEC 0????6INC 6TST 6JMP 6CLR
580 OC$(8)="3SUBA3CMPA3SBCA8SUBD3ANDA3BITA3LDA 0????"
582 OC$(8)=OC$(8)+"3EORA3ADCA3ORA 3ADDA8CMPX4BSR 8LDX 0????
590 OC$(9)="2SUBA2CMPA2SBCA2SUBD2ANDA2BITA2LDA 2STA "
592 OC$(9)=OC$(9)+"2EORA2ADCA2ORA 2ADDA2CMPX2JSR 2LDX 2STX
600 OC$(10)="7SUBA7CMPA7SBCA7SUBD7ANDA7BITA7LDA 7STA 7EORA"
602 OC$(10)=OC$(10)+"7ADCA7ORA 7ADDA7CMPX7JSR 7LDX 7STX "
```

```
 610 OC$(11)="6SUBA6CMPA6SBCA6SUBD6ANDA6BITA6LDA 6STA 6EORA"
 612 OC$(11)=OC$(11)+"6ADCA6ORA 6ADDA6CMPX6JSR 6LDX 6STX "
 620 OC$(12)="3SUBB3CMPB3SBCB8ADDD3ANDB3BITB3LDB O????3EORB"
 622 OC$(12)=OC$(12)+"3ADCB3ORB 3ADDB8LDD O????8LDU O????"
 630 OC$(13)="2SUBB2CMPB2SBCB2ADDD2ANDB2BITB2LDB 2STB 2EORB"
 632 OC$(13)=OC$(13)+"2ADCB2ORB 2ADDB2LDD 2STD 2LDU 2STU "
 640 OC$(14)="7SUBB7CMPB7SBCB7ADDD7ANDB7BITB7LDB 7STB 7EORB"
 642 OC$(14)=OC$(14)+"7ADCB7ORB 7ADDB7LDD 7STD 7LDU 7STU "
 650 OC$(15)="6SUBB6CMPB6SBCB6ADDD6ANDB6BITB6LDB 6STB 6EORB"
 652 OC$(15)=OC$(15)+"6ADCB6ORB 6ADDB6LDD 6STD 6LDU 6STU "
 670 OZ$="3FOSWI2838CMPD8C8CMPY8E8LDY 932CMPD9C2CMPY9E2LDY "
 672 OZ$=OZ$+"9F2STY A37CMPDAC7CMPYAE7LDY AF7STY B36CMPD"
 674 OZ$=OZ$+"BC6CMPYBE6LDY BF6LDY CE8LDS DE2LDS DF2STS "
 676 OZ$=OZ$+"EE7LDS EF7STS FE6LDS FF6STS "
 690 OE$="3FOSWI3838CMPU8C8CMPS932CMPU9C2CMPSA37CMPUAC7CMPS"
 692 OE$=OE$+"B36CMPUBC6CMPS"
 700 H$="0123456789ABCDEF"
 710 INPUT"ANZAHL ZEILEN";ZZ
 760 GOSUB3320
 770 IF AI<>0THEN A=AI
 790 GOSUB3140
 800 C=0:GOSUB3700:0X$=C$
 820 M$=MID$(OC$(HN),LN*5+1,1)
 830 O$=MID$(OC$(HN),LN*5+2,4)
 840 OY$="  ":01$="  ":02$="  ":PB$="  ":SA$="  "
 860 IFO<16ORO>17THEN GOTO970
 880 OE=PEEK(A)
 890 A=A+1
 900 C=OE:GOSUB3700:OY$=C$
 910 IF O=16THEN GOSUB2430
 930 IF O=16AND OE>32AND OE<48 THEN GOSUB2540
 950 IF O=17THEN GOSUB2610
 970 GOSUB1050
 990 GOSUB1220
1000 ZE=ZE+1:IF ZE>=ZZ THEN END
1020 GET T$:IFT$<>""GOTO760
1030 GOTO790
1050 IF M$=""THEN M$="0"
1060 N=ASC(M$)-47
1062 ON N GOSUB1300,1070,1330,1400,1510,1590,1760,1840,2320
1070 RETURN
1220 PRINTA$+" "+0X$+0Y$+" "+PB$+" "+01$+02$+" "+O$+" "+SA$
1280 RETURN
1300 RETURN
1330 C=PEEK(A):GOSUB3700:01$=C$
1340 A=A+1
1350 SA$="$"+01$
1360 RETURN
1400 01=PEEK(A)
1410 C=01:GOSUB3700:01$=C$
1420 A=A+1
1430 SA$="##"+01$
1450 ZW$=MID$(O$,1,3)
1452 IF ZW$="TFR"ORZW$="EXG"THEN GOSUB2700
1460 IF ZW$="PSH"OR ZW$="PUL"THEN GOSUB 2940
1470 RETURN
1510 RA=PEEK(A)
1520 C=RA:GOSUB3700:01$=C$
1530 IF RA>127THEN RA=RA-256
```

```
1540 GOTO1660
1590 O1=PEEK(A)
1600 C=O1:GOSUB3700:O1$=C$
1610 A=A+1
1620 O2=PEEK(A)
1630 C=O2:GOSUB3700:O2$=C$
1640 RA=O1*256+O2
1650 IF RA>32767THEN RA=RA-65536
1660 A=A+1
1670 EA=A+RA
1680 EH=INT(EA/256)
1690 EL=EA-EH*256
1700 C=EL:GOSUB3700:EL$=C$
1720 C=EH:GOSUB3700
1722 SA$="$"+O1$+O2$+"=$"+C$+EL$
1730 RETURN
1760 C=PEEK(A):GOSUB3700:O1$=C$
1770 A=A+1
1780 C=PEEK(A):GOSUB3700:O2$=C$
1790 A=A+1
1800 SA$="$"+O1$+O2$
1810 RETURN
1840 IX=PEEK(A)
1850 C=IX:GOSUB3700:PB$=C$
1860 A=A+1
1890 F5=INT(IX/128):IX=IX-F5*128
1900 RR=INT(IX/32):IX=IX-RR*32
1910 FI=INT(IX/16):IX=IX-FI*16
1940 SA$=",X"
1950 IF RR=1THENSA$=",Y"
1960 IF RR=2THEN SA$=",U"
1970 IF RR=3THENSA$=",S"
2010 IF F5=1GOTO2100
2020 IF FI=1GOTO2070
2040 C=IX:GOSUB3700:SA$="$"+C$+SA$
2050 GOTO2280
2070 C=ABS(IX-16):GOSUB3700:SA$="-$"+C$+SA$
2080 GOTO2280
2100 IF IX=12 OR IX=13THEN SA$=",PCR"
2105 IF IX=4GOTO2270
2120 IF IX=0THENSA$=SA$+"+":GOTO2280
2130 IFIX=1THENSA$=SA$+"++":GOTO2270
2140 IF IX=2THENSA$=",-"+RIGHT$(SA$,1):GOTO2280
2150 IF IX=3THENSA$=",--"+RIGHT$(SA$,1):GOTO2270
2170 IFIX=5THEN SA$="B"+SA$:GOTO2270
2180 IF IX=6THEN SA$="A"+SA$:GOTO2270
2190 IF IX=11THEN SA$="D"+SA$:GOTO2270
2200 C=PEEK(A):GOSUB3700:O1$=C$:A=A+1
2220 IF IX=8 OR IX=12GOTO2250
2230 C=PEEK(A):O2$=C$:A=A+1
2250 SA$="$"+O1$+O2$+SA$
2270 IF FI=1THEN SA$="("+SA$+")"
2280 RETURN
2320 C=PEEK(A):GOSUB3700:O1$=C$
2330 A=A+1
2340 C=PEEK(A):GOSUB3700:O2$=C$
2350 A=A+1
2380 SA$="#$"+O1$+O2$
2390 RETURN
```

```
2430 FOR I=1TO161STEP7
2440 IF OY$=MID$(OZ$,I,2)GOTO2470
2450 NEXT I
2460 RETURN
2470 M$=MID$(OZ$,I+2,1)
2480 O$=MID$(OZ$,I+3,4)
2490 RETURN
2540 LN=OE-INT(OE/16)*16
2550 M$="5"
2560 O$="L"+MID$(OC$(2),LN*5+2,3)
2570 RETURN
2610 FOR I=1TO70STEP7
2620 IF OY$=MID$(OE$,I,2)GOTO2650
2630 NEXT I
2640 RETURN
2650 M$=MID$(OE$,I+2,1)
2660 O$=MID$(OE$,I+3,4)
2670 RETURN
2700 SA$=""
2720 RR=INT(O1/16)
2730 GOSUB2790
2750 RR=O1-RR*16
2770 SA$=SA$+","
2790 IF RR=0THEN SA$=SA$+"D":RETURN
2800 IF RR=1THEN SA$=SA$+"X":RETURN
2810 IF RR=2THEN SA$=SA$+"Y":RETURN
2820 IF RR=3THEN SA$=SA$+"U":RETURN
2830 IF RR=4THEN SA$=SA$+"S":RETURN
2840 IF RR=5THEN SA$=SA$+"PC":RETURN
2850 IF RR=8THEN SA$=SA$+"A":RETURN
2860 IF RR=9THEN SA$=SA$+"B":RETURN
2870 IF RR=10THEN SA$=SA$+"CC":RETURN
2880 IF RR=11THEN SA$=SA$+"DP":RETURN
2900 SA$=SA$+"****"
2910 RETURN
2940 SA$=""
2990 RR=INT(O1/128):O1=O1-RR*128:IF RR=1THEN SA$=SA$+"PC,"
3000 RR=INT(O1/64):O1=O1-RR*64
3010 IF RR=1AND MID$(O$,4,1)="U"THENSA$=SA$+"S,"
3020 IF RR=1AND MID$(O$,4,1)="S"THEN SA$=SA$+"U,"
3030 RR=INT(O1/32):O1=O1-RR*32:IF RR=1THENSA$=SA$+"Y,"
3040 RR=INT(O1/16):O1=O1-RR*16:IF RR=1THEN SA$=SA$+"X,"
3050 RR=INT(O1/8):O1=O1-RR*8:IF RR=1THEN SA$=SA$+"DP,"
3060 RR=INT(O1/4):O1=O1-RR*4:IF RR=1THEN SA$=SA$+"B,"
3070 RR=INT(O1/2):O1=O1-RR*2:IF RR=1THEN SA$=SA$+"A,"
3080 IF O1=1THEN SA$=SA$+"CC,"
3100 SA$=MID$(SA$,1,LEN(SA$)-1)+" "
3110 RETURN
3140 AH=INT(A/256)
3150 AL=A-AH*256
3170 O=PEEK(A)
