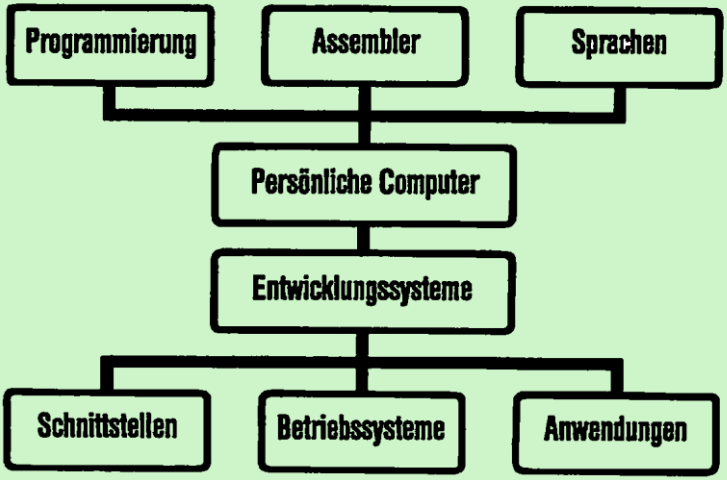
MICRO MAG

DM 9,50 Nr. 42 April 1985



Inhaltsverzeichnis

Screeneditor für AIM 65 mit CPU 65C02 3
Schneller Stringsuchen 23
DOS für AIM 65 30
MC-65 als Speichererweiterung 44
Der Soft Spy für AIM 65 47
Aus der Branche 48
Editorial 49
Bücher 50
Mitgliederverwaltung 51

te-wi aktuell...

„ERFAHRUNGEN MIT GATE ARRAYS“

Dieses Buch (Andy Rappaport) berichtet über das Experiment eines Neuanwenders, seine Schaltung mit etwa 1000 digitalen Gatterfunktionen zum Vergleich mit 3 Herstellern – LSI LOGIC, AMI und IMP – und in 2 Technologien – Gate Arrays und Standardzellen – zu realisieren.

„ERFAHRUNGEN MIT GATE ARRAYS“

beschreibt sämtliche Stufen einer Semicustom-IC-Herstellung: Entwicklung, Übertragung auf Funktionsbibliotheken, Verifizierung durch Simulationen, Produktion und Eingangsprüfung sowie die anwender- und herstellerseitigen Mittel wie Bildschirmarbeitsplätze, transportable Software usw. Eine ausführliche Wirtschaftlichkeitsanalyse liefert zum Abschluß Entscheidungskriterien für Semicustom-ICs.

Etwa 200 Seiten, sehr zahlreiche Abbildungen, Hardcover, DM 66.–

te-wi

te-wi Verlag GmbH
technisch wissenschaftliche Elektronik-Literatur
Theo-Prosel-Weg 1 8000 München 40

tm 4263/1

Friedrich Hofmann, Sperberweg 13, 7922 Herbrechtingen

Screeneditor

für AIM 65 mit CPU 65C02

Einleitung

Der zeilenorientierte Texteditor des AIM 65 arbeitet bei Erstellung von Assemblerprogrammen mit CRT-Modul noch ganz akzeptabel. U.a. bei der Erstellung von Forth-Programmen jedoch ist ein zeichen-orientierter Texteditor sehr vorteilhaft. In diesem Beitrag wird ein Bildschirmeditor vorgestellt, der auch über die TTY-Schnittstelle arbeitet, im Terminalbetrieb also.

Allgemeines

Der Editor ist modular aufgebaut. Das sollte es erlauben, in Kürze einen Makro-Linking-Loader zu integrieren. Auf Tabulatorfunktionen wurde verzichtet, da in Verbindung mit dem Makro-Linking-Loader automatische Tabulatorfunktionen bei fehlendem Label vor dem Befehl und vor dem Kommentar erzeugt werden. Das ist aber noch Zukunft. - Der Editor ist mit dem des AIM voll kompatibel. Somit konnte auf die User-Ein/Ausgabe verichtet werden. Der Textbufferinhalt muß durch Aufruf des AIM-Editors (Taste 'T') über die Listfunktion zum Massenspeicher geleitet werden, ebenso die Eingabe bestehender Texte, so daß man sich hier in gewohnten Hantierungen bewegt. Das Bildschirmformat wurde auf 24 Zeilen und 80 Zeichen/Zeile festgelegt, es kann jedoch durch Ändern der Variablen in der Zeropage beliebig geändert werden. Ebenso können die CTRL-Steuerkommandos für das Videomodul angepaßt werden. Die angegebenen Steuerkommandos sind für das Rockwell CRT-Modul bestimmt. Es wurde darauf geachtet, möglichst wenig Unterprogramme des AIM 65 zu benutzen, damit das Programm leichter für andere Systeme umzuschreiben ist.

Start des Programmes

Ein Kaltspart des Editors wird bei BEGIN vorgenommen. Der Warmstart liegt bei START2, man benutzt ihn bei bereits gefülltem Textbuffer.

Cursor-Kommandos

Cursor Up, CTRL-M bewegt den Cursor eine Zeile nach oben. Die Topzeile des Textbuffers wird nicht überschritten. Beim Erreichen der obersten sichtbaren Zeile des Schirms wird gescrollt, so daß die nächste Zeile sichtbar wird.

Cursor Down, CTRL-N Der Cursor wird eine Zeile nach unten bewegt, die letzte Zeile des Textbuffers wird nicht unterschritten. Nach dem Erreichen der letzten Zeile wird der Cursor an die zuletzt aktive Stelle geführt. Wird die letzte Textbuffer-Zeile nicht erreicht, dafür aber der untere Bildschirmrand, so wird gescrollt und die nächste Zeile angezeigt.

Cursor rechts, CTRL-K
Cursor links, CTRL-J Der Cursor wird sinngemäß bewegt, die Bildschirmränder werden nicht überschrieben.

Cursor ganz nach links, CTRL-W.

List-Kommandos

CTRL-L, Liste die nächste untere Seite auf und setze Cursor auf den Anfang der 12. Zeile.

CTRL-A, Liste die nächste obere Seite auf und setze den Cursor zum Anfang der 12. Zeile.

CTRL-D, Fragt nach einer zweistelligen Zahl, um die der Cursor nach unten geführt wird. Der Befehl listet dann eine Seite (Screen) auf und führt den Cursor in die 12. Zeile.

CTRL-U, Hier gilt das Gleiche wie für CTRL-D, nur wirdderCursor nach oben geführt.

Zeilen-Kommandos

Carriage Return, CR, übernimmt eine eingegebene Zeile oder eine geänderte. Danach sind alle CTRL-Kommandos zulässig.

Trenne Zeile, CTRL-V Im Cursor und bis zum Bildschirmrand stehende Zeichen werden zu einer neuen Zeile generiert und dabei in der alten Zeile gelöscht. Die neue Zeile wird unter dieser im Textbuffer und auf dem Bildschirm eingeschoben. Die Cursorposition bleibt unverändert.

Insert one Line, CTRL-I Erzeugt eine Leerzeile über der Zeile in der der Cursor stand. Der Cursor wird zum Anfang der Leerzeile geführt. Der horizontale Standort des Cursors vor dem Aufruf ist beliebig Die Zeile wird mit einem CR abgeschlossen. Wenn nichts eingegeben wurde, so wird sie durch dieses CR wieder beseitigt.

Multiple Insert one Line, CTRL-R Hier wird nach jedem CR ein Insert one Line aufgerufen. Beendet wird dieser automatische Aufruf durch ein CR in einer leeren Zeile.

Füge Zeichenfolge ein, CTRL-Z Die eingegebenen Zeichen werden links vom Cursor abgespeichert und der Rest der Zeile wird nach rechts verschoben. DEL löscht das zuletzt eingegebene Zeichen links vom Cursor und korrigiert die Zeichenfolge rechts vom Cursor. CTRL-Z wird automatsich aufgerufen, wenn neben dem letzten Zeichen einer abgeschlossenen Zeile einfach weitergeschrieben wird oder wenn der alte Text überschrieben und die aktive Zeichenzahl überschritten wird. CTRL Z kann mit jedem Cursor-Steuerkommando oder CR verlassen werden. Dies vereinfacht Änderungen in einer bestehenden Zeile erheblich.

Delete one Line, CTRL-C Lösche eine Zeile, die Cursorposition bleibt unverändert.

Delete to End of Line, CTRL-E Löscht alle Zeichen die im Cursor und rechts von ihm stehen, die Cursorposition bleibt unverändert.

Sonstige Funktionen

Exit Editor, CTRL-Q Bringt Textende im AIM-Format und führt NOWLN (den Zeiger auf die aktuelle Zeile) zur obersten Zeile und springt anschließend zum System-Monitor. Danach kann in FORTH direkt SOURCE aufgerufen werden.

Top Line, CTRL-T Liste erste Zeile des Textbuffers und führe den Cursor in die oberste Zeile.

Finde Folge, CTRL-F Fragt zuerst nach dem zu suchenden String und sucht dann den String ab aktueller Zeile minus einer. Wird der String gefunden, so wird die Seite, in der er steht, gelistet, der Cursor zeigt auf den gefundenen String. Dieser steht immer in der 12. Zeile. Ist es nicht die gesuchte Zeile mit diesem String, so kann durch Betätigen von SPACE weitergesucht werden. Wenn die von Benutzer gewünschte Zeile gefunden ist, so muß mit CR quittiert werden. Danach sind wieder alle Kommandos zulässig. Wenn das Textende erreicht wurde, so wird das Menue gelistet. Wenn nun CTRL-F wieder betätigt wird, so wird ab der Topzeile gesucht. Ist dies nicht erwünscht, so muß die entsprechende Stelle durch Listanweisungen gegeben werden.

Liste Menue, CTRL-X Listet das Menue auf, ohne die aktuelle Zeilenposition zu verlassen.

Nach jedem Cursor-Kommando oder sonstiger CTRL-Funktion wird der Cursor auf eine sinnvolle und zweckmäßige Position geführt.

Die DEL-Taste, hex 7F Im normalen Eingabemodus hat sie die gewohnte Funktion. Wenn die zuletzt aktive Zeile durch CTRL-Funktionen verlassen wurde, so wird die Funktion der DEL-Taste jedoch geändert. Sie löscht nun das Zeichen, auf das der Cursor zeigt und führt den Rest der Zeile um eine Stelle nach links, d.h. der Cursor ändert seine Position nicht, sondern es erscheint nach jedem Betätigen ein neues Zeichen von rechts her unter dem Cursor.