3190 HN=INT(O/16)
3200 LN=O-HN*16
3220 A=A+1
3250 C=AH:GOSUB3700:AH$=C$
3270 C=AL:GOSUB3700:AL$=C$
3290 A$=AH$+AL$
3300 RETURN
```

```
3320 PRINT CHR$(13);
3330 PRINT "STARTADRESSE ";
3340 AI=0
3360 GET I$:IF I$=""THEN3360
3370 IF ASC(I$)=13THEN3560
3380 IF ASC(I$)=32THEN3600
3400 PRINT I$;
3420 I=ASC(I$)-48
3440 IF I>9THEN I=I-7
3460 IF I<0THEN3530
3470 IF I>15THEN3530
3500 AI=AI*16+I
3510 GOTO3360
3530 PRINT "?";
3540 GOTO3360
3560 AI=AI-(INT(AI/65536)*65536)
3570 PRINT CHR$(13)
3590 RETURN
3600 END
3700 CH=INT(C/16)
3710 CL=C-CH*16
3720 C$=MID$(H$,CH+1,1)+MID$(H$,CL+1,1)
3730 RETURN
```

Eine vom Programm erzeugte Liste

```
RUN
ANZAHL ZEILEN? 7

STARTADRESSE 9500

9500 10CE     1390 LDS  #$1390
9504 BD       7DDC JSR  $7DDC
9507 86       13   LDA  #$13
9509 1F       8B   TFR  A,DP
950B 4F            CLRA
950C 5F            CLRB
950D 97       07   STA  $07
```

(Anm. d. Hrsg.: Um die Durchlaufgeschwindigkeit unter BASIC zu optimieren, kann man die initialisierenden Zeilen 490-710 an den Schluß des Programms verlegen, damit sie nicht immer wieder überstrichen werden müssen.)

Roland Löhr

Relative Dateien

Übersicht

Die Massenspeicherung von Informationen auf dem Mikrocomputer ist ein aktuelles Thema. Allenthalben finden wir Angebote für Datenbanksysteme und Vorschläge für solche. In diesem Aufsatz soll eine Übersicht gegeben werden, wie man mit den Floppies von Commodore relative Dateien einrichtet und verwaltet. Man hat damit einen schnellen wahlfreien Zugriff auf beliebige Datensätze und kann sich eine Datenbank oder ein Auskunftssystem nach den eigenen Wünschen schaffen. Zu den relativen Dateien scheint bisher auch wenig veröffentlicht worden zu sein.

Speicherungsformen: Bei der externen Speicherung auf magnetischen und sonstigen Medien kennen wir zwei Haupttypen von Information, nämlich Programme und Daten. Programme sind dabei Informationsströme von Anweisungen, entweder in Maschinensprache oder in den Formulierungen

und tokens höherer Sprachen. Unter Daten hat man jedwede andere Information zu sehen, sei es nun kaufmännische Information, Prozeßinformation, Text oder was immer. Daten werden mit Programmen erfaßt, ausgewählt, verarbeitet und in bearbeiteter Form gespeichert und wieder ausgegeben.

Daten- und Programmträger waren früher meist Magnetbänder. Heute sind es Disketten und harte Magnetplatten. Magnetbänder haben den Nachteil, daß ein Suchvorgang sich im ungünstigsten Fall auf die ganze Datei erstrecken muß, was zeitaufwendig ist. Wenn eine solche Datei auf Magnetband nicht innerlich formatiert/geblockt ist, dann muß sie nach Veränderungen in eine Tochterdatei kopiert werden, was weiteren Aufwand bedeutet.

Auch bei Floppy Disks kennen wir den Begriff der sequentiellen Dateien. Das sind solche, bei denen neue Information an den Schluß einer offenen Datei hinzugeschrieben wird. Eine solche Datei kann unter BASIC 4 und entsprechendem DOS mit dem Befehl APPEND ('anhängen') zur weiteren Datenaufnahme wieder eröffnet werden. Wenn zusätzliche Daten eine eigene Datei bilden, dann kann mit dem Befehl CONCAT (Verschmelzung) eine Verbindung zweier Dateien bewirkt werden. Die Bearbeitungsmöglichkeiten für eine sequentielle Datei bleiben trotz dieser Befehle gleich mit denjenigen für eine sequentielle Datei auf Magnetband. Sollen Daten gesucht werden, so muß im schlimmsten Fall die gesamte Datei durchgelesen werden. Und beim Verändern muß man ebenso eine Tochterdatei mit dem 'update' schreiben. - Ihrem Wesen nach sind sequentielle Dateien damit vor allem für die laufende Datenerfassung geeignet, sei es für die laufende Belegeingabe, für die Erfassung eines Journals von Bearbeitungen oder für die Meßwerterfassung.

Schlüsseldateien

Eine andere Form der Datenhaltung besteht in indizierten Dateien. Dabei dürfen die eigentlichen Nutzdaten in einer unsortierten zufälligen Folge anfallen und eingeschrieben werden. Zum schnellen Wiederauffinden von Daten ist nur wesentlich, daß zusätzliche Information (Schlüssel, Index) für das oder für die Sortiermerkmal(e) angelegt wird. Bei Plattendateien sollte diese Schlüsselinformation auf Spur und Sektor hinweisen, womit der Schreib-/Lesekopf physisch auf die entsprechende Information positioniert werden kann. Für die Organisation solcher Schlüssel gibt es die verschiedensten Methoden. Sie können als eine eigene Datei gehalten werden, sie können hierarchisch gestuft sein. So mag der oberste Index für Namen z.B. die Anfangsbuchstaben aufgliedern, weitere Schlüssel mögen die Folgebuchstaben verfeinern. Solche Verfeinerungsschlüssel mögen jedoch auch neben der Nutzinformation in Datensätzen stehen. Sie sind dort für das Sortiermerkmal Verweise auf den nachfolgenden und ggfs. auf den vorausgehenden Datensatz. - Wenn es mehrere Sortiermerkmale in einer Datei gibt, so sind mehrere Schlüssel zu führen. Zusammenfassend bezeichnen wir so organisierte Dateien als ISAM-Dateien (Index Sequential Access Method), siehe auch Heft 23. ISAM wird als maschinensprachliches Programm angeboten. Wie man Verkettungsinformation mit den Verweisen auf vorangehende und nachfolgende Datensätze unter BASIC für eine Datenbank herstellt und benutzt, wurde in Heft 36 beschrieben. Die dort beschriebenen Prinzipien lassen sich für den Hausgebrauch auch beim folgenden Dateityp anwenden.