Zum Schluß

Der neue Texteditor ist ein nützliches Werkzeug. Wegen seiner Länge ist er jedoch umständlich abzutippen. Der Autor ist deshalb bereit, das Programm zum Selbstkostenpreis im EPROM zu liefern, wobei der Adreßbereich anzugeben ist, in dem das EPROM arbeiten soll. Wegen der benutzten Befehle läuft es nur mit dem Befehlssatz des 65C02 von Rockwell, nicht jedoch mit dem 65SC02 von GTE, da letztere einen kleineren Befehlssatz hat.

```
LINE  ADDR  OBJECT          SOURCE

0001  0000                 *=$DF
0002  00DF                 NOWLN  *=*+2          ;ZEILENZEIGER
0003  00E1                 LASTZ  *=*+2          ;ZULETZ AKTIVE ZEILE
0004  00E3                 TEXTA  *=*+2          ;TEXTPUFFERANFANG
0005  00E5                 TEXTE  *=*+2          ;TEXTPUFFERENDE
0006  00E7                 STRING *=*+20
0007  00FB                 *=STRING
0008  00E7                 H1     *=*+1          ;DIV. BERECHNUNGEN
0009  00E8                 H2     *=*+1          ;DIV. BERECHNUNGEN
0010  00E9                 HLAST  *=*+2          ; "    "
0011  00EB                 HNOWLN *=*+2          ;"     "
0012  00ED                 TRANSZ *=*+2          ;"     "
0013  00EF                 *=$00
0014  0000                 Y1     *=*+1
0015  0001                 Y2     *=*+1
0016  0002                 CBYT   *=*+1
0017  0003                 ;BYT 0 CTRL-MODUS
0018  0003                 ;" " 1 ERGEBNIS VON CRTEST
0019  0003                 ;" " 2
0020  0003                 ;" " 3 CTRL-E AUTOMATIK PF2
0021  0003                 ;" " 4 MERKER FUER UPF1 CTRL FUNKTION
0022  0003                 ;" " 5   "     "    "    "       "
0023  0003                 LASTY  *=*+1
0024  0004                 STRLEN *=*+1
0025  0005                 OUTALL =$E9BC
0026  0005                 RBYTE  =$E3FD
0027  0005                 INALL  =$E93C
0028  0005                 MONIT  =$E1A1
0029  0005                 LINE   =24
0030  0005                 LINE2  =12
0031  0005                 ZANZ   =79
0032  0005                 CS     =$03           ;AUSGABEZEICHEN FUER CLEAR SCREEN
0033  0005                 IOLSCR =$15           ;AUSGABEZEICEN FUER INSERT ONE LIN
0034  0005                 CELSCR =$02           ;AUSGABEZEICHEN FUER CLEAR TO END
0035  0005                 CESSCR =$04           ;AUSGABEZEICHEN FUER CLEAR TO END
0036  0005                 CLSCR  =$01           ;AUSGABEZEICHEN FUER CLEAR LINE
0037  0005                 CHSCR  =$0E           ;AUSGABEZEICHEN FUER CURSOR HOME O
0038  0005                 CUE    =$0E           ;EINGABEZEICHEN FUER CURSOR UP
0039  0005                 CUA    =$0B           ;AUSGABEZEICHEN FUER CURSOR UP
0040  0005                 CDE    =$0D           ;EINGABEZEICHEN FUER CURSOR DOWN
0041  0005                 CDA    =$0A           ;AUSGABEZEICHEN FUER CURSOR DOWN
0042  0005                 CLE    =$0A           ;EINGABEZEICHEN FUER CURSOR LEFT
0043  0005                 CLA    =$08           ;AUSGABEZEICHEN FUER CURSOR LEFT
0044  0005                 CRE    =$08           ;EINGABEZEICHEN FUER CURSOR RIGHT
0045  0005                 CRA    =$09           ;AUSGABEZEICHEN FUER CURSOR RIGHT
0046  0005                 FZE    =$1A           ;EINGABEZEICHEN FUER FUEGE ZEICHEN
0047  0005                 IOL    =$09           ;EINGABEZEICHEN FUER INSERT ONE LI
0048  0005                 C      =$03           ;EINGABEZEICHEN FUER KILL LINE
0049  0005                 XXX    =$18           ;EINGABEZEICHEN FUER LISTE MENUE
0050  0005                 BOT    =$02           ;EINGABEZEICHEN GEHE ZUR LAST LINE
0051  0005                 TA     =$14           ;EINGABEZEICHEN GEHE ZUR OBERSTEN
0052  0005                 CEL    =$05           ;EINGABEZEICHEN CLEARE AB CURSOR T
0053  0005                 LL2    =$0C           ;EINGABEZEICHEN LISTE NAECHSTE ZEI
0054  0005                 LA2    =$01           ;EINGABEZEICHEN LISTE NAECHSTE ZEI
0055  0005                 FFF    =$06           ;FINDE FOLGE WEITERSUCHE MIT SPACE
0056  0005                 WEX    =$17           ;NICHT MEHR BELEGT
0057  0005                 UXX    =$15           ;EINGABEZEICHEN GEHE XX ZEILEN AUF
0058  0005                 DXX    =$04           ;EINGABEZEICHEN GEHE XX ZEILEN ABW
0059  0005                 REN    =$12           ;RE-ENTER EDITOR FUNKTION WIE "R"
0060  0005                 QEX    =$11           ;EINGABEZEICHEN FUER EXIT EDITOR
0061  0005                 VVV    =$16           ;EINGABEZEICHEN FUER TRENNE ZEILE
0062  0005                 ;
0063  0005                 *=$6400

0064  6400                 ;
0065  6400                 ;INITIALISIERUNG
0066  6400                 ;
0067  6400                 BEGIN
0068  6400  64 02           STZ CBYT
0069  6402  A9 03           LDA #CS               ;LADE CLEAR SCREEN
0070  6404  20 BC E9        JSR OUTALL
```

```
0071  6407  80 14        BRA IO1
0072  6409  A2 00  EDITOR LDX #00
0073  640B  BD 17 64 EDOUT  LDA TABEDI,X
0074  640E  20 BC E9       JSR OUTALL
0075  6411  E8             INX
0076  6412  E0 06          CPX #IO1-TABEDI
0077  6414  D0 F5          BNE EDOUT
0078  6416  60             RTS
0079  6417  45 44 49 TABEDI .BYT 'EDITOR'
0080  641D  20 09 64 IO1    JSR EDITOR
0081  6420  A9 0D          LDA #$0D
0082  6422  20 BC E9       JSR OUTALL
0083  6425  20 A3 E7       JSR $E7A3       ;OUTPUT FROM AIM ROUTINE
0084  6428  AD 1C A4       LDA $A41C       ;LADE ANFANGADRESSE
0085  642B  85 E3          STA TEXTA
0086  642D  85 DF          STA NOWLN
0087  642F  AD 1D A4       LDA $A41C+1     ;LADE ANFANGADRESSE
0088  6432  85 E4          STA TEXTA+1
0089  6434  85 E0          STA NOWLN+1
0090  6436  20 3B E8 BEGIN2 JSR $E83B       ;OUTPUT TWO SPACES AIM OUTINE
0091  6439  20 A7 E7       JSR $E7A7       ;OUTPUT TO AIM ROUTINE
0092  643C  AD 1C A4       LDA $A41C       ;LADE ENDADRESSE
0093  643F  85 E5          STA TEXTE       ;ABSPEICHERN DES TEXTPUFFER ENDE
0094  6441  AD 1D A4       LDA $A41C+1     ;LADE ENDADRESSE
0095  6444  85 E6          STA TEXTE+1
0096  6446  A9 03          LDA #CS
0097  6448  20 BC E9       JSR OUTALL      ;CLEAR SCREEN
0098  644B                 ;
0099  644B                 ;ENDE DER INITIALISIERUNG
0100  644B                 ;
0101  644B  A0 00          LDY #00
0102  644D  A2 00          LDX #00
0103  644F  20 3C E9 INPUT  JSR INALL       ;HOLE ZEICHEN
0104  6452  20 B2 6A       JSR TESTAD
0105  6455  C9 20          CMP #$20        ;PRUEFE CONTROL FUNKTION
0106  6457  B0 03          BCS NOCTRL      ;NORMALES ZEICHEN DANN SPRUNG
0107  6459  4C 97 64       JMP CTRL        ;EINGABE WAR CONTROL FUNKTION
0108  645C  0F 02 18 NOCTRL BBR 0,CBYT,NOIN
0109  645F  48             PHA
0110  6460  B1 DF          LDA (NOWLN)Y
0111  6462  C9 0D          CMP #$0D
0112  6464  D0 0B          BNE OKOD
0113  6466  68             PLA
0114  6467  C9 7F          CMP #$7F
0115  6469  F0 05          BEQ OKOD1
0116  646B  A7 02          SMB 2,CBYT
0117  646D  4C ED 65       JMP UPF1
0118  6470  48       OKOD1 PHA
0119  6471  20 69 6A OKOD  JSR CRTEST
0120  6474  68             PLA
0121  6475  B0 D8          BCS INPUT
0122  6477  C9 7F    NOIN  CMP #$7F        ;PRUEFE OB DEL
0123  6479  D0 10          BNE NODDEL
0124  647B  BF 02 0A       BBS 0,CBYT,DEL3 ;GEHE ZUR CURSOR DEL.ROUTINE
0125  647E  C0 00          CPY #00         ;PRUEFE OB ZEILENANFANG ERREICHT W
0126  6480  F0 CD          BEQ INPUT

0127  6482  88             DEY             ;WENN NICHT ERREICHT DANN ERNIEDRI
0128  6483  20 BC E9       JSR OUTALL
0129  6486  80 C7          BRA INPUT
0130  6488  4C 85 67 DEL3  JMP DEL
0131  648B  C0 4F    NODDEL CPY #ZANZ       ;PRUEFE OB CURSOR MAX ZEICHEN ZAHL
0132  648D  F0 C0          BEQ INPUT
0133  648F  91 DF          STA (NOWLN),Y   ;SPEICERE ZEICHEN AB
0134  6491  20 BC E9       JSR OUTALL
0135  6494  C8             INY
0136  6495  80 B8          BRA INPUT
0137  6497                 ;
0138  6497  C9 0D    CTRL  CMP #$0D        ;PRUEFE OB 'CR'
0139  6499  D0 20          BNE NO7         ;WENN NICHT SUCHE WEITER NACH CTRL
0140  649B  BF 02 1D       BBS 0,CBYT,NO7  ;WENN PROG IM CTRLMODUS DANN > CTR
0141  649E  20 4A 6A       JSR DECNLN      ;PRUEFE ZULETZT EINGEGEBENES ZEICH
```

```
0142  64A1  B1 DF        LDA (NOWLN),Y  ;HOLE ZULETZT EINGEGEBENES ZEICHEN
0143  64A3  20 43 6A     JSR INCNLN
0144  64A6  C9 0D        CMP #$0D       ;'CR' WAR WENN JA GEHE ZU INPUT
0145  64A8  F0 0E        BEQ N09
0146  64AA  A9 0D        LDA #$0D
0147  64AC  91 DF        STA (NOWLN),Y  ;SPEICHERE 'CR' AB
0148  64AE  C8           INY
0149  64AF  20 BC E9     JSR OUTALL
0150  64B2  20 58 6A     JSR ADDY       ;STELLE NOWLN AUF NAECHSTE ZEILE
0151  64B5  20 08 6B     JSR INXO       ;ERHOEHE X REG. BZW LINE
0152  64B8  4C 4F 64  N09  JMP INPUT
0153  64BB              ;
0154  64BB              ;CTRL VERZWEIGUNG
0155  64BB              ;
0156  64BB  8F 02 05  N07  BBS 0,CBYT,N33
0157  64BE  20 FB 6A     JSR LASTNL
0158  64C1  87 02        SMB 0,CBYT
0159  64C3              ;
0160  64C3  DA           N33  PHX
0161  64C4  A2 FF        LDX #$FF
0162  64C6  E8           TCTRL  INX
0163  64C7  E0 16        CPX #JTABLE-TABLE+1
0164  64C9  D0 04        BNE FINDCT
0165  64CB  FA           PLX
0166  64CC  4C 4F 64     JMP INPUT
0167  64CF  DD 45 6E  FINDCT CMP TABLE,X
0168  64D2  D0 F2        BNE TCTRL
0169  64D4  8A           TXA
0170  64D5  FA           PLX
0171  64D6  0A           ASL A
0172  64D7  18           CLC
0173  64D8  69 5A        ADC #<JTABLE
0174  64DA  85 E7        STA H1
0175  64DC  A9 6E        LDA #>JTABLE
0176  64DE  69 00        ADC #0
0177  64E0  85 E8        STA H1+1
0178  64E2  B2 E7        LDA (H1)
0179  64E4  85 EB        STA HNOWLN
0180  64E6  E6 E7        INC H1
0181  64E8  D0 02        BNE TCTRL1
0182  64EA  E6 E8        INC H1+1
0183  64EC  B2 E7     TCTRL1 LDA (H1)
0184  64EE  85 EC        STA HNOWLN+1
0185  64F0  6C EB 00     JMP (HNOWLN)
0186  64F3              ;
0187  64F3              ;DIESES PROGRAMM GIBT DIE LETZTEN 10 ZEILEN AUS
0188  64F3              ;
0189  64F3  0F 02 37  CH   BBR 0,CBYT,N16 ;PROG.DURCHFUEREN WENN IM CTRL MOD
0190  64F6  A5 E1           LDA LASTZ      ;___
0191  64F8  85 DF           STA NOWLN      ;   ¦
0192  64FA  A5 E2           LDA LASTZ+1    ;   ¦ UBERTRAGE LETZTE AKTIVE ZEIL
0193  64FC  85 E0           STA NOWLN+1    ;___¦
0194  64FE  A9 18           LDA #LINE      ;LADE DIE ZU HOEHERGEHENDE ANZAHL
0195  6500  20 14 6B  N18    JSR PTA        ;PRUEFE TEXTPUFFER
0196  6503  B0 06           BCS N17 .
0197  6505  20 B7 6A        JSR UPNLN      ;GEHE EINE ZEILE HOEHER
0198  6508  3A              DEC A
0199  6509  D0 F5           BNE N18
0200  650B  A9 0E        N17  LDA #CHSCR
0201  650D  20 BC E9        JSR OUTALL
0202  6510  A2 00           LDX #00
0203  6512  A0 00           LDY #00
0204  6514  A9 18           LDA #LINE      ;LADE WERT DER AUSZUGEBENDEN ANZAH
0205  6516  20 21 6B  N21    JSR PLZ        ;PRUEFE OB AKTIVES TEXTPUFFERENDE
0206  6519  B0 06           BCS N24
0207  651B  20 2E 6B        JSR CLLIST     ;GEBE EINE ZEILE AUS
0208  651E  3A              DEC A
0209  651F  D0 F5           BNE N21
0210  6521  C4 03        N24  CPY LASTY
0211  6523  F0 08           BEQ N16
0212  6525  B1 DF           LDA (NOWLN)Y   ;:GEBE REST DER ZEILE AUS
```

```
0213  6527  20 BC E9        JSR OUTALL
0214  652A  C8              INY
0215  652B  80 F4           BRA N24
0216  652D  07 02     N16   RMB 0,CBYT
0217  652F  A9 04           LDA #CESSCR
0218  6531  20 BC E9        JSR OUTALL
0219  6534  4C 4F 64        JMP INPUT
0220  6537                 ;
0221  6537                 ;UP CURSOR LEFT
0222  6537                 ;
0223  6537  C0 00     CL    CPY #00         ;PRUEFE OB CURSOR LINKS STEHT
0224  6539  F0 06           BEQ N35
0225  653B  A9 08           LDA #CLA        ;GEBE CURSOR LEFT AUS
0226  653D  20 BC E9        JSR OUTALL
0227  6540  88              DEY
0228  6541  20 21 6B  N35   JSR PLZ
0229  6544  90 06           BCC CL2
0230  6546  C4 03           CPY LASTY
0231  6548  D0 02           BNE CL2
0232  654A  07 02           RMB 0,CBYT
0233  654C  4C 4F 64  CL2   JMP INPUT
0234  654F                 ;
0235  654F                 ;UP FUER CURSOR RECHTS
0236  654F                 ;
0237  654F  20 21 6B  CR    JSR PLZ
0238  6552  90 09           BCC N34         ;PRUEFE NICHT Y REG WENN NICHT IN
0239  6554  C4 03           CPY LASTY
0240  6556  90 05           BCC N34         ;GEBE RECHTS AUS WENN IN AKTIVER Z
0241  6558  07 02           RMB 0,CBYT
0242  655A  4C 4F 64        JMP INPUT
0243  655D  C0 4F     N34   CPY #ZANZ       ;PRUEFE OB CURSOR NACH RECHTS DARF
0244  655F  B0 11           BCS N36
0245  6561  C8              INY
0246  6562  A9 09           LDA #CRA
0247  6564  20 BC E9        JSR OUTALL
0248  6567  20 21 6B        JSR PLZ
0249  656A  90 06           BCC N36
0250  656C  C4 03           CPY LASTY
0251  656E  90 02           BCC N36
0252  6570  07 02           RMB 0,CBYT

0253  6572  4C 4F 64  N36   JMP INPUT
0254  6575                 ;
0255  6575                 ;GEHE MIT CURSOR EINE ZEILE HOEHER
0256  6575                 ;
0257  6575  20 14 6B  CU    JSR PTA
0258  6578  B0 3C           BCS J01
0259  657A  E0 00           CPX #00         ;PRUEFE OB CURSOR OBEN NOCH PLATZ
0260  657C  D0 2D           BNE J02
0261  657E  A9 15           LDA #IOLSCR
0262  6580  20 BC E9        JSR OUTALL      ;GEBE INSERT ONE LINE AUS
0263  6583  20 87 6A        JSR UPNLN       ;GEHE MIT NOWLN EINE ZEILE HOEHER
0264  6586  A5 00           LDA Y1
0265  6588  48              PHA             ;SICHERE INHALT VON Y1
0266  6589  84 00           STY Y1
0267  658B  A0 00           LDY #00
0268  658D  20 34 6B        JSR LIST
0269  6590  A9 0B           LDA #CUA
0270  6592  20 BC E9        JSR OUTALL
0271  6595  A2 00           LDX  #00
0272  6597  20 87 6A        JSR UPNLN
0273  659A  C4 00     J03   CPY Y1          ;WO MUSS CURSOR HIN ?
0274  659C  B0 08           BCS J04
0275  659E  C8              INY
0276  659F  A9 09           LDA #CRA
0277  65A1  20 BC E9        JSR OUTALL
0278  65A4  80 F4           BRA J03
0279  65A6  68        J04   PLA
0280  65A7  85 00           STA Y1
0281  65A9  80 0B           BRA J01
0282  65AB  20 87 6A  J02   JSR UPNLN
0283  65AE  20 0E 6B        JSR DEXO
```

```
0284  65B1  A9 0B        LDA #CUA
0285  65B3  20 BC E9     JSR OUTALL
0286  65B6  4C 4F 64  J01 JMP INPUT
0287  65B9               ;
0288  65B9               ;FUEHERE CURSOR EINE ZEILE TIEFER GGF. LISTEN
0289  65B9               ;
0290  65B9  20 21 6B  CD  JSR PLZ        ;PRUEFE AKTIVES TEXTPUFFERENDE
0291  65BC  B0 1E        BCS LIA        ;GEHE NICHT TIEFER
0292  65BE  84 01        STY Y2
0293  65C0  20 54 6B     JSR YR         ;FUEHRE Y ZUM ZEILENENDE
0294  65C3  20 58 6A     JSR ADDY       ;ERZEUGE NAECHSTE ZEILE
0295  65C6  A9 0D        LDA #$0D
0296  65C8  20 BC E9     JSR OUTALL
0297  65CB  20 08 6B     JSR INX0
0298  65CE  20 21 6B     JSR PLZ
0299  65D1  B0 09        BCS LIA
0300  65D3  20 5E 6B     JSR LIST2
0301  65D6  20 70 6B     JSR CORY       ;CORRIGIERT CURSORSTAND UND STACK
0302  65D9  4C 4F 64     JMP INPUT
0303  65DC  C4 03     LIA CPY LASTY
0304  65DE  F0 08        BEQ LIA1
0305  65E0  B1 DF        LDA (NOWLN)Y
0306  65E2  20 BC E9     JSR OUTALL
0307  65E5  C8           INY
0308  65E6  80 F4        BRA LIA
0309  65E8  07 02     LIA1 RMB 0,CBYT
0310  65EA  4C 4F 64     JMP INPUT
0311  65ED               ;
0312  65ED  20 21 6B  UPF1 JSR PLZ
0313  65F0  90 03        BCC J8
0314  65F2  4C AD 66  J9  JMP J11
0315  65F5  20 87 6B  J8  JSR PRZZ
0316  65F8  B0 F8        BCS J9
0317  65FA  20 69 6A     JSR CRTEST
0318  65FD  9F 02 05     BBS 1,CBYT,OK
0319  6600  90 03        BCC OK
0320  6602  4C A1 66     JMP J1
0321  6605  84 01     OK  STY Y2
0322  6607  48           PHA            ;SICHERE EIGEGEBENES ZEICHEN
0323  6608  20 54 6B     JSR YR
0324  660B  84 E7        STY H1         ;MERKE ZEILENLAENGE
0325  660D  A9 4F        LDA #ZANZ
0326  660F  38           SEC
0327  6610  E5 E7        SBC H1
0328  6612  20 9C 6B     JSR FUEGE
0329  6615  A4 01        LDY Y2
0330  6617  68           PLA
0331  6618  AF 02 03     BBS 2,CBYT,J2
0332  661B  20 3C E9  J4  JSR INALL
0333  661E  27 02     J2  RMB 2,CBYT
0334  6620  C9 1A        CMP #FZE
0335  6622  F0 56        BEQ J3
0336  6624  C9 0E        CMP #CUE
0337  6626  F0 46        BEQ J13
0338  6628  C9 0B        CMP #CRE
0339  662A  F0 42        BEQ J13
0340  662C  C9 0D        CMP #CDE
0341  662E  F0 3E        BEQ J13
0342  6630  C9 0A        CMP #CLE
0343  6632  F0 3A        BEQ J13
0344  6634  C9 0D        CMP #$D
0345  6636  F0 3B        BEQ J12
0346  6638  C9 20        CMP #$20
0347  663A  90 DF        BCC J4
0348  663C  C9 7F        CMP #$7F       ;PRUEFE OB DELET
0349  663E  F0 70        BEQ JDEL
0350  6640  C0 4E        CPY #ZANZ-1
0351  6642  F0 D7        BEQ J4
0352  6644  48           PHA
0353  6645  84 E7        STY H1
0354  6647  B1 DF     J5 LDA (NOWLN)Y
0355  6649  C8           INY
```

```
0356  664A  C9 0D          CMP #$D
0357  664C  D0 F9          BNE J5
0358  664E  C0 4F          CPY #ZANZ
0359  6650  F0 C9          BEQ J4
0360  6652  88       J7    DEY
0361  6653  B1 DF          LDA (NOWLN)Y
0362  6655  C8             INY
0363  6656  91 DF          STA (NOWLN)Y
0364  6658  88             DEY
0365  6659  C4 E7          CPY H1
0366  665B  D0 F5          BNE J7
0367  665D  68             PLA
0368  665E  91 DF          STA (NOWLN)Y
0369  6660  C8             INY
0370  6661  E6 01          INC Y2
0371  6663  20 BC E9       JSR OUTALL
0372  6666  20 5E 6B       JSR LIST2
0373  6669  20 70 6B       JSR CORY
0374  666C  80 AD          BRA J4
0375  666E  D7 02    J13   SMB 5,CBYT
0376  6670  48             PHA
0377  6671  80 07          BRA J3
0378  6673  C7 02    J12   SMB 4,CBYT
0379  6675  A9 0D          LDA #$D
0380  6677  20 BC E9       JSR OUTALL
0381  667A  5A       J3    PHY
0382  667B  20 54 6B       JSR YR          ;HOLE ZEILENLAENGE FUER DELET STRI
0383  667E  84 E7          STY H1
0384  6680  A9 4F          LDA #ZANZ
0385  6682  38             SEC
0386  6683  E5 E7          SBC H1
0387  6685  20 FB 6B       JSR DELSTR
0388  6688  7A             PLY
0389  6689  4F 02 15       BBR 4,CBYT,J1
0390  668C  47 02          RMB 4,CBYT
0391  668E  20 A3 6A       JSR DOWNLN
0392  6691  20 08 6B       JSR INXO