Relative Dateien

Dateien dieses Typs stellen eigentlich nichts anderes dar, als eine Zuordnung einer laufenden Nummer für den Dateisatz zu physischen Bereichen der Diskette. Bereich meint: Spur, Sektor und Nummer des Bytes im Sektor, das das erste Nutzbyte des gemeinten Datensatzes enthält. Relative Dateien sind damit zunächst nicht der organisatorische Überbau für eine Dateiverwaltung, sondern die physische Voraussetzung für einen solchen. - Bei den Commodore-Floppies ab 4040, die diese Dateiform erlauben, ist der Benutzer/Programmierer von jeder Überlegung befreit, wohin seine Information gespeichert wird und wo sie wiederauffindbar ist. Das in die Floppy gelegte DOS nimmt eine selbständige Verwaltung der Dateien vor, so daß wir uns hier vor allem mit der Hantierung solcher Dateien beschäftigen können, ohne den Leser mit zu vielen Einzelheiten belasten zu müssen.

Nach mehr als vier Jahren Einsatz relativer Dateien beim Autor für die Abonnementsverwaltung, gefahren auf den Floppies 4040, 8250 und 1001 läßt sich sagen, daß dieser Filetyp übersichtlich und sicher zu handhaben ist. In dieser Zeit sind zehntausende von Buchungen, Veränderungen und

Ausdrucke aus solchen Dateien vorgenommen worden. Durch die bei einem Gewitter aufgetretenen Spannungsschwankungen sind offensichtlich nur einmal geringe Verwerfungen um eine Zeile (ein logisches Feld) an wenigen Datensätzen eingetreten, die später mit Sicherheitskopien der betreffenden Datei bereinigt werden konnten. Als besonders angenehm wird empfunden, daß angesprochene Datensätze in weniger als einer Sekunde auf dem Bildschirm stehen und daß sie nach ihrer Veränderung genau so schnell zurückgeschrieben sind.

Speicherungsform relativer Dateien

Bei sequentiellen und relativen Dateien haben wir je Sektor 256 Bytes insgesamt. Da die ersten beiden Bytes jedoch als Verwaltungsinformation für das DOS benutzt werden (Pointer), haben wir effektiv 254 Nutzbytes je Sektor.

Datensätze in relativen Dateien dürfen für eine feste Länge von 1 bis maximal 254 Bytes definiert werden, mit einem Statement wie z.B.

```
10 DOPEN#1,"NAME",L80
```

Eröffnung der logischen Datei Nr. 1 unter 'NAME" mit der Länge von 80 Bytes je Datensatz. Der Parameter 'Lxx' eröffnet automatisch eine relative Datei. Weitere Parameter wie z.B. ,R oder ,W (bei sequentiellen Files für READ oder WRITE) sind nicht erforderlich. Ein relatives File ist immer automatisch ein Schreib-/Lesefile, was natürlich in der Programmierung zur Vorsorge gegen unbeabsichtigtes Beschreiben mahnt.

Datensätze, 'Records' werden also mit einer festen Länge definiert. Es ist eine maximale Länge, die nicht unbedingt ausgenutzt werden muß. Sie mag Reserven für künftige Datenfelder enthalten. Datensätze werden auf der Floppy in dichter Folge aneinandergereiht. Außer bei den Längen 1, 127 und 254 (die Teiler von 254 sind) wird es immer Datensätze geben, die in einem Sektor beginnen und die sich in einem anderen Sektor fortsetzen. In obigem Beispiel ist z.B. der 4. Datensatz solchermaßen sektorüberspannend: Der Pointer liegt in den Bytes 0 und 1, darauf folgen in 2-241 die ersten drei Sätze. Der vierte Satz beginnt im Byte 242 und setzt sich in einem anderen Sektor fort.

Für die Verwaltung einer relativen Datei unterhält das DOS Zusatzinformationen. Diese wird in den sog. 'side sectors' abgelegt. Dieser Begriff ist nicht besonders hilfreich, wir sollten ihn mit 'hierarchisch erweiterte Inhaltsverzeichnisse' übersetzen, die in vom DOS zugewiesenen Sektoren der Diskette nach Bedarf angelegt werden. Die Hierarchie der Inhaltsverzeichnisse baut sich damit wie folgt auf:

DIRECTORY (Hauptinhaltsverzeichnis der Diskette)
- Byte Nr. 2 enthält hex 84 zur Kennzeichnung 'relative Datei'
- Bytes 21/22 enthalten Track und Sector des ersten 'Inhaltsverzeichnis Blockes' (Erweiterung des Inhaltsverzeichnisses)
- Byte 23 enthält die mit 'Lxx' definierte maximale Länge eines Datensatzes.

INHALTSVERZEICHNIS-BLOCK (Erweiterung des Inhalsverzeichnisses)
- Bytes 0,1 enthalten Spur und Sektor des nächsten Inhaltsverzeichnis-Blockes. (bei den Floppies 4040 und 8050 gibt es keine weiteren Inhaltsverzeichnis-Blöcke, was zu einer Kapazitätsbeschränkung führt.)
- Byte 2 Zahl der Inhaltsverzeichnis-Blöcke
- Byte 3 Satzlänge (s.o. 'Lxx')
- Bytes 4/5, 6/7, 8/9, 10/11, 12/13, 14/15: Spur-/Sektorzeiger auf 6 Seitensektoren. Die Bytes 4/5 zeigen dabei auf Byte 16 in diesem Block, wo der erste Seitensektor liegt.
- Bytes 16-255 Hier liegen 120 Spur-/Sektorzeiger auf Blöcke zu 256 Bytes, in denen die Daten der relativen Datei sich fortsetzen (Verkettungsinformation).

DATENBLOCK DER RELATIVEN DATEI
- Bytes 0/1 Verkettungsinformation: Spur und Sektor, wo die Datei fortgesetzt wird. Wenn dieser Eintrag nicht mit demjenigen im übergeordneten

Inhaltsverzeichnis-Block übereinstimmt, so liegt das Ende der Datei in diesem Sektor. Das nächst freie Byte ist z.B. mit 00 F3 gekennzeichnet.

Bytes 2-255 Bytes der eigentlichen Datensätze. Bei Initialisieren der Datei ist das erste eines Datensatzes jeweils mit hex FF beschrieben, die folgenden sind ausgenullt. Diese Bytes werden später durch die Nutzinformation überschrieben.

Der vorstehende Aufbau der Verwaltungsinformation für das DOS läßt sich wie folgt zusammenfassen: Die relative Datei ist im Hauptinhaltsverzeichnis neben ihrem Namen mit dem Hexbyte 84 als 'relativ' gekennzeichnet. Dort findet sich auch, in welcher Spur/Sektor die Erweiterung des Directory als Inhaltsverzeichnis-Block steht. Jeder Inhaltsverzeichnis-Block kann auf 6 weitere Blöcke hinweisen, die die eigentlichen 'side sectors' sind. Jeder side sector kann auf 120 Spuren/Sektoren weisen, in denen die Nutzdaten der relativen Dateien liegen. Ein Inhaltsverzeichnis-Block weist damit direkt und indirekt maximal auf 6x120=720 Sektoren (=Blöcke) hin. Wenn wir einmal annehmen, daß die größtmögliche Länge eines relativen Datensatzes von 254 Nutzbyte ausgenutzt werden soll, dann steht ein Inhaltsverzeichnis-Block für maximal ca. 182.000 Bytes.

Floppies vom Typ 4040, 8050, 1541 und 2031 können nur 1 Inhaltsverzeichnis-Block anlegen. 182.000 Bytes sind dort also die maximale Dateigröße. Floppies vom Typ 8250 und 1001 (single drive) lassen mehrere dieser Inhaltsverzeichnis-Blöcke zu, so daß eine Datei eine ganze Diskette mit etwa 1 MB überstreichen kann. Bei den Laufwerken D9090 und D9060 können solche Dateien theoretisch sogar 23 MB überspannen.

Jeder Datensatz hat eine Nummer, aus der das DOS Spur und Sektor sowie innerhalb des Sektors das Beginn-Byte berechnet. Datensatz-Nummern dürfen in Bereich von 1-65535 liegen.

Dieser Aufbau der Zeiger ermöglicht es dem DOS, mit wenigen Zugriffen und Berechnungen auf die eigentlichen Datensätze zu positionieren. Das alles geschieht unsichtbar für den Benutzer, der sich nur um die Programmierlogik kümmern muß.

Anlegen einer relativen Datei

Weiter oben wurde schon die Befehlsform genannt, mit der einer relative Datei erstmalig angelegt wird:

```
10 DOPEN#1, "KONTEN",Lxxx
```

Das logische File 1 erhält den Namen 'Konten'. Der Buchstabe 'L' steht für den Dateityp 'relativ' mit Satzlänge 'xxx' (dezimal), wobei xxx eine Zahl zwischen 1 und 254 sein darf.