0393  6694  A0 00          LDY #0
0394  6696  20 21 6B       JSR PLZ
0395  6699  B0 12          BCS J11
0396  669B  20 5E 6B       JSR LIST2
0397  669E  20 4F 6C       JSR CURLEF
0398  66A1  DF 02 03    J1    BBS 5,CBYT,J14
0399  66A4  4C 4F 64       JMP INPUT
0400  66A7  68       J14   PLA
0401  66A8  57 02          RMB 5,CBYT
0402  66AA  4C C3 64       JMP N33
0403  66AD  4C 41 65    J11   JMP N35
0404  66B0  C0 00    JDEL  CPY #0
0405  66B2  D0 03          BNE J15
0406  66B4  4C 1B 66       JMP J4
0407  66B7  20 BC E9    J15   JSR OUTALL
0408  66BA  84 01          STY Y2
0409  66BC  C6 01          DEC Y2
0410  66BE  88             DEY
0411  66BF  88             DEY
0412  66C0  C8       J10   INY
0413  66C1  C8             INY
0414  66C2  B1 DF          LDA (NOWLN)Y
0415  66C4  88             DEY
0416  66C5  91 DF          STA (NOWLN)Y
0417  66C7  C9 0D          CMP #$D
0418  66C9  D0 F5          BNE J10
0419  66CB  A4 01          LDY Y2
0420  66CD  20 5E 6B       JSR LIST2
0421  66D0  20 70 6B       JSR CORY
0422  66D3  4C 1B 66       JMP J4
0423  66D6             ;
0424  66D6             ;FUEGE GANZE ZEILE EIN
0425  66D6             ;
0426  66D6  20 DC 66    UPF2  JSR UUPF2
0427  66D9  4C 4F 64       JMP INPUT
```

```
0428  66DC  A0 00      UUPF2  LDY #0
0429  66DE  A9 4F             LDA #ZANZ
0430  66E0  20 9C 6B          JSR FUEGE
0431  66E3  A9 15             LDA #IOLSCR
0432  66E5  20 BC E9          JSR OUTALL      ;GEBE INSERT ONE LINE AUS
0433  66E8  20 3C E9    U3    JSR INALL
0434  66EB  C9 0D             CMP #$0D
0435  66ED  F0 19             BEQ U2          ;WENN ZEICHEN EIN CR WAR DANN GEBE
0436  66EF  C9 20             CMP #$20
0437  66F1  90 F5             BCC U3          ;WENN CONTROL-FUNKTION DANN INPUT
0438  66F3  C9 7F             CMP #$7F        ;PRUEFE AUF DELET
0439  66F5  F0 39             BEQ UDEL
0440  66F7  91 DF             STA (NDWLN)Y
0441  66F9  20 BC E9          JSR OUTALL
0442  66FC  C8               INY
0443  66FD  C0 4E             CPY #ZANZ-1
0444  66FF  D0 E7             BNE U3          ;WENN ZEICHENANZAHL ERREICT DANN G
0445  6701  20 3C E9    U4    JSR INALL
0446  6704  C9 0D             CMP #$0D
0447  6706  D0 F9             BNE U4
0448  6708  C0 00       U2    CPY #0
0449  670A  F0 35             BEQ ENDREN
0450  670C  A9 0D             LDA #$D
0451  670E  91 DF             STA (NDWLN)Y
0452  6710  20 BC E9          JSR OUTALL
0453  6713  C8               INY
0454  6714  84 E7             STY H1
0455  6716  A9 4F             LDA #ZANZ
0456  6718  38               SEC
0457  6719  E5 E7             SBC H1
0458  671B  20 FB 6B          JSR DELSTR
0459  671E  A0 00             LDY #0
0460  6720  20 A3 6A          JSR DOWNLN
0461  6723  20 08 6B          JSR INXO
0462  6726  BF 02 06          BBS 3,CBYT,U1
0463  6729  20 5E 6B          JSR LIST2
0464  672C  20 4F 6C          JSR CURLEF
0465  672F  60          U1    RTS
0466  6730  C0 01       UDEL  CPY #1
0467  6732  90 B4             BCC U3
0468  6734  88               DEY
0469  6735  20 BC E9          JSR OUTALL
0470  6738  80 AE             BRA U3
0471  673A  B7 02       RENTER SMB 3,CBYT
0472  673C  20 DC 66          JSR UUPF2
0473  673F  80 F9             BRA RENTER
0474  6741              ;
0475  6741  37 02       ENDREN RMB 3,CBYT
0476  6743  84 E7             STY H1
0477  6745  A9 4F             LDA #ZANZ
0478  6747  38               SEC
0479  6748  E5 E7             SBC H1
0480  674A  20 FB 6B          JSR DELSTR
0481  674D  A9 04             LDA #CESSCR     ;CLEAR TO END OF SCREEN
0482  674F  20 BC E9          JSR OUTALL
0483  6752  86 E7             STX H1
0484  6754  20 2E 6B    U7    JSR CLLIST
0485  6757  20 21 6B          JSR PLZ
0486  675A  B0 26             BCS U6
0487  675C  E0 1B             CPX #LINE
0488  675E  D0 F4             BNE U7
0489  6760  20 5E 6B          JSR LIST2
0490  6763  87 02             SMB 0,CBYT
0491  6765  E4 E7       U8    CPX H1
0492  6767  F0 0B             BEQ U9
0493  6769  20 B7 6A          JSR UPNLN
0494  676C  A9 0B             LDA #CUA
0495  676E  20 BC E9          JSR OUTALL
0496  6771  CA               DEX
0497  6772  80 F1             BRA U8
0498  6774  7F 02 05    U9    BBR 7,CBYT,U10
0499  6777  20 70 6B          JSR CORY
```

```
0500  677A  B0 03        BRA U11
0501  677C  20 4F 6C  U10  JSR CURLEF
0502  677F  4C 4F 64  U11  JMP INPUT
0503  6782  4C 21 65  U6   JMP N24
0504  6785               ;
0505  6785  20 21 6B  DEL  JSR PLZ
0506  6788  B0 20        BCS DEL1
0507  678A  84 01        STY Y2
0508  678C  20 54 6B     JSR YR
0509  678F  C0 02        CPY #2          ;SPERRE DEL WENN ZEILE NUR NOCH EI
0510  6791  08           PHP
0511  6792  A4 01        LDY Y2
0512  6794  28           PLP
0513  6795  F0 13        BEQ DEL1
0514  6797  A9 01        LDA #1
0515  6799  20 FB 6B     JSR DELSTR
0516  679C  20 4F 6C     JSR CURLEF      ;FUEHRE CURSOR NACH LINKS
0517  679F  20 5E 6B     JSR LIST2       ;LISTE EINE ZEILE OHNE CR
0518  67A2  A9 02        LDA #CELSCR
0519  67A4  20 BC E9     JSR OUTALL
0520  67A7  20 70 6B     JSR CORY
0521  67AA  4C 4F 64  DEL1 JMP INPUT
0522  67AD               ;
0523  67AD               ;LOESCHE EINE ZEILE IM MEMO UND AUF SCREEN
0524  67AD               ;GEBE REST BIS SCREEN-ENDE AUS
0525  67AD               ;
0526  67AD  20 21 6B  UPF3 JSR PLZ
0527  67B0  90 06        BCC LALIOK
0528  67B2  20 4F 6C     JSR CURLEF
0529  67B5  4C DC 65     JMP LIA         ;CORIGIERE LASTLINE
0530  67B8  84 01     LALIOK STY Y2      ;SICHERE Y REG.
0531  67BA  20 54 6B     JSR YR          ;HOLE ZEILEN LAENGE
0532  67BD  98           TYA
0533  67BE  A0 00        LDY #0
0534  67C0  20 FB 6B     JSR DELSTR
0535  67C3  A9 00        LDA #0
0536  67C5  48           PHA
0537  67C6  A4 01        LDY Y2
0538  67C8  20 4F 6C     JSR CURLEF
0539  67CB  A9 04        LDA #CESSCR     ;CLEAR TO END OF SCREEN
0540  67CD  20 BC E9     JSR OUTALL
0541  67D0  20 21 6B  LISTLN JSR PLZ
0542  67D3  B0 2F        BCS LALIN
0543  67D5  E0 18        CPX #LINE
0544  67D7  F0 08        BEQ LIOHCU
0545  67D9  20 2E 6B     JSR CLLIST
0546  67DC  68           PLA
0547  67DD  1A           INC A
0548  67DE  48           PHA
0549  67DF  B0 EF        BRA LISTLN
0550  67E1  20 5E 6B  LIOHCU JSR LIST2
0551  67E4  20 4F 6C     JSR CURLEF
0552  67E7  20 70 6B  NOLIST JSR CORY
0553  67EA  68        CUCOR PLA
0554  67EB  F0 0F        BEQ CUOK
0555  67ED  3A           DEC A
0556  67EE  48           PHA
0557  67EF  A9 0B        LDA #CUA
0558  67F1  20 BC E9     JSR OUTALL
0559  67F4  20 0E 6B     JSR DEXO
0560  67F7  20 87 6A     JSR UPNLN
0561  67FA  B0 EE        BRA CUCOR
0562  67FC  20 21 6B  CUOK JSR PLZ
0563  67FF  B0 AC        BCS UPF3
0564  6801  4C 4F 64     JMP INPUT
0565  6804  C4 03     LALIN CPY LASTY
0566  6806  F0 DF        BEQ NOLIST
0567  6808  B1 DF        LDA (NOWLN)Y
0568  680A  20 BC E9     JSR OUTALL
0569  680D  C8           INY
0570  680E  B0 F4        BRA LALIN
0571  6810               ;
```

```
0572  6810  A9 03       UPF4   LDA #CS
0573  6812  20 BC E9             JSR OUTALL
0574  6815  A9 7F               LDA #<TABA
0575  6817  85 EB               STA HNOWLN
0576  6819  A9 6C               LDA #>TABA
0577  681B  85 EC               STA HNOWLN+1
0578  681D  B2 EB       TABA3  LDA (HNOWLN)
0579  681F  F0 0B               BEQ TABA1
0580  6821  20 BC E9             JSR OUTALL
0581  6824  E6 EB               INC HNOWLN
0582  6826  D0 F5               BNE TABA3
0583  6828  E6 EC               INC HNOWLN+1
0584  682A  80 F1               BRA TABA3
0585  682C  4C 4F 64   TABA1   JMP INPUT
0586  682F                     ;
0587  682F  A9 01       VTRLIN LDA #1
0588  6831  84 01               STY Y2
0589  6833  20 9C 6B             JSR FUEGE
0590  6836  A9 0D               LDA #$D
0591  6838  91 DF               STA (NOWLN)Y
0592  683A  20 4F 6C             JSR CURLEF
0593  683D  A9 04               LDA #CESSCR
0594  683F  20 BC E9             JSR OUTALL
0595  6842  E7 02               SMB 6,CBYT
0596  6844  F7 02               SMB 7,CBYT
0597  6846  86 E7               STX H1
0598  6848  E0 18               CPX #LINE
0599  684A  D0 02               BNE NCORX
0600  684C  C6 E7               DEC H1
0601  684E  4C 6B 68   NCORX   JMP CA1
0602  6851                     ;
0603  6851                     ;UP FUER GEBE AB TEXTA 20 ZEILEN AUS
0604  6851                     ;
0605  6851  A5 E3       CA     LDA TEXTA
0606  6853  85 DF               STA NOWLN
0607  6855  A5 E4               LDA TEXTA+1
0608  6857  85 E0               STA NOWLN+1
0609  6859  E7 02               SMB 6,CBYT
0610  685B  64 E7               STZ H1
0611  685D  20 21 6B   CA3      JSR PLZ
0612  6860  B0 7D               BCS LD3
0613  6862  A2 00               LDX #0
0614  6864  A0 00               LDY #0
0615  6866  A9 0E               LDA #CHSCR
0616  6868  20 BC E9             JSR OUTALL
0617  686B  20 21 6B   CA1      JSR PLZ
0618  686E  B0 23               BCS CA2
0619  6870  20 2E 6B             JSR CLLIST
0620  6873  20 21 6B             JSR PLZ
0621  6876  B0 1B               BCS CA2
0622  6878  E0 18               CPX #LINE
0623  687A  D0 EF               BNE CA1
0624  687C  20 5E 6B             JSR LIST2
0625  687F  20 4F 6C             JSR CURLEF
0626  6882  87 02               SMB 0,CBYT
0627  6884  6F 02 05             BBR 6,CBYT,CA4 ;FUERE CURSOR IN GEWUENSCHTE ZEILE
0628  6887  67 02       CA5    RMB 6,CBYT
0629  6889  4C 65 67             JMP UB
0630  688C  A9 0C       CA4    LDA #LINE2
0631  688E  85 E7               STA H1
0632  6890  4C 65 67             JMP UB
0633  6893  EF 02 F1   CA2      BBS 6,CBYT,CA5
0634  6896  4C 21 65             JMP N24
0635  6899                     ;
0636  6899                     ;LU GEHE XX ZEILEN TIEFER UND LIST DANN 20 AUF
0637  6899                     ;
0638  6899  A9 01       LU     LDA #CLSCR
0639  689B  20 BC E9             JSR OUTALL
0640  689E  20 FD E3             JSR RBYTE
0641  68A1  20 5E 6C             JSR DEZ
0642  68A4  48         LU2A   PHA              ;ZWEITER EINSPRUNG MIT ZEILENANZAH
0643  68A5  A9 0E               LDA #CHSCR
```

```
0644  68A7  20 BC E9          JSR OUTALL
0645  68AA  68               PLA
0646  68AB  F0 0B            BEQ LU4
0647  68AD  20 14 6B  LU1     JSR PTA
0648  68B0  B0 09            BCS LU3
0649  68B2  20 B7 6A          JSR UPNLN
0650  68B5  3A               DEC A
0651  68B6  D0 F5            BNE LU1
0652  68B8  4C 5D 68  LU4     JMP CA3
0653  68BB  4C 51 68  LU3     JMP CA
0654  68BE  8A        LU2     TXA
0655  68BF  18                CLC
0656  68C0  69 18             ADC #LINE
0657  68C2  80 E0             BRA LU2A
0658  68C4             ;
0659  68C4             ;LD GEHE XX ZEILEN TIEFER UND LIST DANN 20 AUF
0660  68C4             ;
0661  68C4  A9 01      LD      LDA #CLSCR
0662  68C6  20 BC E9           JSR OUTALL
0663  68C9  20 FD E3           JSR RBYTE       ;HOLE 2 ASCII ZAHLEN
0664  68CC  20 5E 6C           JSR DEZ
0665  68CF  F0 0B      LD2A    BEQ LD4
0666  68D1  20 21 6B  LD1     JSR PLZ
0667  68D4  B0 09            BCS LD3
0668  68D6  20 A3 6A          JSR DOWNLN
0669  68D9  3A                DEC A
0670  68DA  D0 F5            BNE LD1
0671  68DC  4C 5D 68  LD4     JMP CA3
0672  68DF  4C F3 64  LD3     JMP CH
0673  68E2  86 E7      LD2     STX H1
0674  68E4  38                SEC
0675  68E5  A9 18             LDA #LINE
0676  68E7  E5 E7             SBC H1
0677  68E9  80 E4             BRA LD2A
0678  68EB             ;
0679  68EB             ;LEFT FUEHRE CURSOR NACH LINKS
0680  68EB  20 4F 6C  LEFT    JSR CURLEF
0681  68EE  4C 4F 64          JMP INPUT
0682  68F1             ;
0683  68F1             ;FF FINDE FOLGE WEITER MIT SPACE OK MIT RETURN
0684  68F1             ;
0685  68F1  A9 01      FF      LDA #CLSCR
0686  68F3  20 BC E9           JSR OUTALL
0687  68F6  A9 2E             LDA #'.
0688  68F8  20 BC E9           JSR OUTALL
0689  68FB  A9 46             LDA #'F
0690  68FD  20 BC E9           JSR OUTALL
0691  6900  A9 20             LDA #'
0692  6902  20 BC E9           JSR OUTALL
0693  6905  A0 00             LDY #00
0694  6907  20 3C E9  FF2     JSR INALL
0695  690A  C9 0D             CMP #$D
0696  690C  F0 0F             BEQ FF1
0697  690E  C9 7F             CMP #$7F
0698  6910  F0 2F             BEQ DELFF
0699  6912  20 BC E9           JSR OUTALL
0700  6915  99 E7 00           STA STRING,Y
0701  6918  C8                INY
0702  6919  C0 14             CPY #20
0703  691B  D0 EA             BNE FF2
0704  691D  84 04      FF1     STY STRLEN
0705  691F  A2 00      FF3     LDX #0
0706  6921  A0 00             LDY #0
0707  6923  B1 DF      FNEXT   LDA (NOWLN)Y
0708  6925  D5 E7             CMP STRING,X
0709  6927  F0 22             BEQ FOUNDB
0710  6929  C8                INY
0711  692A  C9 0D             CMP #$D
0712  692C  D0 F5             BNE FNEXT
0713  692E  20 5B 6A           JSR ADDY
0714  6931  20 21 6B           JSR PLZ
0715  6934  90 ED             BCC FNEXT
```

```
0717  6936  A5 E3       NOTFD1 LDA TEXTA
0718  6938  85 DF              STA NOWLN
0719  693A  A5 E4              LDA TEXTA+1
0720  693C  85 E0              STA NOWLN+1
0721  693E  4C 10 6B           JMP UPF4
0722  6941  C0 00       DELFF  CPY #0
0723  6943  F0 C2              BEQ FF2
0724  6945  88                 DEY
0725  6946  20 BC E9           JSR OUTALL
0726  6949  80 BC              BRA FF2
0727  694B                    ;
0728  694B  5A          FOUNDB PHY
0729  694C  E8          FNCHR  INX
0730  694D  E4 04              CPX STRLEN
0731  694F  F0 0D              BEQ FOUNDS
0732  6951  C8                 INY
0733  6952  B1 DF              LDA (NOWLN)Y
0734  6954  D5 E7              CMP STRING,X
0735  6956  F0 F4              BEQ FNCHR
0736  6958  A2 00              LDX #0
0737  695A  7A                 PLY
0738  695B  C8                 INY
0739  695C  80 C5              BRA FNEXT
0740  695E  84 00       FOUNDS STY Y1
0741  6960  7A                 PLY
0742  6961  A2 0C              LDX #LINE2
0743  6963  A0 00              LDY #0
0744  6965  A9 0E              LDA #CHSCR
0745  6967  20 BC E9           JSR OUTALL
0746  696A  20 14 6B    FFLIS4 JSR PTA
0747  696D  B0 06              BCS FFLIS5
0748  696F  20 87 6A           JSR UPNLN
0749  6972  CA                 DEX
0750  6973  D0 F5              BNE FFLIS4
0751  6975  86 01       FFLIS5 STX Y2
0752  6977  38                 SEC
0753  6978  A9 0C              LDA #LINE2
0754  697A  E5 01              SBC Y2
0755  697C  85 01              STA Y2
0756  697E  A2 00              LDX #0
0757  6980  20 21 6B    FFLIS2 JSR PLZ
0758  6983  B0 0C              BCS FFLIS3
0759  6985  20 2E 6B           JSR CLLIST
0760  6988  E0 18              CPX #LINE
0761  698A  D0 F4              BNE FFLIS2
0762  698C  20 5E 6B           JSR LIST2
0763  698F  80 34              BRA FFCU
0764  6991  C4 03       FFLIS3 CPY LASTY
0765  6993  F0 16              BEQ FFCU1
0766  6995  B1 DF              LDA (NOWLN)Y
0767  6997  20 BC E9           JSR OUTALL
0768  699A  C8                 INY
0769  699B  80 F4              BRA FFLIS3
0770  699D  20 A3 6A    FFLIS1 JSR DOWNLN
0771  69A0  20 21 6B           JSR PLZ
0772  69A3  90 03              BCC FF4
0773  69A5  4C 36 69           JMP NOTFD1
0774  69A8  4C 1F 69    FF4    JMP FF3
0775  69AB  C0 00       FFCU1  CPY #0
0776  69AD  F0 11              BEQ FFCU2
0777  69AF  A9 0A              LDA #CDA
0778  69B1  20 BC E9           JSR OUTALL
0779  69B4  A9 04              LDA #CESSCR
0780  69B6  20 BC E9           JSR OUTALL
0781  69B9  A9 0B              LDA #CUA
0782  69BB  20 BC E9           JSR OUTALL
0783  69BE  80 05              BRA FFCU
0784  69C0  A9 04       FFCU2  LDA #CESSCR
0785  69C2  20 BC E9           JSR OUTALL
0786  69C5  20 87 6A    FFCU   JSR UPNLN
0787  69C8  A9 0B              LDA #CUA
0788  69CA  20 BC E9           JSR OUTALL
```

```
0789  69CD  20 0E 6B        JSR DEXO
0790  69D0  E4 01           CPX Y2
0791  69D2  D0 F1           BNE FFCU
0792  69D4  A5 00           LDA Y1
0793  69D6  85 01           STA Y2
0794  69D8  20 70 6B        JSR CORY
0795  69DB  20 3C E9        JSR INALL
0796  69DE  C9 0D           CMP #$D
0797  69E0  F0 5E           BEQ R1
0798  69E2  80 B9           BRA FFLIS1
0799  69E4                 ;
0800  69E4  A9 01        START2 LDA #1
0801  69E6  85 02           STA CBYT
0802  69E8  A0 00           LDY #0
0803  69EA  84 03           STY LASTY
0804  69EC  4C 51 68        JMP CA
0805  69EF                 ;
0806  69EF  A9 03        UEXIT  LDA #C5
0807  69F1  20 BC E9           JSR OUTALL
0808  69F4  A5 E1           LDA LASTZ
0809  69F6  A6 E2           LDX LASTZ+1
0810  69F8  85 DF           STA NOWLN
0811  69FA  86 E0           STX NOWLN+1
0812  69FC  A4 03           LDY LASTY
0813  69FE  F0 10           BEQ EX1
0814  6A00  A9 0D           LDA #$D
0815  6A02  91 DF           STA (NOWLN),Y
0816  6A04  C8              INY
0817  6A05  20 58 6A        JSR ADDY
0818  6A08  A5 DF           LDA NOWLN
0819  6A0A  85 E1           STA LASTZ
0820  6A0C  A5 E0           LDA NOWLN+1
0821  6A0E  85 E2           STA LASTZ+1
0822  6A10  A9 00        EX1    LDA #0
0823  6A12  91 DF           STA (NOWLN)Y
0824  6A14  60              RTS
0825  6A15                 ;
0826  6A15  20 EF 69     TEXIT  JSR UEXIT
0827  6A18  A5 E3           LDA TEXTA
0828  6A1A  85 DF           STA NOWLN
0829  6A1C  A5 E4           LDA TEXTA+1
0830  6A1E  85 E0           STA NOWLN+1
0831  6A20  4C A1 E1        JMP MONIT
0832  6A23                 ;
0833  6A23  20 21 6B     CLEARE JSR PLZ          ;LOESCHT ALLES NACH CURSOR
0834  6A26  B0 18            BCS R1
0835  6A28  C0 00            CPY #0
0836  6A2A  F0 14            BEQ R1
0837  6A2C  84 E7            STY H1
0838  6A2E  20 54 6B         JSR YR
0839  6A31  98               TYA
0840  6A32  38               SEC
0841  6A33  E5 E7            SBC H1
0842  6A35  A4 E7            LDY H1
0843  6A37  3A               DEC A
0844  6A38  20 FB 6B         JSR DELSTR
0845  6A3B  A9 02            LDA #CELSCR     ;CLEAR TO END OF LINE
0846  6A3D  20 BC E9         JSR OUTALL
0847  6A40  4C 4F 64     R1     JMP INPUT
0848  6A43                 ;
0849  6A43                 ;UNTERPROGRAMME
0850  6A43                 ;
0851  6A43                 ;ERHOEHE NOWLN
0852  6A43                 ;
0853  6A43  E6 DF        INCNLN INC NOWLN