Mit der Ausführung dieses Befehles werden bereits 2 Sektoren auf der Diskette angelegt. Der erste enthält den Inhaltsverzeichnis-Block, der zweite wird für soviele Nutzdaten-Sätze initialisiert, wie in einen Sektor passen. Die Datei ist damit für die Aufnahme der ersten Datensätze vorbereitet.

Der RECORD-Befehl zum Schreiben und Lesen

Der Befehl RECORD dient dem Positionieren auf einen Datensatz. Er hat die allgemeine Form:

```
RECORD#/logisches File/, /Satz-Nr./, /Byte-Nr./
```

Die einfachen Schrägstriche sind hier bei nicht Bestandteile des Befehls, sondern sie dienen nur zur Abgrenzung der Begriffe in dieser Niederschrift. — RECORD ist damit der zentrale Befehl für relative Files. Er wird im allgemeinen vor dem schreibenden oder lesenden Zugriff gegeben. Soll z.B. in Datensatz 5 ab 6. Byte geschrieben werden, so gibt man:

```
RECORD#1,5,6 : PRINT#1,"Meier"
```

Sofern man den Parameter 'Byte-Nr.' fortläßt, wird auf das erste Byte des Datensatzes positioniert. Beim Schreiben und Lesen wird der Byte-Zeiger automatisch weitergeführt. Das Lesen erfolgt entweder byte-weise mit GET# oder mit INPUT# nach den bekannten Regeln. Auf Besonderheiten wird noch eingegangen.

Erweitern einer relativen Datei

Es ist dem Benutzer überlassen, die relative Datei sich dynamisch mit der jeweiligen weitergreifen-

den Positionierung unter RECORD entwickeln zu lassen. Oft möchte man für eine solche Datei aber von Anfang an ausreichend Platz reservieren, um nicht Gefahr zu laufen, daß die Floppy irgendwann einmal meldet 'Disk Full', ehe man alle vorgesehenen Datensätze untergebracht hat. Diese Reservierung kann man gleich bei der Anlage der Datei bewirken. Es sollen z.B. 500 Datensätze reserviert werden. Dann sagt man:

RECORD #1,500 : PRINT#1,CHR$(255)

Mit diesem Befehl wird irgendetwas in den Datensatz 500 hineingeschrieben. Das DOS initialisiert nun wegen des PRINT# alle Datensätze bis zur Nummer 500 und vermindert in der BAM dementsprechend die Zahl der auf der Diskette noch freien Blöcke. Je nach dem Umfang der Datei kann diese Initialisierung Minuten dauern. - Bei einer späteren zusätzlichen Erweiterung kann man gleiches z.B. bis zum Datensatz 700 bewirken. Die Datei ist also dynamisch erweiterbar, soweit noch Platz auf der Diskette ist und soweit das DOS nicht andere Beschränkungen auferlegt (s.o.).

Feste oder variable Feldlänge?

Ein Datensatz wird meistens mehrere Unterteilungen haben, wie z.B. Name, Anschrift, Konto-Nr. usw. Diese Unterteilungen nennt man 'Felder'. Nun ist bekannt, daß die Begriffe in den Feldern sehr unterschiedliche Länge haben können. Bei Straßennamen gibt es 'Bandwürmer' wie 'Gerhart-Hauptmann-Platz' und bei Orten z.B. '3201 Mölme Post Hogeneggelsen', um nur einige längere zu nennen. Auch Namen können ungewöhnlich lang sein. Bei der Planung einer relativen Datei steht man damit vor der Frage, ob man ein Datenfeld für solche Begriffe immer ab einer bestimmten Byte-Position beginnen lassen will oder ob man in der Länge beweglich bleiben möchte. Im ersteren Fall wird man für die denkbaren Eventualitäten viel Raum reservieren, der meistens nur zu einem Bruchteil in Anspruch genommen wird. Und trotzdem wird es immer einmal Fälle geben, wo ein Eintrag die Feldlänge sprengt. Dateien mit fester Zuordnung von Feldern sind damit in Gefahr, unnötig Platz auf Disketten zu beanspruchen und trotzdem nicht allen Wechselfällen zu genügen.

Datenfelder mit variabler Länge: Bei relativen Dateien sind wir keineswegs zu einer Einteilung mit starren Feldlängen gezwungen. Normalerweise und durchgehend auch vernünftigerweise wird man seine Informationen als Zeichenketten (Strings) abspeichern, die dann im Datensatz durch CR's (Carriage Return) voneinander abgegrenzt sind. Diese Trenner werden vom Statement INPUT# erkannt, so daß es nur noch auf eine geordneten Abfolge der Datenfelder ankommt, die auch leer sein dürfen (sie enthalten dann nur ein CR). Bei einer solchen dynamischen Belegung eines relativen Datensatzes kommt es dann auf eine Programmiertechnik an, die deutlich die Trennzeichen 'CR' forciert und die auch die Bildschirmausgabe der Information unterstützt. Hierzu nachfolgend Programmierbeispiele. Es wird jeweils aus einem Array N$(I) geschrieben bzw. in ein solches gelesen. Es hat 15 Datenfelder. Ab Zeile 3900 findet man die Datenausgabe, ab 3400 die Dateneingabe. Bei der Ausgabe wurde auf die Aufeinanderfolge von 15 PRINT#-Statements zur Floppy zwecks Erzeugung der CR's verzichtet, weil es länger dauert, als was in Zeile 3900 im Computer geschieht. Dort wird die Variable ZW$ schrittweise aufgebaut (concateniert), bis sie die Information aller 15 Variablen enthält. Erst dann erfolgt ein einmaliges PRINT#.

```
 3400 RECORD#1,(Z)
 3600 FORI=0TO14:INPUT#1,N$(I):IF(N$(I)=""ORN$(I)=".")THENN$(I)="  "
 3700 NEXTI:RETURN
 3800 :
 3900 ZW$="":FORI=0TO14:ZW$=ZW$+N$(I)+CHR$(13):NEXTI
 4000 RECORD#1,(Z)
 4100 PRINT#1,ZW$
 4200 RETURN
READY.
```

Man wird zugeben, daß die Ansprache eines Datensatzes wie im Beispiel denkbar kurz und einfach ist. Die Verwendung unterschiedliche langer Datenfelder hat zudem den Vorteil, daß ein sehr lan-

ges Einzelfeld sich häufig mit wesentlich kürzeren anderen zusammenfindet, so daß die maximale Datensatzlänge nicht gesprengt wird.

Besonderheiten

Beim Schreiben in einen relativen Datensatz wird Information abgeschnitten, die die definierte Datensatzlänge überschreiten würde. Anders beim Lesen: Wenn man den letzten String eines Datensatzes gelesen hat und nun einen weiteren INPUT#-Befehl gibt, so liest man bereits im nachfolgenden Datensatz. Es wird also sequentiell weitergelesen. Das heißt: Ein Datensatz muß nicht auf 254 Byte beschränkt sein, wenn man beim Schreiben den Beginn eines Eintrages jeweils mit RECORD auf jeden 2., 3. usw. Datensatz positioniert. Beim Lesen kann man sich vom Anfang eines Datensatzes dann sequentiell weiterarbeiten.

Zusammenfassung

Relative Files können auf den neueren Floppies von Commodore eingerichtet werden, auch auf der 1541, die im Zusammenhang mit den Volkscomputern und anderen Systemen beliebt und oft preiswert ist. Relative Dateien sind ein hervorragendes Instrument, um die 'Intelligenz' des DOS für eigene Anwendungen/Informationssysteme zu benutzen. Dabei kommt es eigentlich nur darauf an, aus einer Hilfsdatei (Kartei oder File auf der Diskette) von einer eingegebenen Suchinformation auf die Datensatz-Nummer Zugriff nehmen zu können. Das Aufschlagen und Zurückschreiben veränderter Datensätze erfolgt dann in Sekundenschnelle. Mit diesen Dateien ist der wahlweise Zugriff auf Information gegeben.