0854  6A45  D0 02            BNE U01
0855  6A47  E6 E0            INC NOWLN+1
0856  6A49  60           U01    RTS
0857  6A4A                 ;
0858  6A4A                 ;ERNIEDRIGE NOWLN
0859  6A4A                 ;
0860  6A4A  48           DECNLN PHA
```

```
0861  6A4B  A5 DF        LDA NOWLN
0862  6A4D  3A           DEC A
0863  6A4E  85 DF        STA NOWLN
0864  6A50  C9 FF        CMP #$FF
0865  6A52  D0 02        BNE NO2
0866  6A54  C6 E0        DEC NOWLN+1
0867  6A56  68     NO2   PLA
0868  6A57  60           RTS
0869  6A58               ;
0870  6A58               ;ADDIERE NOWLN UND Y REG ERG IN NOWLN Y = 00
0871  6A58               ;
0872  6A58  48     ADDY  PHA
0873  6A59  18           CLC
0874  6A5A  98           TYA
0875  6A5B  65 DF        ADC NOWLN
0876  6A5D  85 DF        STA NOWLN
0877  6A5F  A5 E0        LDA NOWLN+1
0878  6A61  69 00        ADC #0
0879  6A63  85 E0        STA NOWLN+1
0880  6A65  A0 00        LDY #0
0881  6A67  68           PLA
0882  6A68  60           RTS
0883  6A69               ;
0884  6A69               ;PRUEFE CURSO UNTER AKTIVEM ZEICHEN STEHT ERGEBNIS IN C
0885  6A69               ;
0886  6A69  48     CRTEST PHA
0887  6A6A  DA           PHX
0888  6A6B  5A           PHY
0889  6A6C  A0 00        LDY #0
0890  6A6E  A9 0D        LDA #$D
0891  6A70  C8     NOTOD  INY
0892  6A71  D1 DF        CMP (NOWLN),Y
0893  6A73  D0 FB        BNE NOTOD
0894  6A75  68           PLA             ;LADE Y VOM STAPEL IN AKKU ZUM VER
0895  6A76  48           PHA
0896  6A77  84 00        STY Y1
0897  6A79  C5 00        CMP Y1
0898  6A7B  F0 04        BEQ SETV
0899  6A7D  17 02        RMB 1,CBYT
0900  6A7F  80 02        BRA NOSETV
0901  6A81  97 02  SETV  SMB 1,CBYT
0902  6A83  7A    NOSETV PLY
0903  6A84  FA           PLX
0904  6A85  68           PLA
0905  6A86  60           RTS
0906  6A87               ;
0907  6A87               ;GEHE MIT NOWLN EINE ZEILE HOEHER
0908  6A87               ;
0909  6A87  48     UPNLN  PHA
0910  6A88  20 14 6B     JSR PTA
0911  6A8B  B0 14        BCS N23
0912  6A8D  20 4A 6A     JSR DECNLN
0913  6A90  20 4A 6A  NO1  JSR DECNLN
0914  6A93  20 14 6B     JSR PTA
0915  6A96  B0 09        BCS N23
0916  6A98  A9 0D        LDA #$D
0917  6A9A  D2 DF        CMP (NOWLN)
0918  6A9C  D0 F2        BNE NO1
0919  6A9E  20 43 6A     JSR INCNLN
0920  6AA1  68     N23   PLA
0921  6AA2  60           RTS
0922  6AA3               ;
0923  6AA3               ;GEHE MIT NOWLN EINE ZEILE TIEFER
0924  6AA3               ;
0925  6AA3  48     DOWNLN PHA
0926  6AA4  A9 0D        LDA #$D
0927  6AA6  20 43 6A  NO3  JSR INCNLN
0928  6AA9  D2 DF        CMP (NOWLN)
0929  6AAB  D0 F9        BNE NO3
0930  6AAD  20 43 6A     JSR INCNLN
0931  6AB0  68           PLA
0932  6AB1  60           RTS
```

```
0934  6AB2                    ;DIESES PROGRAMM TESTEE OB TEXTPUFFER VOLL IST UND BRING
0935  6AB2                    ;
0936  6AB2  48                TESTAD PHA
0937  6AB3  A5 00                     LDA Y1
0938  6AB5  48                       PHA
0939  6AB6  18                       CLC
0940  6AB7  A5 DF                     LDA NOWLN
0941  6AB9  84 00                     STY Y1
0942  6ABB  65 00                     ADC Y1
0943  6ABD  08                       PHP
0944  6ABE  C5 E5                     CMP TEXTE
0945  6AC0  90 13                     BCC N04
0946  6AC2  A5 E0                     LDA NOWLN+1
0947  6AC4  28                       PLP
0948  6AC5  69 00                     ADC #0
0949  6AC7  C5 E6                     CMP TEXTE+1
0950  6AC9  90 0B                     BCC N05
0951  6ACB  68                       PLA
0952  6ACC  68                       PLA
0953  6ACD  68                       PLA
0954  6ACE  68                       PLA
0955  6ACF  20 09 64                 JSR EDITOR
0956  6AD2  4C 36 64                 JMP BEGIN2
0957  6AD5  28                N04    PLP
0958  6AD6  68                N05    PLA
0959  6AD7  85 00                     STA Y1
0960  6AD9  68                       PLA
0961  6ADA  60                       RTS
0962  6ADB                    ;
0963  6ADB                    ;TESTE OB TEXTPUFFER VOLL IST ABER NUR FUER UPF FUNKTION
0964  6ADB                    ;
0965  6ADB  48                TESTLA PHA
0966  6ADC  18                       CLC
0967  6ADD  A5 E1                     LDA LASTZ
0968  6ADF  65 03                     ADC LASTY
0969  6AE1  69 4F                     ADC #ZANZ
0970  6AE3  08                       PHP
0971  6AE4  C5 E5                     CMP TEXTE
0972  6AE6  90 10                     BCC NL04
0973  6AE8  A5 E2                     LDA LASTZ+1
0974  6AEA  28                       PLP
0975  6AEB  69 00                     ADC #0
0976  6AED  C5 E6                     CMP TEXTE+1
0977  6AEF  90 0B                     BCC NL05
0978  6AF1  68                       PLA
0979  6AF2  68                       PLA
0980  6AF3  68                       PLA
0981  6AF4  68                       PLA
0982  6AF5  4C 15 6A                 JMP TEXIT
0983  6AF8  28                NL04   PLP
0984  6AF9  68                NL05   PLA
0985  6AFA  60                       RTS
0986  6AFB                    ;
0987  6AFB                    ;SICHERE WERTE DER ZULETZT AKTIVEN ZEILE
0988  6AFB                    ;
0989  6AFB  48                LASTNL PHA
0990  6AFC  A5 DF                     LDA NOWLN
0991  6AFE  85 E1                     STA LASTZ
0992  6B00  A5 E0                     LDA NOWLN+1
0993  6B02  85 E2                     STA LASTZ+1
0994  6B04  84 03                     STY LASTY
0995  6B06  68                       PLA
0996  6B07  60                       RTS
0997  6B08                    ;
0998  6B08                    ;ERHOEHE X REG.
0999  6B08                    ;
1000  6B08  E0 18             INX0   CPX #LINE
1001  6B0A  F0 01                     BEQ N10
1002  6B0C  E8                       INX
1003  6B0D  60                N10    RTS
1004  6B0E                    ;
```

```
1005  6B0E                    ;ERNIEDRIGE X REG.
1006  6B0E                    ;
1007  6B0E  E0 00     DEX0    CPX #0
1008  6B10  F0 01             BEQ N11
1009  6B12  CA                DEX
1010  6B13  60        N11     RTS
1011  6B14                    ;
1012  6B14                    ;PRUEFE OB TEXTANFANG ERREICHT WURDE ERG. ERREICHT C=1
1013  6B14                    ;
1014  6B14  48        PTA     PHA
1015  6B15  A5 E4             LDA TEXTA+1
1016  6B17  C5 E0             CMP NOWLN+1
1017  6B19  90 04             BCC N12
1018  6B1B  A5 E3             LDA TEXTA
1019  6B1D  C5 DF             CMP NOWLN
1020  6B1F  68        N12     PLA
1021  6B20  60                RTS
1022  6B21                    ;
1023  6B21                    ;PRUEFE OB TEXTENDE ERREICHT WURDE WENN ERREICHT C=1
1024  6B21                    ;
1025  6B21  48        PLZ     PHA
1026  6B22  A5 E0             LDA NOWLN+1
1027  6B24  C5 E2             CMP LASTZ+1
1028  6B26  90 04             BCC N14
1029  6B28  A5 DF             LDA NOWLN
1030  6B2A  C5 E1             CMP LASTZ
1031  6B2C  68        N14     PLA
1032  6B2D  60                RTS
1033  6B2E                    ;
1034  6B2E                    ;FUERE CURSOR GANZ NACH LINKS UND GEBE EINE ZEILE AUS +
1035  6B2E                    ;
1036  6B2E  48        CLLIST  PHA
1037  6B2F  20 4F 6C          JSR CURLEF
1038  6B32  80 01             BRA N19
1039  6B34  48        LIST    PHA
1040  6B35  B1 DF     N19     LDA (NOWLN)Y
1041  6B37  C9 0D             CMP #$D
1042  6B39  F0 06             BEQ N20
1043  6B3B  20 BC E9          JSR OUTALL
1044  6B3E  C8                INY
1045  6B3F  80 F4             BRA N19
1046  6B41  48        N20     PHA
1047  6B42  A9 02             LDA #CELSCR
1048  6B44  20 BC E9          JSR OUTALL
1049  6B47  68                PLA
1050  6B48  20 BC E9          JSR OUTALL
1051  6B4B  C8                INY
1052  6B4C  20 58 6A          JSR ADDY
1053  6B4F  20 08 6B          JSR INXO
1054  6B52  68                PLA
1055  6B53  60                RTS
1056  6B54                    ;
1057  6B54                    ;FUEHRE NUR Y ZUM ZEILENENDE IM SPEICHER
1058  6B54                    ;
1059  6B54  A0 00     YR      LDY #0
1060  6B56  B1 DF     YR1     LDA (NOWLN)Y
1061  6B58  C8                INY
1062  6B59  C9 0D             CMP #$D
1063  6B5B  D0 F9             BNE YR1
1064  6B5D  60                RTS
1065  6B5E                    ;
1066  6B5E                    ;LISTE EINE ZEILE OHNE CR
1067  6B5E                    ;
1068  6B5E  B1 DF     LIST2   LDA (NOWLN)Y
1069  6B60  C9 0D             CMP #$D
1070  6B62  F0 06             BEQ READY
1071  6B64  20 BC E9          JSR OUTALL
1072  6B67  C8                INY
1073  6B68  80 F4             BRA LIST2
1074  6B6A  A9 02     READY   LDA #CELSCR
1075  6B6C  20 BC E9          JSR OUTALL
1076  6B6F  60                RTS
```

```
1077  6B70                    ;
1078  6B70                    ;CORRIGIERT Y AUF SCREEN VERWENDET Y2
1079  6B70                    ;
1080  6B70  C4 01     CORY    CPY Y2
1081  6B72  F0 12             BEQ CORY2
1082  6B74  90 08             BCC CORY1
1083  6B76  88                DEY
1084  6B77  A9 0B             LDA #CLA
1085  6B79  20 BC E9          JSR OUTALL
1086  6B7C  80 F2             BRA CORY
1087  6B7E  C8        CORY1   INY
1088  6B7F  A9 09             LDA #CRA
1089  6B81  20 BC E9          JSR OUTALL
1090  6B84  80 EA             BRA CORY
1091  6B86  60        CORY2   RTS
1092  6B87                    ;
1093  6B87                    ;PRUEFE ZEICHENANZAHL IN AKTUELLER ZEILE
1094  6B87                    ;ERGEBNIS IN CARRY
1095  6B87                    ;
1096  6B87  20 21 6B  PRZZ    JSR PLZ         ;PRUEFE OB IN LETZER ZEILE
1097  6B8A  B0 0F             BCS PRZZ1
1098  6B8C  5A                PHY
1099  6B8D  48                PHA
1100  6B8E  A0 00             LDY #0
1101  6B90  B1 DF     PRZZ2   LDA (NOWLN),Y
1102  6B92  C8                INY
1103  6B93  C9 0D             CMP #$D         ;PRUEFE OB ZEILEN ENDE ERREICHT
1104  6B95  D0 F9             BNE PRZZ2
1105  6B97  C0 4F             CPY #ZANZ       ;VERGLEICHE Y REG.MIT GEN.ZANZ
1106  6B99  68                PLA
1107  6B9A  7A                PLY
1108  6B9B  60        PRZZ1   RTS
1109  6B9C                    ;
1110  6B9C                    ;DIESES PROGRAMM ERWEITERT MEMORY UM AKKU-BYT-ZAHL
1111  6B9C  20 DB 6A  FUEGE   JSR TESTLA
1112  6B9F  48                PHA
1113  6BA0  18                CLC
1114  6BA1  98                TYA
1115  6BA2  65 DF             ADC NOWLN       ;BERECHNE LETZTES ZU UBERTRAGENDES
1116  6BA4  85 EB             STA HNOWLN      ;ERGEBNIS IN HNOWLN
1117  6BA6  A9 00             LDA #0
1118  6BA8  65 E0             ADC NOWLN+1
1119  6BAA  85 EC             STA HNOWLN+1
1120  6BAC                    ;
1121  6BAC  18                CLC
1122  6BAD  A5 03             LDA LASTY
1123  6BAF  65 E1             ADC LASTZ       ;BERECHNE ERSTES ZU UBERTR. ZEICHE
1124  6BB1  85 E9             STA HLAST
1125  6BB3  A9 00             LDA #0
1126  6BB5  65 E2             ADC LASTZ+1
1127  6BB7  85 EA             STA HLAST+1
1128  6BB9                    ;
1129  6BB9  68                PLA             ;HOLE TRANSFERWEITE
1130  6BBA  48                PHA
1131  6BBB  18                CLC
1132  6BBC  65 E9             ADC HLAST
1133  6BBE  85 ED             STA TRANSZ      ;TRANSFERZIELADRESSE
1134  6BC0  A5 EA             LDA HLAST+1
1135  6BC2  69 00             ADC #0
1136  6BC4  85 EE             STA TRANSZ+1
1137  6BC6                    ;
1138  6BC6  68                PLA             ;HOLE TRANSFERWEITE UND BERECHNE L
1139  6BC7  18                CLC
1140  6BC8  65 E1             ADC LASTZ
1141  6BCA  85 E1             STA LASTZ
1142  6BCC  A5 E2             LDA LASTZ+1
1143  6BCE  69 00             ADC #0
1144  6BD0  85 E2             STA LASTZ+1
1145  6BD2                    ;
1146  6BD2  B2 E9     TR4     LDA (HLAST)
1147  6BD4  92 ED             STA (TRANSZ)
1148  6BD6  A5 EA             LDA HLAST+1
```

```
1149  6BD8  C5 EC           CMP HNOWLN+1
1150  6BDA  D0 08           BNE TR3
1151  6BDC  A5 E9           LDA HLAST
1152  6BDE  C5 EB           CMP HNOWLN
1153  6BE0  D0 02           BNE TR3
1154  6BE2  80 16           BRA TROK
1155  6BE4  C6 E9     TR3   DEC HLAST
1156  6BE6  A5 E9           LDA HLAST
1157  6BE8  C9 FF           CMP #$FF
1158  6BEA  D0 02           BNE TR1
1159  6BEC  C6 EA           DEC HLAST+1
1160  6BEE  C6 ED     TR1   DEC TRANSZ
1161  6BF0  A5 ED           LDA TRANSZ
1162  6BF2  C9 FF           CMP #$FF
1163  6BF4  D0 DC           BNE TR4
1164  6BF6  C6 EE           DEC TRANSZ+1
1165  6BF8  80 D8           BRA TR4
1166  6BFA  60        TROK  RTS
1167  6BFB                  ;
1168  6BFB                  ;DIESES PROGR. LOESCHT AB NOWLN+Y BIS NOWLN+Y+AKKU
1169  6BFB                  ;
1170  6BFB  48        DELSTR PHA
1171  6BFC  18               CLC
1172  6BFD  98               TYA
1173  6BFE  65 DF            ADC NOWLN
1174  6C00  85 ED            STA TRANSZ
1175  6C02  A9 00            LDA #0
1176  6C04  65 E0            ADC NOWLN+1
1177  6C06  85 EE            STA TRANSZ+1
1178  6C08                   ;
1179  6C08  18               CLC
1180  6C09  68               PLA
1181  6C0A  65 ED            ADC TRANSZ
1182  6C0C  85 EB            STA HNOWLN
1183  6C0E  A9 00            LDA #0
1184  6C10  65 EE            ADC TRANSZ+1
1185  6C12  85 EC            STA HNOWLN+1
1186  6C14                   ;
1187  6C14  18               CLC
1188  6C15  A5 E1            LDA LASTZ
1189  6C17  65 03            ADC LASTY
1190  6C19  85 E9            STA HLAST
1191  6C1B  A5 E2            LDA LASTZ+1
1192  6C1D  69 00            ADC #0
1193  6C1F  85 EA            STA HLAST+1
1194  6C21                   ;
1195  6C21  B2 EB     TRD4  LDA (HNOWLN)
1196  6C23  92 ED           STA (TRANSZ)
1197  6C25  A5 EC           LDA HNOWLN+1
1198  6C27  C5 EA           CMP HLAST+1
1199  6C29  D0 08           BNE TRD3
1200  6C2B  A5 EB           LDA HNOWLN
1201  6C2D  C5 E9           CMP HLAST
1202  6C2F  D0 02           BNE TRD3
1203  6C31  80 0E           BRA TRDOK
1204  6C33  E6 EB     TRD3  INC HNOWLN
1205  6C35  D0 02           BNE TRD1
1206  6C37  E6 EC           INC HNOWLN+1
1207  6C39  E6 ED     TRD1  INC TRANSZ
1208  6C3B  D0 E4           BNE TRD4
1209  6C3D  E6 EE           INC TRANSZ+1
1210  6C3F  80 E0           BRA TRD4
1211  6C41  A5 ED     TRDOK LDA TRANSZ
1212  6C43  38              SEC
1213  6C44  E5 03           SBC LASTY
1214  6C46  85 E1           STA LASTZ
1215  6C48  A5 EE           LDA TRANSZ+1
1216  6C4A  E9 00           SBC #0
1217  6C4C  85 E2           STA LASTZ+1
1218  6C4E  60              RTS
1219  6C4F                  ;
```

```
1220  6C4F                  ;DIESES PROGRAMM FUEHRT CURSOR NACH LINKS
1221  6C4F                  ;
1222  6C4F  48              CURLEF PHA
1223  6C50  C0 00           CF1    CPY #0
1224  6C52  F0 08                  BEQ CF2
1225  6C54  88                     DEY
1226  6C55  A9 08                  LDA #CLA
1227  6C57  20 BC E9               JSR OUTALL
1228  6C5A  80 F4                  BRA CF1
1229  6C5C  68              CF2    PLA
1230  6C5D  60                     RTS
1231  6C5E                  ;
1232  6C5E                  ;HEX - DEC. UMRECHNUNG
1233  6C5E                  ;
1234  6C5E  A2 00           DEZ    LDX #0
1235  6C60  86 01                  STX Y2
1236  6C62  48                     PHA
1237  6C63  29 F0                  AND #$F0
1238  6C65  4A                     LSR A
1239  6C66  4A                     LSR A
1240  6C67  4A                     LSR A
1241  6C68  4A                     LSR A
1242  6C69  85 00                  STA Y1
1243  6C6B  F0 0B                  BEQ DEZ1
1244  6C6D  A9 00                  LDA #0
1245  6C6F  18              DEZ2   CLC
1246  6C70  69 0A                  ADC #10
1247  6C72  C6 00                  DEC Y1
1248  6C74  D0 F9                  BNE DEZ2
1249  6C76  85 01                  STA Y2
1250  6C78  68              DEZ1   PLA
1251  6C79  29 0F                  AND #$F
1252  6C7B  18                     CLC
1253  6C7C  65 01                  ADC Y2
1254  6C7E  60                     RTS
1255  6C7F  41 20 4C        TABA   .BYT 'A LISTE AUFWAERTS 20 ZEILEN',$D
1256  6C9B  42 20 41               .BYT 'B AUSGABE DER LETZTEN ZEILEN',$D
1257  6CB8  43 20 43               .BYT 'C CLEAR LINE',$D
1258  6CC5  44 20 47               .BYT 'D GEHE XX ZEILEN ABWAERTS UND '
1259  6CE3  4C 49 53               .BYT 'LISTE AUF',$D
1260  6CED  45 20 43               .BYT 'E CLEAR TO END OF LINE',$D
1261  6D04  46 20 46               .BYT 'F FINDE FOLGE',$D
1262  6D12  49 20 46               .BYT 'I FUEGE EINE ZEILE EIN',$D
1263  6D29  4A 20 43               .BYT 'J CURSOR LINKS',$D
1264  6D38  4B 20 43               .BYT 'K CURSOR RECHTS',$D
1265  6D48  4C 20 4C               .BYT 'L LISTE 20 ZEILEN',$D
1266  6D5A  4D 20 43               .BYT 'M CURSOR DOWN',$D
1267  6D68  4E 20 43               .BYT 'N CURSOR UP',$D
1268  6D74  57 20 43               .BYT 'W CURSOR GANZ NACH LINKS',$D
1269  6D8D  52 20 52               .BYT 'R R-ENTER EDITOR',$D
1270  6D9E  54 20 4C               .BYT 'T LISTE AB TEXTANFANG 20 ZEILEN',$D
1271  6DBE  55 20 47               .BYT 'U GEHE XX ZEILEN HOEHER UND '
1272  6DDA  4C 49 53               .BYT 'LISTE AUF',$D
1273  6DE4  56 20 54               .BYT 'V TRENNE ZEILE',$D
1274  6DF3  46 31 20               .BYT 'F1 NEU KALT START',$D
1275  6E05  46 32 20               .BYT 'F2 WIEDEREINTRITT',$D
1276  6E17  51 20 47               .BYT 'Q GEHE ZUR ERSTEN ZEILE UND '
1277  6E33  44 41 4E               .BYT 'DANN ZUM MONITOR',$D
1278  6E44  00                     .BYT 0
1279  6E45                  WAITTB
1280  6E45  02              TABLE  .BYT BOT,CUE,CDE,CRE,CLE,FZE,IOL
1281  6E4C  03                     .BYT C,XXX,$7F,REN,CEL
1282  6E51  14                     .BYT TA,LL2,LA2,FFF,UXX,DXX,WEX,VVV,OEX
1283  6E5A  F3 64           JTABLE .WOR CH,CU,CD,CR,CL,UPF1,UPF2,UPF3,UPF4
1284  6E6C  85 67                  .WOR DEL,RENTER,CLEARE,CA,LD2,LU2,FF,LU,LD,LEFT
1285  6E80  2F 68                  .WOR VTRLIN,TEXIT
1286  6E84                         *=$10C
1287  010C  4C 00 64               JMP BEGIN
1288  010F  4C E4 69               JMP START2
1289  0112                         .END
```