Bücher

Leventhal, L. A.; Saville, W.: 6502 Assembly Language Subroutines. Bei Osborne/McGraw-Hill, Berkely, CA 1982, 550 S., ISBN0-931988-59-4. Das in Englisch geschriebene Buch hat folgende Hauptkapitel: Programmiermethoden - Wie man neue Befehle und Adressierungsarten schafft - Häufige Programmierfehler - 8 Kapitel mit Programmen - Anhänge. – Leventhal ist Autor und CO-Autor vieler Bücher zur Assembler-Programmierung, 6502, 6809, Z80, 68000. In diesem Buch merkt man, wieviel Übersicht bei diesen Arbeiten gewonnen wurde. Die ersten 155 Seiten bringen bereits eine Fülle von Programmiertechniken, die jeweils in kurzen Befehlsfolgen erklärt werden, z.B. zu arithmetischen und logischen Operationen, Entscheidungsfindung, Bearbeitung von Arrays, Tabellenverarbeitung, Zeichenumsetzung, Multiplikation/Division, Parameterübergabe, Interface-Programmierung (E/A). Der zweite Abschnitt beschreibt neue Befehle und Adressierungsarten, die in Software abgebildet werden. Dieser interessante Teil beschreibt Programmiertechniken, die man auf anderen CPU's manchmal schon im Befehlsatz findet, wie bedingte Sprünge und Unterprogrammaufrufe, bedingtes Return, Pre- und Post Increment und Decrement usw. Etwa 340 Seiten umfaßt der Hauptteil des Buches, indem wir vollständige kommentierte Unterprogramme finden. Sie sind wie folgt gruppiert: Code bzw. Zeichenwandlung - Arraybearbeitung/Indizierung - Rechenroutinen binär und dezimal - Beeinflussung/Abfrage von Bits/Verschiebeoperationen - Zeichenkettenbearbeitung - Arbeiten mit Arrays - Ein-/ und Ausgabe - Interrupte. Insgesamt sind es wohl 60 Programmbeispiele, jeweils mit der Beschreibung des Zweckes, der Anfangs- und Endbedingungen, der benutzten Register, der Ausführungszeit und des benötigten Speicherplatzes. Daran schließen jeweils Anwendungsbeispiele an. Es sind also einerseits die Programmiertechniken und andererseits die vielen thematisch geordneten Beispiele, die den Wert dieses Buches für den Assembler-Programmierer ausmachen. Wenn es auch auf den Standard-Befehlssatz der CPU 6502 beschränkt ist (also nicht die CPU R65C02 und ähnliche berücksichtigt), so ist es wegen seiner nützlichen Vorschläge und Lösungen sehr zu empfehlen. - Es ist darauf hinzuweisen, daß es dort auch ein gleichartiges Buch für die CPU Z80 gibt, das im gleichen Umfang Programmiertechniken und Unterprogramme enthält.

Kleinanzeigen

Commodore Rechner CBM 8296 mit 128 KB RAM, 5 Monate alt, günstig. Ebenso Textsystem PROTEXT für 8032 und 8296 einsetzbar, siehe Heft 37. R. Löhr, Tel. 04102-55 816

MMF 900-Besitzer (Mini Mainframe) zwecks Gedankenaustausch gesucht. Wolfgang Proch, Postfach 44, 7582 Bühlertal.

Roland Löhr

GMA-Text für CBM 720

Einleitung

In den Heften 37 und 38 stellten wir bereits zwei Textverarbeitungsprogramme für CBM vor, nämlich PROTEXT und WORDCRAFT. Beim Autor konnte vor einiger Zeit nun das Textsystem der Gesellschaft für Mikroprozessor-Anwendung (Hamburg) in Betrieb genommen werden. Es ist, wie wir noch sehen werden, ein Spitzenprodukt in der Textbearbeitung, und es kann die vom Hersteller und von den Softwarehäusern etwas vernachlässigte Serie 7xx allein wegen dieses Anwender-Programmes interessanter machen.

Heft 32 stellte bereits den CBM 710 vor. Seine angenehmsten Eigenschaften sind zusammenfassend und wiederholend: Beweglicher und entspiegelter Bildschirm mit deutlich lesbaren Zeichen in einer Matrix von 9x14 Punkten. Ferner eine massive bewegliche Tastatur mit ausgeformten Tastenkappen, 10 besonderen Funktionstasten und einem separaten Tastenfeld für das Rechnen. Die Ausstattung der derzeitigen Modelle des 8296 fällt dagegen, auch unter ergonomischen Gesichtspunkten, deutlich ab. Vom Gerät her sind damit günstige Voraussetzungen für den Einsatz an Arbeitsplätzen gegeben.

Von einem Textsystem müssen wir hierzulande einen deutschen Zeichensatz auf dem Bildschirm und eine deutsche Tastaturbelegung fordern. Über den Druckertreiber müssen ferner deutsche Zeichen ausgegeben werden. Und weil es soviele Drucker und Schnittstellen gibt, muß auch diesen unterschiedlichen Anforderungen entsprochen werden können. Das wird hier selbstverständlich geleistet. Das System muß ferner absturzsicher sein, damit der Benutzer keine Texte (und damit Arbeit) verliert. Und es soll vor allem so anwenderfreundlich sein, daß man es unbedenklich in die Hände von Mitarbeitern geben kann. Man muß also fordern, daß es von Anwendern bedienbar ist, die keinerlei Computerkenntnisse haben. Auch diese Bedingungen sind erfüllt.

Anwenderfreundlichkeit

Beim recht verbreiteten WORDPRO störte den Autor, daß man zu oft in der etwa einhundertseitigen Dokumentation blättern mußte. Hier hat man nun die Hilfe-Funktion mit einem Tastendruck hereingerufen. Sie gestattet es, in etwa 70 Bildschirmseiten mit erklärendem Text zu blättern, wobei hier natürlich eine Menueführung eingebaut ist. Die Bedienung des Textsystems ist ferner in einer 40-seitigen gedruckten Dokumentation beschrieben, in der man in Kurzform die Befehle, Hinweise und Beispiele findet. Nach kurzer Zeit der Benutzung schaut man in diese Hilfen nur noch seltener hinein, weil man bei aller Fülle der Gegebenheiten meistens mit gleichartigen Arbeiten beschäftigt ist, für die man die Befehle dann kennt. Wie beim Erlernen einer Programmiersprache ist Experimentieren an der Tastatur natürlich auch hier zu empfehlen. Nützlich ist es natürlich auch, wenn man wichtige Funktionstasten z.B. gruppenweise für Editier-, Sprung- usw. -befehle farbig oder ähnlich kennzeichnet. Es erleichtert das Zurechtfinden.

Mit dem Cursor springen und rangieren

Eine Reihe von Funktionen erlaubt ein bequemes Aufsetzen des Cursors auf beliebige Schreibstellen: Die vier Cursortasten sind wie üblich belegt für die Bewegung nach rechts, links, oben und unten. Man kann an den Anfang oder an das Ende des gesamten Textes springen oder bildseitenweise (Screens) vorwärts

und rückwärts. Ferner ist ein Springen von Wort zu Wort oder von Absatz zu Absatz vor- und rückwärts möglich, ebenso in die Vor- und in die Folgezeile. Man kann auch auf eine Zeile mit Nummer springen. Wenn der Anwender es nicht anders bestimmt, erfolgen Tabulationen in 10er-Schritten. Er kann das aufheben, indem er eigene horizontale oder auch vertikale Tabulatoren setzt. Vertikale Tabulatoren sind sicher immer dann nützlich, wenn man in einem Text häufiger etwas in bestimmten Zeilen ändern will. Es gibt jedoch mit den Blockzeichen auch andere Möglichkeiten dafür. Tabulatoren können wahlweise mit dem Text zusammen abgespeichert werden. Für das später anzusprechende Zahlenrechnen vom Textsystem aus gibt es ferner die Einrichtung der numerischen Tabulatoren. Texte können im Schnellgang nach oben und nach unten über den Schirm gerollt werden, auch seitlich wenn man eine Schreibbreite von größer 80 Zeichen (bis 240) benutzt. Für das Setzen und Löschen der Tabulatoren und für deren Anzeige können bequem die Funktionstasten benutzt werden.

Editierbefehle

Man möchte Texte komfortabel verändern und bearbeiten können. Erste Voraussetzung ist die vorerwähnte Beweglichkeit des Cursors, der Schreibposition. Zu diesen Voraussetzungen gehört auch die Finde-Funktion, auf die wir noch zurückkommen. Einzelne Zeichen werden mit der Taste INS/DEL auf die übliche Art eingefügt oder getilgt. Einzelne Zeilen können eingezogen (Insert) oder gelöscht werden. Der Einfügemodus erlaubt das Einsetzen beliebig langer Passagen ab Cursorposition. Ganze Worte können mit einem Tastendruck gelöscht werden. Eine andere Funktion erlaubt wahlweise das Tilgen ganzer Sätze, Absätze, des Textrestes ab Cursorposition oder des gesamten Textes. Man kann aber auch mit dem Cursor seitwärts oder herunterfahrend einen überstrichenen Textbereich tilgen, auch mitten in einer Zeile beginnend und endend. Der verbleibende Text wird danach zusammengezogen.

Textpassagen können markiert werden. Danach kann man sie kopieren oder sie an einen anderen Platz umsetzen. Das arbeitet entweder auf die neu eingenommene Cursorposition oder aber man nimmt den markierten Bereich gewissermaßen 'auf den Haken' und bugsiert ihn an seinen neuen Platz. Elegant ist auch die Möglichkeit, einen Spaltenbereich zu markieren und ihn dann 'lavierend' in andere Spalten zu kopieren. In Assemblertexten hat man oft hintereinander den Zeilenbeginn '.BYT $...' usw. Mit dem Lavieren kann man solche Texte mühelos in viele Zeilen vervielfältigen. Eine ähnliche Arbeit ergibt sich z.B. in Preislisten bei gleich beginnender Artikelnummer oder Artikelbezeichnung. Gleiches gilt gilt für Autorenverzeichnisse, wie z.B. 'Bach, Joh. S.:'

Besondere Befehle dienen der Groß-/Kleinschreibung. Für Buchstaben gibt es den Befehl, der 'Caps Lock' entspricht. Daneben kann man ab Cursorstelle auf inverse Schreibung von Worten umstellen, um z.B. Fremdtexte zu überarbeiten. Die Texteingabe erfolgt fließend. Wenn ein Wort nicht mehr ganz in eine Zeile paßt, wird es insgesamt in die neue Zeile gezogen, um nicht späteren Formatierungsanweisungen das Leben schwer zu machen. Bei Editieren kann es jedoch vorkommen, daß ein zusammenhängendes Wort danach in zwei Zeilen steht. Hier kann man nun den Justierbefehl hereinrufen, der die Worte wieder ungetrennt in eine Zeile bringt, und zwar wahlweise für den laufenden Absatz oder für den ganzen Text. Für die spätere Ausgabeformatierung mit halbautomatischer Silbentrennung auf dem Drucker oder Bildschirm kann jedes Wort mit einer 'Sollbruchstelle' für die Trennung versehen werden, die dann ggfs. vom System benutzt wird.

Die Erwähnung der Formatierung führt auch auf die interne Darstellung: Bei der

Ausgabe erzeugt das System am Zeilenende ein 'eingeschossenes' CR (Carriage Return). In der Abspeicherung erscheint demgegenüber nur dann ein CR, wenn die entspr. Taste gedrückt wurde. Das zieht erhebliche Beweglichkeit nach sich. Wir können nämlich in der Formatierung der Ausgabe bestimmen, wie breit der Druckspiegel sein soll. Im Text enthaltene CR's würden da nur stören, eine alte Erfahrung auch aus dem Fotosatz. Das CR hat damit seine Hauptbedeutung als Absatzzeichen. Texte werden im CBM-ASCII-Code gespeichert, wodurch Lesbarkeit durch andere Textsysteme erzielt wird. Im Gegensatz zu der besprochenen Version des PROFILA werden Spaces am ungenutzten Zeilenende nicht mit in die Abspeicherung übernommen, was speziell im Lichtsatz Unbequemlichkeiten hervorzog. (Beim PROFILA für den 720 soll das behoben sein.) - Bemerkenswert ist auch, daß fremde Texte, die im Bildschirmformat (WORDPRO) abgespeichert wurden, in das System unter Codewandlung eingelesen werden können. Es war dem Autor sogar möglich, früher auf dem AIM als PRG-File geschriebene Texte einzulesen. Man kann natürlich jedes SEQ-File hereinziehen. Insgesamt hat man 714 Zeilen je Bank im Zugriff. Zeilen dürfen bis 240 Zeilen breit sein. Beim 'echten' 720 mit seinen 256 KB RAM kann man Texte in drei Bänken vorrätig halten und blitzschnell hin- und herschalten und findet dabei den Cursor an der alten Position.

Finden und Ersetzen

Die Such- oder Finde-Funktion ist beim Editieren eine unentbehrliche Hilfe, um bestimmte Textstellen ansteuern zu können. Wir finden natürlich auch diese. Sie ist einerseits unter einer Escape-Sequenz eingebunden, kann aber als häufig benutzte Dienstleistung (wie auch andere häufige) mit einer Funktionstaste definiert und repetiert werden. Das Gleiche gilt für die Textänderung von Instrings (Zeichenketten innerhalb eines Strings). Für letzteres wird ein Ersatzstring definiert. Die Durchsuchung des Textes erfolgt entweder ab Cursorstellung oder global, wenn es sich um verbundene Dateien handelt.

Das Wörterbuch als Editierhilfe

Auf der Systemdiskette sind derzeit etwa 30.000 Wörter gespeichert. Man kann nun jeden Text, der sich auf Diskette befindet, auf Rechtschreibung oder unbekannte Wörter absuchen lassen. Das System bringt solche Abweichungen in inverser Schrift auf den Bildschirm. Der Anwender kann nun entscheiden, ob es sich a) wirklich um einen Fehler handelt. In diesem Falle berichtigt er ihn. b) Er kann über das Wort hinweggehen, weil es richtig ist, aber eigentlich nur sehr selten gebraucht wird, so daß ein Merken nicht lohnt. c) Er kann drittens sagen, die Vokabel sein neu und für das System merkenswert. Er drückt die entsprechende Taste, und das 'Anwenderlexikon' wird um diese Vokabel erweitert. so kann sich jeder Anwender neben dem reichen Grundlexikon ein solches für seine Fachsprache schaffen, sei er nun Rechtsanwalt, Lebensmittelchemiker oder in einem noch anderen Berufe tätig. Das System schreibt natürlich den berichtigten Text unter dem alten Namen zurück, so daß man den jeweils aktuellen korrigierten Textabzug gespeichert hat. Die Durchsuchung eines Textes erfolgt mit Windeseile. Die Durchsuchung eines Buchtextes von 100 Seiten ist in etwa einer halben Stunde erledigt, so daß man hier eine wirksame Korrekturhilfe hat. - Für Auslandskorrespondenz mag erwähnenswert sein, daß sich ein entsprechendes englisches Lexikon in Aufbau befinden soll.

Tischrechnerfunktionen

Ein nützliches Merkmal ist sicherlich, daß man innerhalb des Textsystem die

vier Grundrechenarten benutzen kann, um z.B. für ein Angebot den Preis zu errechnen. Die Anzeige von Ergebnissen findet man in der Statuszeile, von wo das Endergebnis in den Text übertragen werden kann. Der Hergang der Kalkulation ist in diesem Falle für den Leser des Textes unsichtbar. Es gibt aber auch die Möglichkeit, Rechenzeilen und Tabellen mit Operanden sichtbar in den Text aufzunehmen und die Ergebnisse von Zeilen- und Spaltenrechnungen an entsprechende numerische Tabulatorstellungen kolonnengerecht auswerfen zu lassen. Besondere Kniffs ergeben sich, wenn man Kalkulationsvorgaben in den möglichen Kommentarzeilen versteckt. Im Text einer Preisliste werden dann z.B. nur die Katalogpreise sichtbar, nicht jedoch ihre Herleitung. Bei einer Neuausgabe ist dann eine Überarbeitung bequem möglich, indem man die Operanden in den Kommentarzeilen ansteuert.

Diskettenmodus und Textausgabe

Das Textsystem hat ein komplettes Disketteninterface. Man kann Inhalts-verzeichnisse betrachten und auch als Text in den Computer speichern. Damit kann man sich einerseits ein kommentiertes Verzeichnis seiner Files schaffen, andererseits hat man mit einem solchen File die Möglichkeit des gezielten Ladens aus einem Inhaltsverzeichnis heraus. Namen können wie üblich mit einem Stern abgekürzt werden, Fragezeichen in einem Namen können wie üblich als Stellvertreterzeichen dienen. Natürlich sind auch alle anderen Disketten-funktionen geregelt, die des Abspeicherns (auch separat nur für markierte Textpassagen), Laden mit Einzug ab Cursorposition, Kopieren, Backup usw.

Ein großes Kapitel und oft ein Problem in Textsystemen ist die Ausgabe und ihre Formatierung. Beim GMA-Text haben wir die üblichen großzügigen Möglichkeiten. Die verschiedenen Druckertypen werden über entsprechende Treiberprogramme angepaßt. Hier werden Zeichen ggfs. auf einen anderen Code umgeformt und Formatierungsbefehle z.B. für Unterstreichung, Fettschrift usw. in die benötigten Escape-Sequenzen geformt. Im Auswahlmenue kann ferner mit unterschiedlichen Device-Nummern gearbeitet werden, um z.B. einmal einen Matrixdrucker und alternativ ein Schönschreibwerk zu bedienen. Unabhängig vom Druckertreiber kann der Anwender ggfs. auch eigene Escapesequenzen erzeugen.