Martin Albrecht, 4350 Recklinghausen

Schneller Stringsuchen

mit dem Algorithmus von Boyer & Moore

Motivation:
Moderne Tischcomputer, insbesondere 16-Bit Modelle der PC-Klasse, werden zum überwiegenden Teil nur zur Textverarbeitung eingesetzt. Fast alle Textsysteme verfügen über die Möglichkeit, eine Zeichenkette im Text zu suchen. Vergleichende Tests einiger Textverarbeitungsprogramme für den IBM-PC, die die Zeitschrift BYTE zwischen Oktober 1984 und Januar 1985 veröffentlicht hat, zeigen ungewöhnlich lange Ausführungszeiten für die Suchfunkion. Es ergaben sich je nach Programm Suchzeiten zwischen 3,5 und 100 Sekunden bezogen auf einen 24 KByte/4000 Wort Standardtext (rund sechs Textseiten). Gesucht wurde jeweils das letzte Wort des Textes.

Überträgt man die Ergebnisse auf einen Text von 15 Seiten, wie er mit 8-Bit Computern (64 K RAM) speicherresident bearbeitet werden kann, dann betragen die Suchzeiten zwischen 8,5 Sekunden im günstigsten Fall und über 250 (!) Sekunden im schlechtesten Fall und führen so zu regelrechten Zwangspausen am Terminal.

Dafür verantwortlich ist neben Diskettenzugriffen auch der eigentliche Suchalgorithmus. Es besteht also ausreichend Grund, sich einmal mit Suchalgorithmen auseinanderzusetzen. Im folgenden Aufsatz wird der Standardalgorithmus dem schnelleren Boyer-Moore-Algorithmus gegenübergestellt.

Voraussetzungen:
Voraussetzungen für die Algorithmen sind: Ein Text und ein Suchstring endlicher Länge, wobei die Länge des Suchstrings kleiner gleich der Länge des Textes sein soll. Ferner sei ein endlicher Zeichenvorrat vorausgesetzt. Beide Voraussetzungen sind durch die Gegebenheiten eines realen Computers erfüllt. Der Zeichensatz besteht im allgemeinen aus maximal 256 verschiedenen Zeichen, entsprechend den 2 hoch 8 möglichen Kombinationen mit 8 Bit (1 Byte).

Ziel der Suchfunktion ist, das erste Auftreten eines Suchstings innerhalb eines Textes festzustellen und die Position des Suchstrings im Text zur Weiterverarbeitung zur Verfügung zu stellen. Enthalten ist auch der Fall, daß die gesuchte Sequenz im Text nicht vorkommt.

Der Standardalgorithmus:
Der Text und der Suchstring können als eindimensionale Felder definiert werden. I sei die Länge des Textes und J die Länge des Suchstrings:

```
I:          constant integer;
J:          constant integer;
TEXT:       array (0..I-1) of character;
SUCHSTRING: array (0..J-1) of character;   /* J <= I */;
```

Der Standardalgorithmus stellt sich dann in Pseudocode folgendermaßen dar:

```
K:=0;
L:=0;
while (K < J) and (L <= I-J) do
   K:=0;
   while (K < J) and SUCHSTRING(K) = TEXT (K+L) do K:=K+1 end;
   if K < J then L:=L+1 end;
end.
```

Suchsting und Text werden von links beginnend überlagert (L=0) und mit Hilfe der inneren Schleife zeichenweise verglichen. Stimmen zwei Zeichen nicht überein, dann wird die innere 'while'-Schleife verlassen, und der Suchstring um eine Position nach rechts verschoben (L:=L+1), mit dem Text überlagert und erneut verglichen.

Die äußere 'while'-Schleife wird ausgeführt, solange K < L ist (Suchstring nicht gefunden) und L die Maximalposition I-J nicht überschritten hat. Wenn der String gefunden wurde ist K=J und L die Startadresse in TEXT. Ist der Suchstring nicht vorhanden, dann ist K < J.

Dieses Verfahren ist das meistgebrauchte. Es ist kompakt und leicht zu programmieren.

Zeitverhalten:
Wie steht es um den Zeitbedarf dieses Algorithmus ? Sei N die Position des Suchstrings in TEXT, dann sind bis dahin mindestens N Zeichenvergleiche erforderlich, wenn beim Vergleich durch die innere Schleife gleich das erste Zeichen nicht übereinstimmt. Im schlechtesten Fall erhöht sich die Zahl der Tests jedoch bis auf N*J, wobei J die Länge des Suchstrings ist. Das passiert, wenn TEXT und SUCHSTRING sich beim Vergleich ab der Stelle L nur in einem Zeichen unterscheiden. Beispiel: TEXT='AAAAAAA...AAB' und SUCHSTRING='AAAB'. Im schlimmsten Fall ist das Verhalten also 'quadratisch', während jedoch bei sprachlichen Texten die Zahl der Vergleiche im allgemeinen in der Nähe von N liegt.

Der Boyer-Moore Algorithmus:
Robert S. Boyer und J. Strother Moore haben im Jahr 1977 einen schnelleren Suchalgorithmus vorgestellt /1/.
Wie läßt sich die Stringsuche beschleunigen? Auf die innere Vergleichsschleife kann nicht verzichtet werden. Um Gleichheit zwischen SUCHSTRING und TEXT festzustellen, ist der Vergleich aller J Zeichen des Suchstrings mit den korrespondierenden Zeichen des Textes erforderlich. So bleibt nur die Möglichkeit, die Testposition L in größeren Schritten nach rechts zu verschieben. Auf den ersten Blick ist nicht einsichtig, wie dies möglich ist.

Dazu zunächst einige Beispiele für mögliche Verschiebungen.

Diesmal beginnen wir den Vergleich von Text und Suchstring vom Ende des Suchstrings aus (Position 0) nach links fortschreitend. Sind Text und Suchstring an der verglichenen Stelle nicht identisch, dann werden wir beim Vergleich eine Position erreichen, wo die korrespondierenden Buchstaben nicht übereinstimmen. Dies ist die Schlüsselstelle für das weitere Vorgehen. Betrachten wir ein Beispiel:

```
LET UB CONSIDER A SIMPLE EXAMPLE          Text
     !                                    Schlüsselstelle
EXAMPLE                                   Suchstring
6543210
       EXAMPLE                            nächster Vergleich
```

Schlüsselstelle: 0, Buchstabe: ' ', Verschiebung: 7

Hier unterscheiden sich beide Strings gleich an der Position 0. Das Leerzeichen ' ' im Text stimmt nicht mit dem 'E' in 'EXAMPLE' überein. Suchstring und Text können deshalb nicht übereinstimmen. Darüberhinaus wissen wir, daß das Leerzeichen ' ' in 'EXAMPLE' nicht vorkommt. Wir können daher 'EXAMPLE' um seine volle Länge nach rechts verschieben, ohne eine Übereinstimmung zwischen Text und Suchstring zu verpassen:

```
LET UB CONSIDER A SIMPLE EXAMPLE          Text
         !                                Schlüsselstelle
       EXAMPLE                            Suchstring
       6543210
            EXAMPLE                       nächster Vergleich
```

Schlüsselstelle: 1, Buchstabe: 'D', Verschiebung: 6

Das 'D' aus 'CONSIDER' kommt in 'EXAMPLE' nicht vor. Wir können 'EXAMPLE' wieder (von der Schlüsselstelle aus) um seine ganze Länge nach rechts verschieben, so daß der nächste Vergleich erst hinter dem 'D' stattfindet:

```
LET UB CONSIDER A SIMPLE EXAMPLE          Text
              !                           Schlüsselstelle
            EXAMPLE                       Suchstring
            6543210
                  EXAMPLE                 nächster Vergleich
```

Schlüsselstelle: 0, Buchstabe: 'I', Verschiebung: 7

Das 'I' kommt in 'EXAMPLE' nicht vor, so daß wir unseren Suchstring widerum um seine volle Länge nach rechts verschieben können:

```
LET UB CONSIDER A SIMPLE EXAMPLE          Text
                    !                     Schlüsselstelle
                  EXAMPLE                 Suchstring
                  6543210
                      EXAMPLE             nächster Vergleich
```

Schlüsselstelle: 0, Buchstabe: 'X', Verschiebung: 5

Diesmal finden wir ein 'X'. Es stimmt nicht mit dem letzten Zeichen unseres Suchstrings überein, tritt jedoch im Suchstring auf. Das erste Auftreten von 'X' in 'EXAMPLE' ist fünf Zeichen vom rechten Ende entfernt. Wenn wir 'EXAMPLE' weniger als fünf Zeichen nach rechts verschieben, dann können wir keine Übereinstimmung erzielen. Denn das 'X' an der Schlüsselstelle müßte dabei mit einem 'X' in 'EXAMPLE' übereinstimmen, das rechts des 'X' in 'EXAMPLE' stände! Wir können also unseren Suchstring fünf Zeichen nach rechts verschieben, um das 'X' im Text mit dem 'X' in 'EXAMPLE' in Übereinstimmung zu bringen.

Beim nächsten Vergleich finden wir dann eine Übereinstimmung zwischen Text und Suchstring.

Das gezeigte Verfahren erfordert, daß wir für jeden Buchstaben des Textes, der nicht mit dem korrespondierenden Buchstaben des Suchstrings übereinstimmt, die maximale Verschiebung angeben können, bei der keine Übereinstimmung verpaßt wird. Wir wissen, daß wir für einen bestimmten Buchstaben in TEXT den Suchstring soweit verschieben können, daß das am weitesten rechts stehende Exemplar des Buchstabens im Suchstring mit dieser Stelle in Übereinstimmung gebracht wird. Tritt der Buchstabe in SUCHSTRING nicht auf, dann können wir ihn um seine ganze Länge verschieben, so als ob der Buchstabe ein Zeichen links des Suchstrings stände. Das erste Auftreten jedes Buchstaben unseres Alphabets in SUCHSTRING, von rechts zählend, wird vor dem eigentlichen Suchvorgang in einer Tabelle festgehalten.

Nehmen wir das Wort 'EXAMPLE' und als Alphabet die Großbuchstaben von 'A' bis 'Z' und das Leerzeichen, dann ergibt sich folgendes Bild:

```
E  X  A  M  P  L  E     Buchstaben
6  5  4  3  2  1  0     Position
```

daraus folgt die Buchstabentabelle:

```
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z ' '   Buchstabe
4 7 7 7 0 7 7 7 7 7 7 1 3 7 7 2 7 7 7 7 7 7 7 7 5 7 7  7   Position
```

Für alle Buchstaben, die nicht in 'EXAMPLE' vorkommen, ist die Position gleich der Länge des Wortes, so als ob sie an der Position 7 ständen.

Diese Tabelle gibt uns für jeden Buchstaben die Verschiebedistanz für die Schlüsselstelle 0 an. Liegt die Schlüsselstelle weiter links, so kann nur entsprechend weniger verschoben werden. Die Verschiebedistanz errechnet sich also aus:

Verschiebedistanz := Tabellenwert(Buchstabe) - Schlüsselstelle

Dies gilt jedoch nur, wenn wir an der Schlüsselstelle einen Buchstaben finden, der erst links davon in unserem Suchstring auftritt. Betrachten wir folgendes Beispiel:

```
...IT IS ELEMENTARY                    Text
       !                               Schlüsselstelle
     EXAMPLE                           Buchstring
     6543210
   EXAMPLE                             nächster Vergleich ?
```

Schlüsselstelle: 2, Buchstabe: 'E', Tabellenwert: 0, Verschiebung: -2 ?

Hier müßte nach unserer Rechnung 'EXAMPLE' zwei Stellen zurück geschoben werden, da wir das rechteste 'E' in 'EXAMPLE' bereits überschritten haben. Stattdessen verschieben wir unseren Suchstring in diesem Fall eine Position weiter nach rechts, wie beim Standardalgorithmus. Wir erhalten für die Verschiebedistanz:

Verschiebedistanz := Tabellenwert(Buchstabe) - Schlüsselstelle
Wenn Verschiebedistanz <= 0 dann Verschiebedistanz := 1

Zeitverhalten:
Betrachten wir unser Beispiel: 'EXAMPLE' wird an der Position 25 gefunden. Mit dem Standardalgorithmus wären minimal 25+Länge(Suchstring)

Zeichenvergleiche erforderlich. Bei dem neuen Algorithmus kommen wir mit 5+Länge(Suchstring) Vergleichen aus.
Der gezeigte Algorithmus hat im schlechtesten Fall ebenfalls ein quadratisches Verhalten. Im Durchschnitt sind jedoch weniger als N Vergleiche erforderlich, wenn N die Position des Suchstrings im Text ist. Das Verfahren verbessert sein Verhalten, je länger der Suchstring ist. Er kann dann über größere Strecken verschoben werden. Das Verhalten verschlechtert sich, wenn das benutzte Alphabet kleiner wird. Für ein binäres Alphabet besteht kein Vorteil gegenüber dem Standardalgorithmus.
Natürlich muß auch die Zeit berücksichtigt werden, die zur Erstellung der Verschiebetabelle erforderlich ist.

Formulieren wir den Algorithmus in Pseudocode:

```
I:          constant integer;
J:          constant integer;
K,L,DIST:   integer;
X:          character;
TEXT:       array (0..I-1) of character;
SUCHSTRING: array (0..J-1) of character; /* J <= I */;
ALPHABET:   array ('A'..'Z',' ') of integer;

/* Anlegen der Verschiebetabelle */;
/* fülle ALPHABET mit Länge (SUCHSTRING) */;
for X:='A' to ' ' do ALPHABET(X):=J end;

/* trage Verschiebedistanz ein */;
L:=J-1;
for K:=0 to J-1 do
   ALPHABET(SUCHSTRING(K)):=L;
   L:=L-1;
end;

/* Suchvorgang */;
L:=0;
while (K >= 0) and (L <= I-J) do
   K:=J-1;
   while (K >=0) and SUCHSTRING(K) = TEXT(L+K) do K:=K-1 end;
   if K >= 0 then do
      DIST:=ALPHABET(TEXT(L+K)) - (J-1-K);
      if DIST <= 0 then DIST:=1;
      L:=L+DIST;
   end;
end.
```

Die Verschiebetabelle wird zunächst mit der Länge des Suchstrings gefüllt. Danach wird der Suchstring von links nach rechts durchlaufen, und für jeden Buchstaben die Position (L) eingetragen. Tritt ein Buchstabe weiter rechts ein zweites mal auf, dann überschreibt der neue Wert den alten Eintrag.
Der eigentliche Suchvorgang unterscheidet sich nicht wesentlich vom Standardalgorithmus. Der Vergleich beginnt vom Ende des Suchstrings und die Testposition L wird jetzt mit Hilfe der Tabelle weitergeschaltet.

Der gezeigte Algorithmus entspricht nicht genau dem ursprünglichen Boyer-Moore Algorithmus /1/. Er ist jedoch kompakter und kann, da er kaum länger ist, den Standardalgorithmus leicht ersetzen. Für eine ausführlichere Darstellung und den Beweis der Richtigkeit sei auf /2/ verwiesen.

Praktischer Vergleich zweier Maschinenprogramme:
Um die Effektivität des schnelleren Algorithmus zu testen, wurde ein entsprechendes Maschinenprogramm auf dem CMOS 65SC02 erstellt (siehe Listing). Als Alphabet dienen die 128 ASCII-Zeichen von $00 bis $7F. Die Erweiterung auf 256 Zeichen ist jedoch leicht möglich.

Zum Vergleich treten die Findfunktion des AIM-65 Texteditors und das gelistete Programm gegeneinander an. Die Tabelle zeigt die Suchzeiten für deutschen Text. Die Zeiten wurden für 40 Durchläufe eines 10 KByte langen Textes (400 KByte) ermittelt und auf 24 KByte umgerechnet. Der Systemtakt des 6502 beträgt 1 MHz. Gesucht wurde jeweils das letzte Wort.

Länge (KByte, Zeichen) Text	Suchstring	Suchzeiten (Sekunden) AIM-65	Boyer-Moore	Boyer&Moore X mal schneller
24 KByte	1 Zeichen	1,22 sec	1,34 sec	0,9
24 KByte	4 Zeichen	1,36 sec	0,37 sec	3,7
24 KByte	19 Zeichen	1,22 sec	0,11 sec	11.1
24 KByte	26 Zeichen	-	0,08 sec	-
24 KByte	46 Zeichen	-	0,07 sec	-

Hier tritt der deutliche Vorteil des Boyer-Moore Algorithmus zu Tage. Schon bei normalen Wortlängen (4 Zeichen) ist er rund viermal schneller. Bei längeren Suchstrings wird der Unterschied noch wesentlich deutlicher. Die Suchzeiten bleiben im vertretbaren Rahmen. Mit schnelleren 16-Bit Prozessoren sind sicher noch erheblich bessere Zeiten möglich.