Die Formatbefehle betreffen zum einen die Anordnung auf den Papier, die Blattlänge, die Druckzeilen je Blatt, den linken und rechten Rand, Kopfzeilen und Fußzeilen. Einstellbar ist ferner der Pitch (Zeichen je Zoll, z.B. 8, 10, 12 und 15), die Zahl der Zeilen je Zoll. Die nächsten Befehle betreffen Fettschrift, Breitschrift und Sonderschrift, das Unterstreichen von Text und nächstes Zeichen halbhoch oder halbtief (in Formeln). Wir können den Rand Lösen, zentrieren, rechtsbündig und im Blocksatz drucken, der Zeilenabstand kann vorgegeben werden, es kann horizontal und vertikal tabuliert werden, soweit der Drucker das versteht. Es besteht die Möglichkeit, die formatierte Ausgabe auf den Bildschirm zu rufen, wobei die Besonderheiten invers dargestellt werden. Zur Entlastung des Computers kann ein (auch verbundenes) Textfile im Hintergrunddruck (Floppy an Drucker) ausgegeben werden.

Das Textsystem hat die Möglichkeit der halbautomatischen Silbentrennung. Auf volle Automatik wurde verzichtet, weil es dadurch zu Wort- und Begriffsverstümmelungen kommen kann. In einem Upgrade, daß beim Druck dieser Zeilen bereits zur Verfügung stehen dürfte, ist mit Trennvorschlägen am Bildschirm zu rechnen. Damit wird der individuellen Steuerung der Trennung und einer befriedigenden Füllung der Zeilen Rechnung getragen, denn bei der Verwendung formatierender Zeichen im Text kann man im Voraus schwer absehen, wie der Zeilenfall sein wird.

Die Ausgabe kann jederzeit angehalten und fortgesetzt werden. Man kann ferner definiert ab Seite Nr. xxx ausgeben. Andere Textsysteme fangen immer wieder von vorne an, was zeitraubend und lästig ist. Man kann auch z.B. die ersten Seiten auf den Bildschirm ausgeben und dann auf Druck umschalten. Texte können auch auf den (beeinflußbaren) seriellen Kanal ausgegeben werden. Es können mehrere Ausdrucke geordert werden und man kann einmal nur die ungeraden (Vorderseiten) und dann die geraden Seiten ausgeben. Auf den Einzug von Fülldateien kommen wir hernach.

Blocksymbole und Fülldateien, Sonderzeichen

Im Text lassen sich deutlich erkennbare Blocksymbole anbringen, die Stellvertreterzeichen für spätere Texteinschübe sind. Diese sind entweder namenlos und werden in der Reihenfolge ihres Auftretens gefüllt oder aber sie haben ein Label, das aus einem der Buchstaben a...z besteht. In diesem Falle können die Begriffe der ebenfalls gegliederten Einzugsdatei in beiliebiger Reihenfolge aufgerufen werden. Ein Nachnahme kann also z.B. in einem Schemabrief an verschiedenen Stellen wiederholt eingezogen werden. Das Textsystem kann aber auch prüfen, ob der Begriff an einer Stelle z.B. 'd' gewissen Zeichenfolgen entspricht, etwa '78*' für Postleitzahlen im Raum Freiburg 'M?ier' für die Namen 'Meier' und 'Maier'. Solche Blockzeichen lassen sich mit und ohne Verschiebung nachfolgenden Textes definieren, und sie können auch von Hand gefüllt, probeweise nach Füllung angezeigt und wieder entleert werden. Wir sind damit schon in der Nähe von Datenbankfunktionen, wie ja Textsysteme sowieso gelegentlich als Auskunftssysteme benutzt werden. - Es ist damit zu rechnen, daß beim Versand dieses Heftes bereits die Ansprache, Prüfung auf Kriterien und die Editierung in relativen Dateien im Textsystem geregelt sein wird, womit wir dann den echten Übergang zum Datenbanksystem haben.

In gemeinsamen Druck mit der Taste CTRL können einige Sonderzeichen erzeugt werden, nämlich die Wurzel, großes Sigma (Summenzeichen), µ, Y und der Backshlash sowie Klammeraffe. Im Zusammenhang mit ESC und Zifferntasten erzeugt man invers dargestellte Stellvertreterzeichen, die man als Kürzel für die später anzuwendende Ersetzungsfunktion oder als Steuerzeichen für den Lichtsatz benutzen kann.

Zusammenfassung

GMA-Text ist ein in Jahren der Vorarbeit sorgfältig entwickeltes Textsystem, das die hardwaremäßigen Möglichkeiten des CBM 720 geschickt benutzt. Es läßt bereits in der hier vorliegenden Form praktisch keine Wünsche offen. In den nächsten Wochen sind zusätzliche Leistungsmerkmale zu erwarten, die es praktisch in den Datenbankbetrieb erweitern. In seinen Such-, Spring- und Editierfunktionen ist es bequem, es hat deutsche Tastaturbelegung, den deutschen Zeichensatz am Bildschirm und in der anpaßbaren Druckerausgabe. Texte können so zur Kontrolle auf dem Bildschirm dargestellt werden, wie sie nachher in der formatierten Druckerausgabe erscheinen. Die Hilfe-Funktionen in den auf Diskette gespeicherten Screens und in der begleitenden Dokumentation sind derart, daß man das Textsystem in die Hände von Anwendern geben kann, die nichts von Computern verstehen. Die Form der Textabspeicherung verträgt sich weiterhin sehr gut mit den Anforderungen, die an Lichtsatz von der Diskette zu stellen sind. Der Autor meint, daß dieses Textprogramm den höchsten Anforderungen gerecht wird, die man heute stellen kann. Er wird daher nicht weiter auf andere hier vorhandene Textprogramme zurückgreifen. Dieser Text wurde auf die Typenradschreibmaschine Olivetti ET 221 ausgegeben. Er hat etwa 2500 Worte, davon 1000 verschiedene.

Aus der Branche

GTE Microcircuits, Montenstraße 11 in 8000 München 19 stellt auf der ELECTRONICA in München die neue Generation der 65xxx Mikroprozessoren vor. Die CMOS-Typen GS65SC816 und -802 sollen Anfang 1985 in Serie lieferbar Jetzt schon lieferbar sind verschiedene CPU's mit 8 Bit in CMOS sowie die volle Palette von CMOS-Interface-Bausteinen, nämlich -21 PIA, -22 VIA, -32 RIOT und -51 ACIA. Ebenfalls in CMOS sind lieferbar der -151 Telecommunication Microcomputer und der gleich bezeichnete -150, auf die wir noch eingehen werden. GTE hat damit derzeit wohl das umfangreichste Angebot an CMOS-Devices für 65xxx in 1 und 2 MHz. – Für den -816 wird der Herausgeber in den nächsten Monaten einen Assembler schreiben. Wünsche dazu dürfen schon geäußert werden.

Ingenieurbüro Stecker, Köln: In den nächsten Wochen finden wieder Seminare zur Datenfernübertragung statt. Ein Prospekt dafür liegt diesem Heft bei. Man hat dort bereits umfangreiche Erfahrungen, die einen sicheren Betrieb ermöglichen.

Vieweg-Verlag: Im Dezember erscheint das Buch 'Der mächtige Draht' von K. B. Hacker, das moderne kommunikationstechniken und Mailbox-Systeme beschreibt.

MOTOROLA hält am 12./13. Nov. 85 im zeitlichen Zusammenhang mit der ELECTRONICA ein VME-Bus-Seminar in München ab.

VALVO: VMEbus-Seminar am 27./28. Nov. in Hamburg. – S68000-Workshop am 13.-15.Nov. in Hamburg.

DRAM-Controller N2964B von Valvo: Er erzeugt die Refresh-Adressen und RAS und CAS für Speicher mit bis zu 256 Worten. 40-poliges Gehäuse.

Cambridge LISP 68000 auf dem SAGE-Mikrocomputer der Fa. mm-Computer in 8210 Prien, Hallwanger Str. 59.

Editorial

Für die begleitenden Zeilen des Herausgebers bleibt einmal wieder wenig Platz. Daher in Kürze: Verschiedene Grundideen sind in diesem Heft wiederaufgenommen worden und zu einer Verbesserung geführt worden, so der Disassembler UNASS, C-64 am IEEE-Bus oder der Disassembler für MC6809. Es sind dieses Hilfsmittel zur Systementwicklung und -Nutzung. – 68000 und 68008 sollen in den folgenden Heften weiter verfolgt werden, auch der neue Prozessor GS65SC816 mit seinen neuen Befehlen und der Adressierungsmöglichkeit von 16 MB.