Literatur:

/1/ R.Boyer, J.S.Moore,
A fast String Searching Algorithm
Commun. Assoc. Comput. Mach. 20(10), 762-772, (1977)

/2/ R.Boyer, J.S.Moore,
The Correctness of a Fast String Searching Algorithm
A Computational Logic, 283-308,
Academic Press Inc., (1979)

```
------------------------------------------------------------
Suchen mit dem Algorithmus von Boyer & Moore

0000          ;für CMOS-CPU 65SC02 oder 65C02
0000          ;Parameter bei Aufruf:
0000          ;Textbeginn: TPTR, Textende: TBOT
0000          ;Suchstring: SPTR, Suchstringlänge: SLEN
0000          ;sucht nur ASCII-Zeichen zwischen $00 und $7F
0000          ;Suchstringlänge maximal 128 Zeichen
0000
0000          ;Aufruf mit gesetzten Parametern: JSR SUCH
0000          ;Return: Carry=1: String gefunden, TPTR = Adresse
0000          ;        Carry=0: nicht gefunden,  TPTR > END
0000
------------------------------------------------------------
Variablen

0000 TPTR           =$DF            ;Textpointer, bei Aufruf auf Beginn
0000 TBOT           =$E1            ;Textende
0000 END            =$E7            ;Suchende
0000 SLEN           =$E9            ;Suchstringlänge
0000 SPTR           =$200           ;Suchstring
0000
0000 ALPHAB         =$300           ;Alphabet-Tabelle 128 Zeichen $0 - $7F
0000                                ;Buchstabe ist Index in ALPHAB
0000
0000                *=$400
0400
------------------------------------------------------------
```

Längeneintrag in ALPHAB

```
0400          ;erzeuge Tabelle ALPHAB mit 128 Einträgen der Stringlänge
0400 SUCH     A5E9    LDA SLEN          ;Suchstringlänge
0402          A000    LDY #0
0404 SUCH1    990003  STA ALPHAB,Y      ;trage Länge in ALPHAB ein
0407          C8      INY
0408          10FA    BPL SUCH1         ;bis Y=$7F
040A
-------------------------------------------------------------
Verschiebedistanz in ALPHAB

040A          ;trage Buchstabenposition in ALPHAB ein
040A          A200    LDX #0            ;beginne bei Suchstring,0
040C          A5E9    LDA SLEN
040E          3A      DEC A             ;Eintrag: von Stringlänge-1 bis 0
040F SUCH2    BC0002  LDY SPTR,X        ;Buchstabe ist Index in ALPHAB
0412          990003  STA ALPHAB,Y
0415          E8      INX
0416          3A      DEC A
0417          10F6    BPL SUCH2         ;bis Suchstringende
0419
-------------------------------------------------------------
berechne Suchende

0419          ;Suchende = Textlänge - Suchstringlänge (in END)
0419          A5E1    LDA TBOT
041B          38      SEC
041C          E5E9    SBC SLEN          ;subtrahiere Suchstringlänge
041E          85E7    STA END           ;Suchende in END
0420          A5E2    LDA TBOT+1
0422          E900    SBC #0
0424          85E8    STA END+1
0426
-------------------------------------------------------------
Suche Suchstring im Bereich TPTR bis END

0426          ;String gefunden: Carry=1, TPTR = Adresse
0426          ;nicht gefunden:  Carry=0, TPTR >= END
0426 SUCH3    A4E9    LDY SLEN          ;beginne Vergleich vom Stringende
0428          88      DEY
0429          A200    LDX #0            ;Schlüsselposition
042B SUCH4    B1DF    LDA (TPTR),Y      ;Textzeichen
042D          D90002  CMP SPTR,Y        ;vergleiche mit Suchstring
0430          D005    BNE SUCH5         ;ungleich
0432          E8      INX               ;Schlüsselposition + 1
0433          88      DEY               ;ein Zeichen weiter links
0434          10F5    BPL SUCH4         ;gefunden, wenn Y < 0
0436          60      RTS               ;gefunden, Carry=1

0437          ;Suchstring nicht gefunden, neue Position berechnen
0437 SUCH5    A8      TAY               ;Buchstabe ist Index in ALPHAB
0438          8A      TXA               ;Schlüsselstelle
0439          49FF    EOR #$FF          ;Komplement
043B          38      SEC               ;2er Komplement
043C          790003  ADC ALPHAB,Y      ;Verschiebedistanz - Schlüsselstelle
043F          F002    BEQ SUCH6         ;Distanz = 0 --> Distanz:=1
0441          1002    BPL SUCH7         ;Distanz > 0
0443 SUCH6    A901    LDA #1
0445 SUCH7    18      CLC
0446          65DF    ADC TPTR          ;addiere Distanz zu Textpointer TPTR
0448          85DF    STA TPTR
044A          9002    BCC SUCH8         ;kein übertrag
044C          E6E0    INC TPTR+1
044E
044E          ;teste ob TPTR <= END. Wenn ja, dann weiter
044E SUCH8    A5E0    LDA TPTR+1
0450          C5E8    CMP END+1
0452          90D2    BCC SUCH3         ;TPTR < END
0454          D008    BNE SUCH9         ;TPTR > END
0456          A5DF    LDA TPTR
0458          C5E7    CMP END
045A          90CA    BCC SUCH3         ;TPTR < END
045C          F0C8    BEQ SUCH3         ;TPTR = END
045E SUCH9    18      CLC               ;nicht gefunden, Carry=0
045F          60      RTS
0460
0460
0460                  .END
ERRORS=0000
```

Andreas Zilker, 8300 Landshut

DOS für AIM 65

1. Einleitung

Über die Vorteile eines Floppy-Disk-Laufwerkes bei der Arbeit mit einem Computer wurde in dieser Zeitschrift bereits ausführlich referiert. Hier soll nun eine weitere Möglichkeit dargestellt werden, den AIM 65 mit einem Floppy-Laufwerk auszustatten, wobei das Augenmerk weniger auf die physikalische Anpassung, sondern mehr auf die logische Organisation gerichtet ist. Das nachstehende Programm stellt indes keine fertige 'Abtipp-Lösung' dar, vielmehr soll es ein leicht modifizierbares Grundkonzept zur Diskverwaltung bieten. Es wurde für einen 'intelligenten' Floppy-Disk-Controller (FDC) der Firma Siemens geschrieben. Die Anpassung an andere FDC z.B. aus MICRO MAG (1) oder MC (2) sollte auf Grund der dort dargestellten Treiber-Software und des modularen DOS-Aufbaues leicht möglich sein. Die Anbindung an den AIM-Monitor erfolgt über die User-Vektoren, wodurch Kompatibilität mit den Sprachen und Hilfsprogrammen gewährleistet ist.

2. Organisation der Disk

Um Dateien anzulegen, ist eine Aufteilung der Disk in logische Einheiten nötig, unabhängig vom physikalischen Format. Diese Einheiten werden im folgenden Blöcke genannt, zur Unterscheidung von Tracks und Sektoren. Die Verkettung der Blöcke zu Dateien kann nach zwei verschiedenen Prinzipien erfolgen:

Linked List: Jeder Block enthält am Ende einen Verweis auf den nachfolgenden innerhalb der Datei. Diese Methode ist einfach zu realisieren, direkter Zugriff auf einzelne Blöcke ist jedoch nicht möglich. Bei mehrfachem Löschen und Neueintragen von Dateien entsteht ziemlich bald ein unübersichtliches Gewirr, wodurch spezielle 'compress'-Funktionen notwendig werden (3).

Bitmap: Die aktuelle Diskbelegung wird auf eine Bit-Tabelle abgebildet. Die einzelnen Blöcke sind durchnumeriert; ein belegter Block wird durch ein besetztes Bit markiert. Die Blöcke werden entsprechend den Platznummern an die Dateien vergeben. Dieses Konzept wird auch in dem weit verbreiteten Betriebssystem CP/M der 80er-Prozessorfamilie verwendet (4).

2.1 Directory

Das Inhaltsverzeichnis der Disk befindet sich meist auf den äußeren Spuren, schon wegen der Datensicherheit. Hier im konkreten Fall steht es auf Track 0. Eine Verschiebung zu Gunsten von Systemspuren (für Bottstrap-Lader) ist leicht möglich, da es voll in die Bitmap eingegliedert ist. Bei Dateizugriffen wird die Directory, die in einzelne File-Control-Blöcke (FCB) eingeteilt ist, komplett ins RAM geladen. Dieses ermöglicht schnellen Zugriff und störsicheren Betrieb.

Organisation der Directory:

```
<<----------------  1. FCB  ----------------->> <<-------------  2. FCB ---
-------------------------------------------------------------------------------
!   Status  !   Dateiname  !  Blocknummern ...  !  Status  !  Dateiname  !  ...
-------------------------------------------------------------------------------
Bytes:  1         NAMLEN       FCBLEN - (NAMLEN+1)      1         ...

---  n-ter FCB  ----->>
-----------------------------------------------------
!  ... FCBNUM-mal ...  !            Bitmap           !
-----------------------------------------------------


Statusbyte:     Bit 7 = 1 --> FCB belegt
                Bit 6 = 1 --> Datei gegen überschreiben geschützt
                Bit 5 bis Bit 0 frei für Erweiterungen
```

Dateiname: sieben Zeichen (bei Bedarf zu ändern)
alle druckbaren ASCII-Zeichen erlaubt

Blocknummern: Hier werden vom DOS die Nummern der, für diese Datei zugewiesenen, Blöcke eingetragen. Da nur ein Byte pro Block benutzt wird, beträgt die maximale Blockanzahl pro Disk 255 ($0 ... $FE). Als End-of-File (EOF) Marke wird $FF verwendet.

2.2. Anpassung an andere Disk-Formate

Die im folgenden verwendeten Variablennamen beziehen sich auf die im Listing gebrauchten Symbole.

Vorgabe (je nach Disk - Format und gewünschter Dateigröße) :

```
DISK = 80 kByte, BLOCK = 0.5 kByte, DIRSIZ = 2048 Byte
FCBLEN = 64 Byte, NAMLEN = 7 Byte
Randbedingung: DISK / BLOCK =< 255 ( siehe 2.1. )
Bei doppelseitigen Laufwerken eventuell in zwei
"logische" Drives aufteilen.
```

Resultierende Größen:

```
BLKNUM = FCBLEN - ( NAMLEN + 1 )              = 56
max. Dateigröße = BLKNUM * BLOCK              = 28 kByte
MAPLEN = DISK / ( BLOCK * 8 )                 = 20 Byte
FCBNUM = INT(( DIRSIZ - MAPLEN ) / FCBLEN )   = 31
```

Weitere format- und hardware-spezifischen Konstanten und Programmteile sind im Listing mit " **** " gekennzeichnet und ggfs. zu ändern.

2.3 Deblocking

Je nach Formatierung der Diskette stimmen Block- und Sektorgröße nicht immer überein. Das portionsgerechte Aufbereiten der Daten für den FDC läuft in der Literatur unter dem Namen 'Deblocking'. Beim AIM kommen die Daten byte-weise über die User-Vektoren und werden zunächst in einem RAM-Puffer zwischengespeichert. Wenn dieser überläuft, werden vom DOS Track und Sektor des nächsten Blockes berechnet und an die Treiberroutinen des FDC übergeben (Datails siehe Listing).

3. Bedienung des DOS

Mit der Taste (F1) wird der FDC initialisiert und mit Hilfe des MONLINK-Vektors werden fünf neue Kommandos eingebunden (5). Damit über die User-Vektoren nicht jedes Byte in zwei ASCII-Zeichen umgewandelt wird, sind folgendes 'Patches' im AIM-Monitor empfehlenswert (6):

Adresse	vorher	nachher	
$E3FD	AD 12 A4	4C 93 E9	JMP INALL
$E548	4C 46 EA	4C BC E9	JMP OUTALL

Das DOS funktioniert auch ohne diese Änderungen. Der Platz- und Zeitbedarf bei Disk-Aktivitäten verdoppelt sich jedoch.

3.1 Lesen einer Datei

In der jeweiligen Kommando-Ebene (Monitor, Editor usw.) wird auf 'IN=' mit 'U' (User) geantwortet. Das DOS liest nun die Directory und erwartet nach 'F=' den Dateinamen. Falls man weniger als NAMLEN Zeichen eingibt (vorzeitiges CR), wird der Rest mit Blanks aufgefüllt. Existiert die angegebene Datei auf der Diskette nicht, erfolgt die Fehlermeldung 'Keine Datei'. Nach erfolgreichem Abschluß landet man wieder in der Kommandoebene.

3.2 Schreiben einer Datei

Die Kommandoeingabe erfolgt analog zu 3.1. Eine bereits vorhandene Datei gleichen Namens wird überschrieben (Ausnahme siehe 3.5). Ist vollständiges Abspeichern der Datei nicht möglich (kein Platz auf der Diskette oder im FCB), wird 'Disk Voll' ausgegeben, und man landet im Monitor. Mit einer neuen Diskette bzw. Aufteilung der Datei kann man es dann nochmal versuchen. Das leidige Problem des fehlenden EOF-Zeichens beim AIM wurde durch ein eigenes Kommando gelöst (siehe 3.3 und (7)).

3.3 Datei schließen (CNTL-U, update disk)

Dieses Kommando ist nur vom Monitor aus verfügbar. Bei Benutzung des Editors oder einer Sprache ist dies zu berücksichtigen. Der letzte Block aus dem RAM-Puffer und die Direktory werden weggeschrieben. Damit ist die Datei auf der Diskette fixiert. **Vorsicht:** Falls dies vor einem erneuten DOS-Komando nicht geschieht, sind Dateiverluste und entsprechend markige Äußerungen des Benutzers nicht auszuschließen. Die Verwendung des Control-Codes dient als Schutz gegen Vertippen.

3.4 Directory anzeigen ((F) -- files)

Die Dateinamen werden ausgegeben, und zwar mit Angabe der Dateigröße (auf volle kByte gerundet). Das Ausgabeformat ist anpaßbar (siehe Listing). Am Ende wird der noch verbleibende Speicherplatz angezeigt (auf 0,5 kByte genau).

3.5 Schreibschutz ((W) -- writeprotect)

Der Dateiname wird mit 'F=' abgefragt und entsprechend Bit 6 im Statusrebyte gesetzt. Soll nun nach 3.2 eine Datei gleichen Namens geschrieben werden, erfolgt die Nachfrage 'Überschreiben J/N'. Bei 'N' landet man im Monitor.

3.6 Datei löschen ((C) -- clear file)

Nach Eingabe des Dateinamens wird Bit 7 im Statusbyte rückgesetzt und die Blocknummern im FCB mit $FF überschrieben. Ein eventueller Schreibschutz wird nicht berücksichtigt.

3.7 Formatieren ((O) -- originate disk)

Um ungewolltes 'Putzen' einer Diskette zu vermeiden, wird mit 'Formatieren J/N' nachgefragt. Bei den 17xx-Controllern wäre aus Platzgründen auch an ein eigenes Dienstprogramm zu denken, wobei das Initialisieren der Directory aus dem DOS zu übernehmen ist.

4. Betriebserfahrungen

Das vorgestellte DOS läuft seit fast einem Jahr auf zwei AIM's problemlos. Es arbeitet mit allen vorhandenen Sprachen (Forth, GWK- und Microsoft-BASIC, PL/65) einwandfrei zusammen. Mit der in (7) vorgeschlagenen Änderung funktioniert auch das Assemblieren von Disk. Die Hardware-Anpassung ans DOS ist durch vier definierte Schnittstellen, hoffentlich, leicht möglich.

5. Mögliche Erweiterungen

Falls mehrere Laufwerke verwendet werden, kann man in NAMINP bzw. NAMSRC umschalten. Wem die User-Vektoren zu langsam sind, der kann auch per 'DMA' direkt auf den Blockpuffer zugreifen. Bei sonstigen Geschwindigkeitsproblemen (double density mit 1 MHz-CPU o.ä.) sei auf die interessanten Ideen in (8) hingewiesen. Die freien Bits im Statusbyte könnten zur Verkettung von FCB's zu größeren Dateien verwendet werden. Eventuelle Hardware-FEhler sollte man im FDC-Basistreiber anzeigen.

Literatur:

(1) M. Zimmermann Mini Floppy Interface für AIM Micro Mag 33/34
(2) A. Pausch Mini-Floppy-Interface mc 11,12/82
(3) DAIM Manual Compas Microsystems 1980
(4) A. Johnson-Laird The Programmer's CP/M Handbook McGraw-Hill
(5) Redder/Wollenberg REMON Micro Mag 28
(6) F. Agerer Landshut persönl. Mitteilung
(7) P. Rix AIM-Betriebssystem für CBM-Floppy Micro Mag 33
(8) T. Netsch Zeit sparen mit 6502 c't 11/84

```
000000                  0000 ; ****************************
000000                  0001 ; *                          *
000000                  0002 ; *      DOS für AIM 65      *
000000                  0003 ; *                          *
000000                  0004 ; ****************************
000000                  0005
000000                  0006 ; Copyright
000000                  0007 ; Andreas Zilker
000000                  0008 ; Schöplergasse 20b
000000                  0009 ; 8300 Landshut
000000                  0010
000000                  0011 ; Version 2.3   84-11
000000                  0012
000000                  0013 ; Verwendetes Diskettenformat:
000000                  0014 ; 5 1/4 Zoll, einseitig, einfache Dichte,
000000                  0015 ; 40 Tracks, 16 Sektoren mit 128 Byte
000000                  0016 ; --> 80 kByte pro Diskette
000000                  0017
000000                  0018 ; $$$$$$$  SYMBOLE  $$$$$$$
000000                  0019
000000                  0020 ; KONSTANTEN
000000                  0021 DISK   =80             ; Diskettenkapazität in kByte
000000                  0022 BLOCK  =512            ; Blockgröße in Byte
000000                  0023 FCBNUM =31             ; Anzahl der Directory-Einträg
e
000000                  0024 FCBLEN =64             ; Länge eines FCB
000000                  0025 NAMLEN = 7             ; Länge eines Dateinamens
000000                  0026 MAPLEN =20             ; Länge der Bitmap in Byte
000000                  0027 NPLIN  = 3             ; Anzahl der Dateinamen pro Ze
ile
000000                  0028                        ; bei der Directory-Ausgabe
000000                  0029 BLKEMP =$FF            ; Markierung für leeren Block
000000                  0030 CR     =$0D            ; ASCII <CR>
000000                  0031
000000                  0032 ; ADRESSEN
000000                  0033
000000                  0034 BLKBUF =$8600          ; Puffer für Deblocking *****
000000                  0035 DIRBUF =$8800          ; Puffer für Directory  *****
000000                  0036 BITMAP =DIRBUF+2016 ( ... + FCBNUM * FCBLEN ) *****
000000                  0037 FNAME  =BITMAP+MAPLEN  ; Puffer für Dateinamen
000000                  0038 SAVA   =FNAME+NAMLEN   ; temp. Accu
000000                  0039 NAMCNT =SAVA+1         ; Zähler und Zwischenspeicher
000000                  0040 BLKSUM =NAMCNT+1       ; Rechenregister für Directory
-Ausgabe
000000                  0041 BLKCNT =BLKSUM+1       ; Blockzähler für Deblocking
000000                  0042
000000                  0043 ; POINTER
000000                  0044
000000                  0045 DPNT   =$FC            ; Directory
000000                  0046 BLKP   =DPNT+2         ; Deblocking
000000                  0047
000000                  0048 ; MONITOR
000000                  0049
000000                  0050 UIN    =$108           ; User-Vektoren
000000                  0051 UOUT   =$10A
000000                  0052 MONLIN =$A40A          ; Command-Vektor ( REMON )
000000                  0053 RDRUB  =$E95F
000000                  0054 OM     =$E7D4
000000                  0055 OUTPUT =$E97A
000000                  0056 REDOUT =$E973
000000                  0057 COMIN  =$E1A1
000000                  0058 NUMA   =$EA46
000000                  0059
000000                  0060         *=$10C
00010C                  0061
00010C  4C 00 80        0062         JMP DOSINI     ; <F1>-Taste
00010F                  0063
00010F                  0064         *=$8000
008000                  0065
008000                  0066 ; DOS INITIALISIEREN
008000                  0067
```

```
008000  20 B7 83     0068 DOSINI JSR OPEN          ; Floppy-Controller initial.
008003  A9 22        0069        LDA #<AIMDOS      ; Monitor-Kommandos erweitern
008005  8D 0A A4     0070        STA MONLIN
008008  A9 22        0071        LDA #>AIMDOS
00800A  8D 0B A4     0072        STA MONLIN+1
00800D  A9 7C        0073        LDA #<RDFILE      ; User-Vektoren setzen
00800F  8D 08 01     0074        STA UIN
008012  A9 7C        0075        LDA #>RDFILE
008014  8D 09 01     0076        STA UIN+1
008017  A9 C6        0077        LDA #<WRFILE
008019  8D 0A 01     0078        STA UOUT
00801C  A9 C6        0079        LDA #>WRFILE
00801E  8D 0B 01     0080        STA UOUT+1
008021  60           0081        RTS
008022               0082
008022               0083 ; EINBINDUNG DER FLOPPY-KOMMANDOS IN MONITOR
008022               0084
008022  C9 4F        0085 AIMDOS CMP #'O'          ; "Originate disk"
008024  D0 03        0086        BNE *+5
008026  4C 48 B0     0087        JMP DISINI
008029  C9 57        0088        CMP #'W'          ; "Writeprotect"
00802B  D0 03        0089        BNE *+5
00802D  4C 88 B0     0090        JMP WRTPRO
008030  C9 43        0091        CMP #'C'          ; "Clear file"
008032  D0 03        0092        BNE *+5
008034  4C A2 B0     0093        JMP ERAFIL
008037  C9 46        0094        CMP #'F'          ; "Show files"
008039  D0 03        0095        BNE *+5
00803B  4C BF B0     0096        JMP PRIDIR
00803E  C9 15        0097        CMP #$15          ; CNTL-U "Update disk"
008040  D0 03        0098        BNE *+5
008042  4C 6D B1     0099        JMP CLOSFI
008045  4C D4 E7     0100        JMP QM            ; kein gültiges Kommando
008048               0101
008048               0102 ; DISKETTE INITIALISIEREN
008048               0103
008048  A0 00        0104 DISINI LDY #0
00804A  B9 BD 83     0105 DILP0  LDA FORMSG,Y
00804D  20 7A E9     0106        JSR OUTPUT        ; Rückversicherung gegen
008050  C8          0107        INY               ; Bedienungsfehler
008051  C0 11        0108        CPY #17
008053  D0 F5        0109        BNE DILP0
008055  20 73 E9     0110        JSR REDOUT
008058  C9 4A        0111        CMP #'J'
00805A  F0 03        0112        BEQ DISI1
00805C  4C A1 E1     0113        JMP COMIN
00805F               0114
00805F  20 B8 83     0115 DISI1  JSR FORM1         ; Diskette formatieren
008062  20 51 82     0116        JSR SETDP
008065  A0 00        0117 DILP1  LDY #0            ; Directory initialisieren
008067  98           0118        TYA
008068  91 FC        0119        STA (DPNT),Y      ; Statusbyte löschen
00806A  20 E2 82     0120        JSR CLBLK         ; Blockeinträge löschen
00806D  20 5C 82     0121        JSR ADVDP         ; Pointer inkrementieren
008070  CA           0122        DEX               ; X-Reg. zählt FCB's
008071  D0 F2        0123        BNE DILP1
008073  A0 00        0124        LDY #0
008075  A9 F0        0125        LDA #%11110000    ; (DIRP) zeigt auf 1. Byte der
 Bitmap
008077  99 E0 BF     0126        STA BITMAP,Y      ; Track 0 für Directory in
00807A               0127                          ; der Bitmap sperren
00807A  98           0128        TYA
00807B  C8           0129        INY
00807C  99 E0 BF     0130 DILP2  STA BITMAP,Y      ; Rest der Bitmap löschen
00807F  C8           0131        INY
008080  C0 14        0132        CPY #MAPLEN
008082  D0 F8        0133        BNE DILP2
008084  20 BC 83     0134        JSR WRDIR         ; neue Directory auf Disk schr
eiben
008087  60           0135        RTS
008088               0136
```

```
0080B8                  0137 ; DATEI SCHREIBSCHÜTZEN
0080B8                  0138
0080B8  20 BB B3        0139 WRTPRO JSR RDDIR        ; Directory einlesen
0080BB  20 6B B2        0140        JSR NAMINP       ; Namen eingeben
0080BE  20 99 B2        0141        JSR NAMSRC       ; und suchen
0080C1  90 03           0142        BCC *+5
0080C3  4C BB B2        0143        JMP NOTFND       ; C-Flag = 1 --> nicht gefunde
n
0080C6  A0 00           0144        LDY #0
0080C8  B1 FC           0145        LDA (DPNT),Y
0080CA  09 40           0146        ORA #$40         ; write-protect flag setzen
0080CC  91 FC           0147        STA (DPNT),Y
0080CE  20 BC B3        0148        JSR WRDIR        ; Directory wieder wegschreibe
n
0080A1  60              0149        RTS
0080A2                  0150
0080A2                  0151 ; DATEI LÖSCHEN
0080A2                  0152
0080A2  20 BB B3        0153 ERAFIL JSR RDDIR        ; siehe WRTPRO
0080A5  20 6B B2        0154        JSR NAMINP
0080A8  20 99 B2        0155        JSR NAMSRC
0080AB  90 03           0156        BCC *+5
0080AD  4C BB B2        0157        JMP NOTFND
0080B0  A0 00           0158        LDY #0
0080B2  98              0159        TYA
0080B3  91 FC           0160        STA (DPNT),Y     ; Status Byte löschen
0080B5  20 3A B3        0161        JSR BLKEX        ; Bitmap auf neuen Stand bring
en
0080B8  20 E2 B2        0162        JSR CLBLK        ; Blockeinträge löschen
0080BB  20 BC B3        0163        JSR WRDIR
0080BE  60              0164        RTS
0080BF                  0165
0080BF                  0166 ; DIRECTORY ANZEIGEN MIT JEWEILIGER DATEIGRÖSSE UND
0080BF                  0167 ; VERBLEIBENDEM SPEICHERPLATZ
0080BF                  0168
0080BF                  0169 ; ***** Bei anderen Diskformaten Rechnung *****
0080BF                  0170 ; ***** und Dezimalwandlung evtl. ändern  *****
0080BF                  0171
0080BF  20 BB B3        0172 PRIDIR JSR RDDIR
0080C2  A9 0D           0173        LDA #CR
0080C4  20 7A E9        0174        JSR OUTPUT       ; neue Zeile
0080C7  A9 00           0175        LDA #0
0080C9  8D FD BF        0176        STA BLKSUM       ; Rechenregister löschen
0080CC  20 51 B2        0177        JSR SETDP        ; Pointer auf Directory-Anfang
0080CF  A9 03           0178        LDA #NPLIN
0080D1  8D FC BF        0179        STA NAMCNT       ; Zähler für Ausgabeformatieru
ng
0080D4                  0180
0080D4  A0 00           0181 PRILP1 LDY #0
0080D6  B1 FC           0182        LDA (DPNT),Y
0080D8  29 80           0183        AND #$80         ; belegten FCB suchen
0080DA  F0 52           0184        BEQ NXTONE
0080DC  C8              0185 PRILP2 INY
0080DD  B1 FC           0186        LDA (DPNT),Y     ; Namen ausgeben
0080DF  20 7A E9        0187        JSR OUTPUT
0080E2  C0 07           0188        CPY #NAMLEN
0080E4  D0 F6           0189        BNE PRILP2
0080E6  A9 20           0190        LDA #' '         ; Zwischenraum
0080E8  20 7A E9        0191        JSR OUTPUT
0080EB  C8              0192        INY
0080EC  B1 FC           0193 FILSIZ LDA (DPNT),Y     ; gültige Blockeinträge zählen
0080EE  C9 FF           0194        CMP #BLKEMP
0080F0  F0 05           0195        BEQ FILS2        ; ab dann alles leer !
0080F2  C8              0196        INY
0080F3  C0 40           0197        CPY #FCBLEN      ; FCB komplett ?
0080F5  D0 F5           0198        BNE FILSIZ
0080F7  98              0199 FILS2  TYA
0080F8  38              0200        SEC              ; effekt. Blockanzahl eines
0080F9  E9 08           0201        SBC #NAMLEN+1    ; FCB berechnen
0080FB  48              0202        PHA
0080FC  18              0203        CLC
```

```
0080FD  6D FD BF      0204          ADC BLKSUM        ; zur Gesamtzahl addieren
008100  8D FD BF      0205          STA BLKSUM
008103  68            0206          PLA
008104  4A            0207          LSR A             ; Blockanzahl / 2 -> Dateigröß
e in kByte
008105  AB            0208          TAY
008106  90 01         0209          BCC *+3
008108  C8            0210          INY               ; Rest bei Division 1/2 k aufr
unden
008109  20 9E B3      0211          JSR BBTOD         ; Dezimalwandlung
00810C  20 46 EA      0212          JSR NUMA          ; und ASCII Ausgabe
00810F  A9 4B         0213          LDA #'K'
008111  20 7A E9      0214          JSR OUTPUT
008114  A9 20         0215          LDA #' '          ; für die "Optik" !
008116  20 7A E9      0216          JSR OUTPUT
008119  20 7A E9      0217          JSR OUTPUT
00811C  20 7A E9      0218          JSR OUTPUT
00811F  CE FC BF      0219          DEC NAMCNT        ; Zeile komplett ?
008122  D0 0A         0220          BNE NXTONE
008124  A9 03         0221          LDA #NPLIN        ; Zähler neu setzen
008126  8D FC BF      0222          STA NAMCNT
008129  A9 0D         0223          LDA #CR           ; neue Zeile
00812B  20 7A E9      0224          JSR OUTPUT
00812E                0225
00812E  20 5C B2      0226 NXTONE   JSR ADVDP         ; nächster FCB
008131  CA            0227          DEX
008132  D0 A0         0228          BNE PRILP1
008134  A9 0D         0229          LDA #CR
008136  20 7A E9      0230          JSR OUTPUT        ; 2 Leerzeilen
008139  20 7A E9      0231          JSR OUTPUT
00813C  4E FD BF      0232 DSKSIZ   LSR BLKSUM        ; / 2 ergibt kByte
00813F  08            0233          PHP               ; save flags
008140  38            0234          SEC
008141  A9 4E         0235          LDA #DISK-2       ; Diskgröße - Directory *****
008143  ED FD BF      0236          SBC BLKSUM        ; verbleibenden Speicherplatz
berechnen
008146  A8            0237          TAY
008147  28            0238          PLP               ; get flags
008148  08            0239          PHP               ; and save again
008149  90 01         0240          BCC *+3
00814B  88            0241          DEY               ; bei Divisionsrest abrunden
00814C  20 9E B3      0242          JSR BBTOD
00814F  20 46 EA      0243          JSR NUMA          ; dezimal ausgeben
008152  28            0244          PLP               ; get flags once more
008153  90 0A         0245          BCC DSKS2         ; Divisionsrest berücksichtige
n
008155  A9 2E         0246          LDA #'.'
008157  20 7A E9      0247          JSR OUTPUT        ; ".5" nachschicken
00815A  A9 35         0248          LDA #'5'
00815C  20 7A E9      0249          JSR OUTPUT
00815F  A0 00         0250 DSKS2    LDY #0
008161  B9 CE B3      0251 DSKSLP   LDA SIZMSG,Y      ; Garnierungstext
008164  20 7A E9      0252          JSR OUTPUT
008167  C8            0253          INY
008168  C0 0B         0254          CPY #11
00816A  D0 F5         0255          BNE DSKSLP
00816C  60            0256          RTS
00816D                0257
00816D                0258 ; DATEI ABSCHLIESSEN UND DIRECTORY WEGSCHREIBEN
00816D                0259
00816D  AC FE BF      0260 CLOSFI   LDY BLKCNT
008170  B1 FC         0261          LDA (DPNT),Y
008172  20 AB B3      0262          JSR TANDB
008175  20 BA B3      0263          JSR WRBLK
008178  20 BC B3      0264          JSR WRDIR
00817B  60            0265          RTS
00817C                0266
00817C                0267 ; DATEI VON DISK EINLESEN ( IN=U )
00817C                0268
00817C  B0 1F         0269 RDFILE   BCS RDFIL1
00817E  8A            0270          TXA               ; Carry = 0 --> Initialisierun
g
```

```
00817F  48           0271        PHA             ; Register sichern
008180  98           0272        TYA             ; mit 65 C 02 CPU geht's einfa
cher
008181  48           0273        PHA
008182  20 BB 83     0274        JSR RDDIR       ; Directory einlesen
008185  20 6B 82     0275        JSR NAMINP      ; Dateinamen eingeben
008188  20 99 82     0276        JSR NAMSRC      ; und suchen
00818B  90 03        0277        BCC *+5
00818D  4C BB 82     0278        JMP NOTFND
008190  A0 08        0279        LDY #NAMLEN+1   ; Namen gefunden
008192  8C FE 8F     0280        STY BLKCNT
008195  20 64 83     0281        JSR NXTBLK      ; 1. Block von Disk lesen
008198  68           0282        PLA
008199  A8           0283        TAY
00819A  68           0284        PLA
00819B  AA           0285        TAX
00819C  60           0286        RTS
00819D               0287
00819D  8A           0288 RDFIL1 TXA             ; Carry = 1 --> Lesen
00819E  48           0289        PHA
00819F  98           0290        TYA
0081A0  48           0291        PHA
0081A1  A5 FE        0292        LDA BLKP
0081A3  D0 0C        0293        BNE RDFIL2      ; Blockpuffer komplett
0081A5  A5 FF        0294        LDA BLKP+1      ; ausgelesen ?
0081A7  C9 02        0295        CMP #>BLKBUF+2  ; 2 Pages *****
0081A9  D0 06        0296        BNE RDFIL2
0081AB  EE FE 8F     0297        INC BLKCNT      ; Pointer auf Blocknr. erhöhen
0081AE  20 64 83     0298        JSR NXTBLK      ; nächsten Block von Disk lese
n
0081B1  A0 00        0299 RDFIL2 LDY #0
0081B3  B1 FE        0300        LDA (BLKP),Y    ; Byte aus Puffer holen
0081B5  8D FB 8F     0301        STA SAVA        ; und zwischenspeichern
0081B8  E6 FE        0302        INC BLKP        ; Block-Pointer erhöhen
0081BA  D0 02        0303        BNE RDFIL3
0081BC  E6 FF        0304        INC BLKP+1
0081BE  68           0305 RDFIL3 PLA
0081BF  A8           0306        TAY
0081C0  68           0307        PLA
0081C1  AA           0308        TAX
0081C2  AD FB 8F     0309        LDA SAVA        ; Byte zurück
0081C5  60           0310        RTS
0081C6               0311
0081C6               0312 ;FILE AUF DISK SCHREIBEN ( OUT=U )
0081C6               0313
0081C6  90 03        0314 WRFILE BCC *+5
0081C8  4C 19 82     0315        JMP WRFIL1      ; für "Branch" zu lang
0081CB  8A           0316        TXA
0081CC  48           0317        PHA
0081CD  98           0318        TYA
0081CE  48           0319        PHA
0081CF  20 BB 83     0320        JSR RDDIR       ; Directory einlesen
0081D2  20 6B 82     0321        JSR NAMINP      ; Dateinamen eingeben
0081D5  20 99 82     0322        JSR NAMSRC      ; und suchen
0081D8  A0 00        0323        LDY #0
0081DA  90 0F        0324        BCC OLDFIL      ; Datei schon vorhanden ?
0081DC  20 CB 82     0325        JSR FCBFRE      ; neuen FCB anfordern
0081DF  B9 F4 8F     0326 COPLP  LDA FNAME,Y
0081E2  C8           0327        INY
0081E3  91 FC        0328        STA (DPNT),Y    ; Dateinamen in FCB kopieren
0081E5  C0 07        0329        CPY #NAMLEN
0081E7  D0 F6        0330        BNE COPLP
0081E9  F0 21        0331        BEQ NEWFIL      ; "branch allways"
0081EB  B1 FC        0332 OLDFIL LDA (DPNT),Y    ; Datei existiert bereits
0081ED  29 40        0333        AND #$40        ; "schreibgeschützt" ?
0081EF  F0 15        0334        BEQ UNPROT
0081F1  B9 F0 83     0335 OLDLP  LDA WPMSG,Y     ; Rückversicherung !
0081F4  20 7A E9     0336        JSR OUTPUT
0081F7  C8           0337        INY
0081F8  C0 13        0338        CPY #19
0081FA  D0 F5        0339        BNE OLDLP
0081FC  20 73 E9     0340        JSR REDOUT
```

```
0081FF  C9 4A        0341        CMP #'J'          ; nur dann überschreiben
008201  F0 03        0342        BEQ UNPROT
008203  4C A1 E1     0343        JMP COMIN
008206  20 3A 83     0344 UNPROT JSR BLKEX         ; alte Blockbelegung in FCB
008209  20 E2 82     0345        JSR CLBLK         ; und Bitmap löschen
00820C  A9 08        0346 NEWFIL LDA #NAMLEN+1
00820E  8D FE 8F     0347        STA BLKCNT        ; Pointer setzen und
008211  20 83 83     0348        JSR NEWBLK        ; 1. Block anfordern
008214  68           0349        PLA
008215  A8           0350        TAY
008216  68           0351        PLA
008217  AA           0352        TAX
008218  60           0353        RTS
008219               0354
008219               0355
008219  68           0356 WRFIL1 PLA               ; Byte vom Stack holen
00821A  8D FB 8F     0357        STA SAVA
00821D  8A           0358        TXA
00821E  48           0359        PHA
00821F  98           0360        TYA
008220  48           0361        PHA
008221  A5 FE        0362        LDA BLKP          ; Blockpuffer voll ?
008223  D0 17        0363        BNE WRFIL2
008225  A5 FF        0364        LDA BLKP+1
008227  C9 02        0365        CMP #>BLKBUF+2    ; 2 Pages *****
008229  D0 11        0366        BNE WRFIL2
00822B  AC FE 8F     0367        LDY BLKCNT
00822E  B1 FC        0368        LDA (DPNT),Y      ; aktuelle Blocknr. holen
008230  20 AB 83     0369        JSR TANDS
008233  20 BA 83     0370        JSR WRBLK         ; Blockpuffer auf Disk schreib
en
008236  EE FE 8F     0371        INC BLKCNT        ; Pointer auf Blocknr. erhöhen
008239  20 83 83     0372        JSR NEWBLK        ; nächsten Block anfordern
00823C  AD FB 8F     0373 WRFIL2 LDA SAVA
00823F  A0 00        0374        LDY #0
008241  91 FE        0375        STA (BLKP),Y      ; Byte im Blockpuffer ablegen
008243  E6 FE        0376        INC BLKP          ; Pointer erhöhen
008245  D0 02        0377        BNE WRFIL3
008247  E6 FF        0378        INC BLKP+1
008249  68           0379 WRFIL3 PLA
00824A  A8           0380        TAY
00824B  68           0381        PLA
00824C  AA           0382        TAX
00824D  AD FB 8F     0383        LDA SAVA          ; falls später noch benötigt
008250  60           0384        RTS
008251               0385
008251               0386
008251               0387 ; #####################################
008251               0388 ; ########   UNTERPROGRAMME   ########
008251               0389 ; #####################################
008251               0390
008251               0391 ; Dictionary Pointer initialisieren
008251               0392 ; Entry: ---
008251               0393 ; Exit: DPNT zeigt auf Anfang von DIRBUF
008251               0394 ;        X-Reg. = FCBNUM
008251               0395
008251  A9 00        0396 SETDP  LDA #<DIRBUF
008253  85 FC        0397        STA DPNT
008255  A9 00        0398        LDA #>DIRBUF
008257  85 FD        0399        STA DPNT+1
008259  A2 1F        0400        LDX #FCBNUM
00825B  60           0401        RTS
00825C               0402
00825C               0403 ; Dictionary Pointer inkrementieren
00825C               0404 ; Entry: DPNT zeigt auf aktuellen FCB
00825C               0405 ; Exit: DPNT zeigt auf nächsten FCB
00825C               0406
00825C  18           0407 ADVDP  CLC
00825D  A9 40        0408        LDA #FCBLEN
00825F  65 FC        0409        ADC DPNT
008261  85 FC        0410        STA DPNT
008263  90 02        0411        BCC *+4
```

```
008265  E6 FD        0412          INC DPNT+1
008267  60           0413          RTS
00826B               0414
00826B               0415 ; Eingabe des Dateinamens mit "Prompt"
00826B               0416 ; Entry: ---
00826B               0417 ; Exit: Dateiname in FNAME-Puffer
00826B               0418
00826B  A9 0D        0419 NAMINP LDA #CR
00826A  20 7A E9     0420          JSR OUTPUT
00826D  A9 46        0421          LDA #'F'          ; Prompt ausgeben
00826F  20 7A E9     0422          JSR OUTPUT
008272  A9 3D        0423          LDA #'='
008274  20 7A E9     0424          JSR OUTPUT
008277  A0 00        0425          LDY #0
008279  20 5F E9     0426 NAMIN1 JSR RDRUB
00827C  C9 0D        0427          CMP #CR           ; vorzeitiger Abbruch mit <CR>
00827E  F0 08        0428          BEQ NAMIN2
008280  99 F4 8F     0429          STA FNAME,Y       ; Zeichen ablegen
008283  C8           0430          INY
008284  C0 07        0431          CPY #NAMLEN
008286  D0 F1        0432          BNE NAMIN1
008288  A9 20        0433 NAMIN2 LDA #' '          ; evtl. Rest mit blanks füllen
00828A  C0 07        0434 NAMIN3 CPY #NAMLEN
00828C  F0 06        0435          BEQ NAMIN4
00828E  99 F4 8F     0436          STA FNAME,Y
008291  C8           0437          INY
008292  D0 F6        0438          BNE NAMIN3
008294  A9 0D        0439 NAMIN4 LDA #CR
008296  4C 7A E9     0440          JMP OUTPUT
008299               0441
008299               0442 ; Dateinamen in Directory suchen
008299               0443 ; Entry: gültiger Dateiname in FNAME
008299               0444 ; Exit: gefunden --> Carry = 0, DPNT zeigt auf
008299               0445 ;                            erstes Byte des FCB
008299               0446 ;       nicht gefunden --> Carry = 1
008299               0447
008299  20 51 82     0448 NAMSRC JSR SETDP
00829C  A0 00        0449 NAMS2  LDY #0
00829E  B1 FC        0450          LDA (DPNT),Y
0082A0  29 80        0451          AND #$80          ; FCB belegt ?
0082A2  F0 0F        0452          BEQ NXTNAM
0082A4  B9 F4 8F     0453 NAMS1  LDA FNAME,Y
0082A7  C8           0454          INY
0082A8  D1 FC        0455          CMP (DPNT),Y      ; erstes nicht übereinstimmend
es Zeichen
0082AA  D0 07        0456          BNE NXTNAM        ; beendet Suche im jeweiligen
FCB
0082AC  C0 07        0457          CPY #NAMLEN       ; Name vollständig ?
0082AE  D0 F4        0458          BNE NAMS1
0082B0  18           0459          CLC               ; Gefunden !
0082B1  90 07        0460          BCC NMEX          ; "branch allways"
0082B3  20 5C 82     0461 NXTNAM JSR ADVDP         ; Pointer erhöhen
0082B6  CA           0462          DEX               ; ganze Directory durchsucht ?
0082B7  D0 E3        0463          BNE NAMS2
0082B9  38           0464          SEC               ; war wohl nichts !
0082BA  60           0465 NMEX   RTS
0082BB               0466
0082BB               0467 ; Nachricht ausgeben, falls Datei nicht existiert
0082BB               0468 ; Entry: ---
0082BB               0469 ; Exit: Rückkehr in AIM-Monitor-Hauptschleife
0082BB               0470
0082BB  A0 00        0471 NOTFND LDY #0
0082BD  B9 DA 83     0472 NFLP   LDA NTFMSG,Y
0082C0  20 7A E9     0473          JSR OUTPUT
0082C3  C8           0474          INY
0082C4  C0 0C        0475          CPY #12
0082C6  D0 F5        0476          BNE NFLP
0082C8  4C A1 E1     0477          JMP COMIN
0082CB               0478
0082CB               0479 ; Freien FCB in der Directory suchen
0082CB               0480 ; Entry: ---
```

```
0082CB                  0481 ; Exit: DPNT zeigt auf 1. Byte von freiem FCB
0082CB                  0482 ;       X-Reg. = FCBNUM
0082CB                  0483
0082CB  20 51 82        0484 FCBFRE JSR SETDP         ; Pointer init.
0082CE  A0 00           0485        LDY #0
0082D0  B1 FC           0486 FCBFLP LDA (DPNT),Y
0082D2  F0 09           0487        BEQ FCBFEX        ; Statusbyte = 0 --> FCB frei
0082D4  20 5C 82        0488        JSR ADVDP
0082D7  CA              0489        DEX
0082D8  D0 F6           0490        BNE FCBFLP        ; ganze Directory durchsucht ?
0082DA  4C FC 82        0491        JMP DSKFUL        ; kein Platz mehr
0082DD  A9 80           0492 FCBFEX LDA #$80
0082DF  91 FC           0493        STA (DPNT),Y      ; FCB gleich als belegt markie
ren
0082E1  60              0494        RTS
0082E2                  0495
0082E2                  0496 ; Blockeinträge im FCB löschen
0082E2                  0497 ; Entry: DPNT zeigt auf aktuellen FCB
0082E2                  0498 ; Exit: ---
0082E2                  0499
0082E2  A0 0B           0500 CLBLK  LDY #NAMLEN+1     ; hinter dem Dateinamen anfang
en
0082E4  A9 FF           0501        LDA #BLKEMP
0082E6  91 FC           0502 CLBLP  STA (DPNT),Y      ; Markierung eintragen
0082E8  C8              0503        INY
0082E9  C0 40           0504        CPY #FCBLEN       ; FCB komplett ?
0082EB  D0 F9           0505        BNE CLBLP
0082ED  60              0506        RTS
0082EE                  0507
0082EE                  0508 ; Freien Block in der Bitmap suchen
0082EE                  0509 ; Entry: ---
0082EE                  0510 ; Exit: Block gefunden --> Blocknummer in Reg. A
0082EE                  0511 ;       nichts mehr frei --> Nachricht
0082EE                  0512
0082EE  A0 00           0513 SCAMAP LDY #0
0082F0  B9 E0 8F        0514 SCALP1 LDA BITMAP,Y
0082F3  C9 FF           0515        CMP #$FF          ; Byte voll ?
0082F5  D0 15           0516        BNE SCA1
0082F7  C8              0517        INY
0082F8  C0 14           0518        CPY #MAPLEN       ; ganze Bitmap durchsucht ?
0082FA  D0 F4           0519        BNE SCALP1
0082FC  A0 00           0520 DSKFUL LDY #0            ; $$$$$ Einsprung für FCBFRE $
$$$$
0082FE  B9 E6 83        0521 DFLP   LDA DFMSG,Y       ; Nachricht, falls kein Platz
mehr
008301  20 7A E9        0522        JSR OUTPUT
008304  C8              0523        INY
008305  C0 0A           0524        CPY #10
008307  D0 F5           0525        BNE DFLP
008309  4C A1 E1        0526        JMP COMIN         ; Notlandung !
00830C  48              0527 SCA1   PHA
00830D  98              0528        TYA               ; Reg. Y enthält Byte-Position
00830E  0A              0529        ASL A
00830F  0A              0530        ASL A             ; Byte-Position * 8
008310  0A              0531        ASL A
008311  8D FC 8F        0532        STA NAMCNT        ; tempor. Speicher
008314  68              0533        PLA
008315  A0 00           0534        LDY #0
008317  0A              0535 SCALP2 ASL A             ; Bit-Position extrahieren
008318  90 03           0536        BCC SCA2
00831A  C8              0537        INY
00831B  D0 FA           0538        BNE SCALP2
00831D  98              0539 SCA2   TYA               ; Reg. Y enthält jetzt Bit-Pos
ition
00831E  18              0540        CLC
00831F  6D FC 8F        0541        ADC NAMCNT        ; "Block# = Byte * 8 + Bit"
008322  60              0542        RTS
008323                  0543
008323                  0544 ; Belegten Block in der Bitmap markieren
008323                  0545 ; Entry: Blocknummer in Reg. A
008323                  0546 ; Exit: ---
```

```
008323  48          0548 SETBIT PHA
008324  4A          0549        LSR A           ; Byteposition = Block# / 8
008325  4A          0550        LSR A
008326  4A          0551        LSR A
008327  A8          0552        TAY             ; Reg. Y als Index
008328  68          0553        PLA
008329  29 07       0554        AND #07         ; Bitposition extrahieren
00832B  AA          0555        TAX             ; Reg. X als Zähler
00832C  A9 00       0556        LDA #0
00832E  38          0557        SEC
00832F  6A          0558 SBLP   ROR A           ; zu setzendes Bit in passende
008330  CA          0559        DEX             ; Position schieben
008331  10 FC       0560        BPL SBLP        ; Bit 0 - 7 !
008333  19 E0 BF    0561        ORA BITMAP,Y    ; Bit einfügen
008336  99 E0 BF    0562        STA BITMAP,Y    ; geändertes Byte ablegen
008339  60          0563        RTS
00833A              0564
00833A              0565 ; Überflüssige Blöcke aus Bitmap entfernen
00833A              0566 ; Entry: DPNT zeigt auf aktuellen FCB
00833A              0567 ; Exit: ---
00833A              0568
00833A  A0 08       0569 BLKEX  LDY #NAMLEN+1
00833C  B1 FC       0570 BEXLP1 LDA (DPNT),Y
00833E  C9 FF       0571        CMP #BLKEMP
008340  F0 21       0572        BEQ BEXIT       ; ab hier Rest sowieso leer !
008342  8C FC BF    0573        STY NAMCNT      ; tempor. Speicher
008345  48          0574        PHA
008346  4A          0575        LSR A           ; siehe SETBIT !
008347  4A          0576        LSR A
008348  4A          0577        LSR A
008349  A8          0578        TAY
00834A  68          0579        PLA
00834B  29 07       0580        AND #07
00834D  AA          0581        TAX
00834E  A9 FF       0582        LDA #$FF
008350  18          0583        CLC
008351  6A          0584 BEXLP2 ROR A           ; zu löschendes Bit auf passen
den
008352  CA          0585        DEX             ; Platz schieben
008353  10 FC       0586        BPL BEXLP2
008355  39 E0 BF    0587        AND BITMAP,Y    ; Bit "rausstanzen"
008358  99 E0 BF    0588        STA BITMAP,Y
00835B  AC FC BF    0589        LDY NAMCNT
00835E  C8          0590        INY
00835F  C0 40       0591        CPY #FCBLEN     ; FCB komplett ?
008361  D0 D9       0592        BNE BEXLP1
008363  60          0593 BEXIT  RTS
008364              0594
008364              0595 ; Blockpuffer von Disk füllen
008364              0596 ; Entry: BLKCNT zeigt auf aktuelle Blocknr. in FCB
008364              0597 ; Exit: BLKP steht auf Anfang von BLKBUF
008364              0598
008364  A9 00       0599 NXTBLK LDA #<BLKBUF
008366  85 FE       0600        STA BLKP        ; Pointer setzen
008368  A9 00       0601        LDA #>BLKBUF
00836A  85 FF       0602        STA BLKP+1
00836C  AC FE BF    0603        LDY BLKCNT
00836F  C0 40       0604        CPY #FCBLEN
008371  D0 03       0605        BNE NXBL1
008373  4C FC B2    0606        JMP DSKFUL      ; hier wäre Dateiverkettung mö
glich
008376  B1 FC       0607 NXBL1  LDA (DPNT),Y    ; Blocknummer holen
008378  C9 FF       0608        CMP #BLKEMP
00837A  F0 06       0609        BEQ NXBL2       ; Datei komplett !
00837C  20 AB B3    0610        JSR TANDS       ; Track und Sektor berechnen
00837F  20 B9 B3    0611        JSR RDBLK       ; Disk lesen
008382  60          0612 NXBL2  RTS
008383              0613
008383              0614 ; Neuen Block für Datei anfordern
008383              0615 ; Entry: ---
008383              0616 ; Exit: BLKP gesetzt, Eintrag in FCB und Bitmap
008383              0617
```

```
008383  A9 00        0618 NEWBLK LDA #<BLKBUF
008385  85 FE        0619        STA BLKP          ; Pointer init.
008387  A9 00        0620        LDA #>BLKBUF
008389  85 FF        0621        STA BLKP+1
00838B  20 EE B2     0622        JSR SCAMAP        ; Bitmap durchforsten
00838E  AC FE BF     0623        LDY BLKCNT
008391  C0 40        0624        CPY #FCBLEN       ; FCB voll ?
008393  D0 03        0625        BNE NEWBL1
008395  4C FC B2     0626        JMP DSKFUL        ; hier wäre Dateiverkettung mö
glich
008398  91 FC        0627 NEWBL1 STA (DPNT),Y      ; Blocknummer eintragen
00839A  20 23 B3     0628        JSR SETBIT        ; auch in  der Bitmap
00839D  60           0629        RTS
00839E               0630
00839E               0631 ; Binär-Dezimal Umwandlung
00839E               0632 ; Entry: Binärzahl in Reg. Y
00839E               0633 ; Exit: Dezimalzahl (BCD) in Reg. A
00839E               0634
00839E  18           0635 BBTOD  CLC
00839F  F8           0636        SED
0083A0  A9 00        0637        LDA #0
0083A2  88           0638 BTLP   DEY               ; nur Arithmetik dezimal
0083A3  30 04        0639        BMI BTEX
0083A5  69 01        0640        ADC #1
0083A7  D0 F9        0641        BNE BTLP          ; "branch allways"
0083A9  D8           0642 BTEX   CLD
0083AA  60           0643        RTS
0083AB               0644
0083AB               0645 ; Berechnung von Track und Sektor aus Blocknr.
0083AB               0646 ; Entry: Blocknr. in Reg. A
0083AB               0647 ; Exit: Track in Reg. X, Sektor in Y
0083AB               0648
0083AB               0649 ; ***** Bei anderer Diskorganisation *****
0083AB               0650 ; ***** entsprechend anzupassen      *****
0083AB               0651
0083AB  48           0652 TANDS  PHA
0083AC  4A           0653        LSR A
0083AD  4A           0654        LSR A             ; Track = Blocknr. / 4
0083AE  AA           0655        TAX
0083AF  68           0656        PLA
0083B0  29 03        0657        AND #03           ; A = Blocknr. MOD 4
0083B2  0A           0658        ASL A
0083B3  0A           0659        ASL A             ; Sektor - 1 = A * 4
0083B4  A8           0660        TAY
0083B5  C8           0661        INY               ; Sektor 0 für FDC meist nicht
 definiert !
0083B6  60           0662        RTS
0083B7               0663
0083B7               0664 ; ************************************************
0083B7               0665 ; *****  Hardware-abhängige Unterprogramme  *****
0083B7               0666 ; ************************************************
0083B7               0667
0083B7               0668 ; Initialisierung für I/O-Bausteine und FDC
0083B7               0669
0083B7               0670 OPEN
0083B7               0671                          ; je nach verwendetem FDC
0083B7               0672
0083B7  60           0673        RTS
0083B8               0674
0083B8               0675 ; Formatieren der Diskette
0083B8               0676 ; Entry: ---
0083B8               0677 ; Exit: eine "blitzblanke" Disk
0083B8               0678
0083B8               0679 FORM1
0083B8               0680                          ; je nach verwendetem FDC
0083B8               0681
0083B8  60           0682        RTS
0083B9               0683
0083B9               0684 ; Füllen des Blockpuffers von der Disk
0083B9               0685 ; Track in Reg. X, Nummer des ersten Sektors in Y
0083B9               0686
```

```
0083B9                  0687 RDBLK
0083B9                  0688                            ; Sektor aus Reg. Y und die dr
ei folgenden
0083B9                  0689                            ; Sektoren ( insgesamt 512 Byt
e ) lesen
0083B9                  0690                            ; Register frei verfügbar
0083B9                  0691                            ; DIRP und BLKP nicht ändern
0083B9                  0692
0083B9  60              0693            RTS
0083BA                  0694
0083BA                  0695 ; Wegschreiben des Blockpuffers auf die Disk
0083BA                  0696 ; Track in Reg. X, Nummer des ersten Sektors in Y
0083BA                  0697
0083BA                  0698 WRBLK
0083BA                  0699                            ; Sektor aus Reg. Y und die dr
ei folgenden
0083BA                  0700                            ; Sektoren ( insgesamt 512 Byt
e ) schreiben
0083BA                  0701                            ; siehe RDBLK
0083BA                  0702
0083BA  60              0703            RTS
0083BB                  0704
0083BB                  0705 ; Einlesen der kompletten Directory von Disk
0083BB                  0706
0083BB                  0707 RDDIR
0083BB                  0708                            ; 16 Sektoren von Track 0 ( 20
48 Byte ) lesen
0083BB                  0709                            ; multiple record mode vorteil
haft
0083BB                  0710                            ; Register und Pointer frei
0083BB                  0711            .
0083BB  60              0712            RTS
0083BC                  0713
0083BC                  0714 ; Wegschreiben der kompletten Directory
0083BC                  0715
0083BC                  0716 WRDIR
0083BC                  0717                            ; 16 Sektoren von Track 0 ( 20
48 Byte ) schreiben
0083BC                  0718                            ; siehe RDDIR
0083BC                  0719
0083BC  60              0720            RTS
0083BD                  0721
0083BD                  0722 ; System messages
0083BD                  0723
0083BD  46 4F           0724 FORMSG .BYT 'FORMATIEREN J/N ?'
0083CE                  0725
0083CE  4B 20           0726 SIZMSG .BYT 'K BYTE FREI',CR
0083D9  0D              0726
0083DA                  0727
0083DA  4B 45           0728 NTFMSG .BYT 'KEINE DATEI',CR
0083E5  0D              0728
0083E6                  0729
0083E6  44 49           0730 DFMSG  .BYT 'DISK VOLL',CR
0083EF  0D              0730
0083F0                  0731
0083F0  5D 42           0732 WPMSG  .BYT 'ÜBERSCHREIBEN J/N ?'
008403                  0733
008403                  0734        .END
```