Sie als Leser sind höflich gebeten, zu den aktuellen Themen mit weiteren Artikeln beizusteuern. Die Mikroprozessorei ist selbst in unseren CPU-Familien inzwischen so umfangreich geworden, daß einzelne Personen nicht mehr das ganze Spektrum beobachten, bewerten und beschreiben können. Es ist weiterhin die Vielfalt aus den Betätigungsgebieten erwünscht, die dieser kleinen Zeitschrift einen besonderen Rang gegeben hat.

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

Texterfassung und Lichtsatz

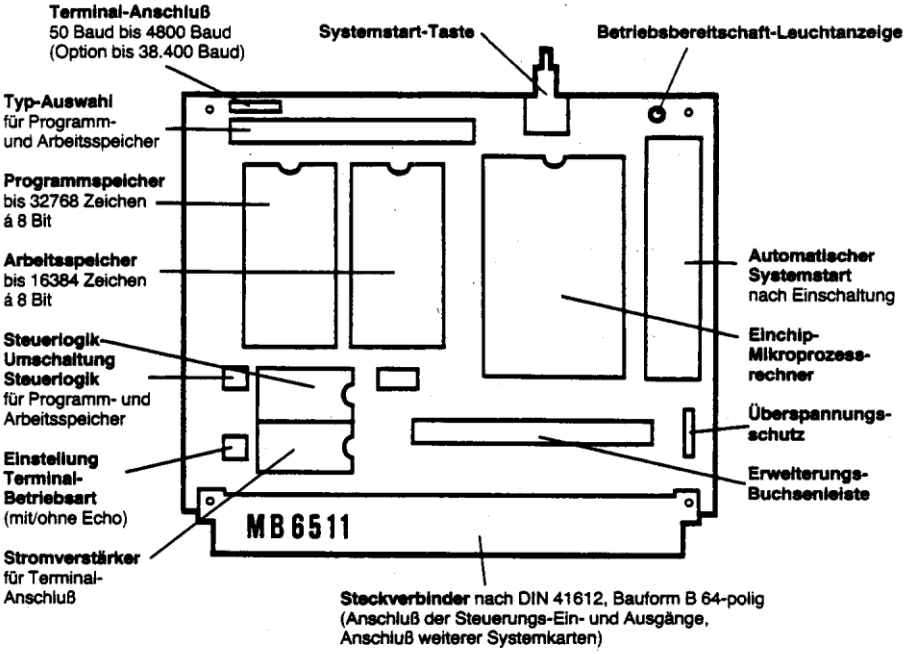
Für die Texterfassung von Büchern, Dokumentationen, Dissertationen etc. mit Fließtext nach klarem Manuskript, auch Sonderzeichen, hervorgehobenen Zwischenüberschriften usw. stehen auftragsweise folgende Möglichkeiten zur Verfügung:

- Sofortige Erfassung und reproreife hier korrigierte Niederschrift mit IBM-Composer (wie in dieser Zeitschrift)
- Texterfassung auf Commodore 8250-Diskette mit späterer Korrekturmöglichkeit, Probeausdruck und anschließendem Lichtsatz oder formatiertem Schreibmaschinensatz
- Regieführung bis zum Druck, auch mit mehrfarbigem Foto auf dem Buchumschlag
- Gestaltung mit Graphikerin, eingabesichere Mitarbeiter

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

8408

MINIBEE- MB 6511



TECHNISCHE DATEN

Leiterplattengröße	100mm x 80mm
Gesamtmaße	100mm x 88mm x 14mm
Kartenmaterial	Glasfaser-Epoxy 1,5mm, beidseitig Lötstoplack
Gewicht	80g
Leistungsbedarf	2W
Spannungsversorgung	5V ± 5%
Systemtakt	2MHz (Option 1MHz)
Arbeitstemperaturbereich	0°C bis +70°C
Programmspeicher	2kByte bis 32kByte BYTEWIDE
Arbeitsspeicher	2kByte bis 16kByte BYTEWIDE
extern erweiterbar bis	4MByte (256 x 16kByte)
Terminal-Anschluß	TTL-kompatibel

RECHNER DATEN

Einchip-Mikroprozessor	R6511Q (Rockwell/NCR)
Rechnerstruktur	6500
Programmierung	softwarekompatibel zu 6500 60 Befehle, 14 Adressierungsarten
Ein-/Ausgabe-Anschlüsse	24 O.C. mit Pullup, 8 Tristate, alle TTL kompatibel
Zähler/Zeitmesser/Takt	1 voll programmierbar, retriggerbar 1 für serielle Schnittstelle
Serielle Schnittstelle	voll duplex, asynchron, synchron
Programm-Unterbrechung (IRQ)	10 (je 2 positiv, negativ, Zähler, Serielle Schnittstelle; 1 unbedingte Unterbrechung (NMI)

EINCHIP-MIKROPROZESSOR-STEUERUNG

mit R6511Q

DM 350,– + Mwst

BRÜHL ELEKTRONIK ENTWICKLUNGS-GESELLSCHAFT mbH

8500 Nürnberg 10, Hegelstr.10 TEL. 0911-359088

Suchen Sie eine preiswerte Alternative zur Dezentralisierung Ihrer Datenverarbeitung?
. . . dann ist 20 mA Current Loop die Lösung für Ihr lokales Punkt zu Punkt Netzwerk!

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

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

DBGM: G 8331 081.9
G 8336 080.8

INGENIEURBÜRO
STECKER

5000 Köln 60 (Niehl)
Postfach 60 07 66
Delmenhorster Str. 20
Tel. (02 21) 7 12 40 18

C64/IEEE-488 Steckmodul

Dieser ausgereifte, weltweit erprobte IEEE-488-Modul eröffnet dem Commodore 64 über seinen parallelen Ausgang ungeahnte Einsatzmöglichkeiten wie:
große, IEEE-kompatible CBM-Peripherie am C-64, simultanen (seriell – VC/paralleler – IEEE) Datenverkehr. Konfliktfreie, speicherverschiebliche Modulsoftware. Im Einsatz beispielsweise **in Schulen** ermöglicht der IEEE-488-Steckmodul problemlose Mehrbenutzersysteme am IEC-Bus wie auch durch die rationell genutzte Peripherie: z. B. zahlreiche Computer an einer Doppelfloppy.
In der Industrie bietet der IEEE-488-Steckmodul die Möglichkeit für preisgünstige IEC-Meß-/Steuersysteme mit dem Commodore 64 als Controller. Zu diesem Modul wird ein **Betriebshandbuch** geliefert, in dem Beschreibungen zu fast sämtlichen Anwendungsfällen mit Programmbeispielen, Belegungstabellen, Angaben zum erforderlichen Kabel- und Steckermaterial, Literatur etc. aufgeführt sind. Zusätzlich können zum IEEE-488-Steckmodul **Anwendungshilfen** wie u. a. Disketten mit Lesekennzeichen, Utility-Disketten usw. bezogen werden.

IEEE-Steckmodul für Commodore 64
einschließlich Betriebshandbuch DM 239,— inkl. MwSt.

te-wi

te-wi Verlag GmbH
Theo-Prosel-Weg 1
8000 München 40

MICRO MAG

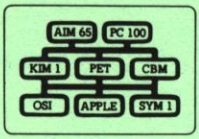
HERAUSGEBER/EDITOR:
DIPL.-VOLKSWIRT ROLAND LÖHR
HANSDORFER STRASSE 4
D-2070 AHRENSBURG
☎ (04102) 5 58 16

Bezugsbedingungen: Abonnement ab laufender Ausgabe für 6 Hefte DM 54,-- im Inland, bzw. DM 59,-- im Ausland (surface mail). Luftpostzustellung auf Anfrage. Abonnements laufen bis auf Widerruf mit Kündigungsmöglichkeit bis zu 4 Wochen vor deren Ablauf. - Nachliefermöglichkeiten siehe unten.

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

Leser-Service des Herausgebers

Das Buch 7-13 des 65.. MICRO MAG



340 Seiten, DM 42,--

Nachliefermöglichkeiten:

Vergriffen sind 'Das Buch 1-6 des 65xx MICRO MAG' sowie die Hefte 1-13 der Zeitschrift. Es erfolgt keine Neuauflage.

Lieferbar: 'Das Buch 7-13 des 65xx MICRO MAG' zu DM 42,-- sowie die Hefte 14-38 zu DM 7,80/St. (ab 10 St. in einer Sendung: DM 6,--/St.).

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.