Kleinanzeige

Video-Terminal mit Cherry-Tastatur (im Gehäuse), MC-Video-Platine (Regge) und ggfs. Sanyo-Monitor DM 5912CX (grün, 18 MHz Bandbreite) funktionierend abzugeben. Betrieb mit Schnittstellen RS232C, TTY, einstellbare Baudraten und Übertragungsarten. R. Löhr, Tel. 04102-55 816.

VMEbus-Karte FORCE SYS68k, CPU1 mit MC68000 (8 MHz), 128 KB RAM (on board erweiterbar bis 512 KB), 4 EPROM-Steckplätze (bis 64 KB), 3 RS232C-Kanäle mit einstellbaren Parametern, Interfacebaustein PI/T MC68230 (Parallel Interface/Timer), mit Betriebsystem/Monitor/Debugger/line by line Assembler wowie FORCE IDEAL Screen Editor und 2 Pass-Assembler, mit Dokumentation, Neuwert ca. DM 4200 günstig abzugeben: R. Löhr, Tel. 04102 - 55 816.

Martin Albrecht, 4350 Recklinghausen
Hans-Joachim Regge, 2800 Bremen

MC-65 als Speichererweiterung

am AIM 65

64 K RAM und die RAM/ROM-Umschaltung mit Boot-Möglichkeit sind die herausragenden Eigenschaften der MC-65 /1/, /2/. Wer schon einen AIM 65 besitzt und die Anschaffung eines MC-65 und eines Terminals scheut, braucht trotzdem nicht darauf zu verzichten: Die MC-65 CPU-Karte läßt sich durch Weglassen einiger Bauteile als Speichererweiterung für den AIM 65 einsetzen. Sie kann später zum vollständigen System ergänzt werden.

Die Verwendung der MC-65 Karte bietet gegenüber bisherigen Speichererweiterungen viele Vorteile. Die Platine im Europaformat findet auch im kleinsten Gehäuse Platz und kommt mit +5 Volt Betriebsspannung aus. Die Stromaufnahme des AIM-65 erhöht sich auf maximal 1,1 A, so daß das vorhandene Netzteil sicher ausreicht. Erstmals steht ja auch im Bereich von hex B000-FFFF RAM zur Verfügung. Per Tastendruck wird der ROM-Inhalt ins RAM übertragen und zwischen RAM und ROM umgeschaltet. Das lästige Umstecken der ROMs beim Wechsel der Programmiersprache entfällt. Sie wird jetzt einfach ins RAM geladen. Durch Umschalten auf ROM steht eine zweite Sprache simultan zur Verfügung.

Hardware ...

Grundlage für den Aufbau der MC-65 Karte ist die Beschreibung in /1/ oder /2/. Tabelle 1 zeigt die geänderte Stückliste. Weggelassen werden vor allem der RIOT 6532, der VIA 6522, das Kassettenrekorder-Interface, die Reset-Logik, die Single Step-Logik und die Bustreiber für den RM 65-Bus.

Der Prozessor und die ROMs finden ihren neuen Platz auf der MC-65 Karte. Sämtliche RAM-Bausteine 2114 (Z2, Z3, Z6, Z7, Z11, Z12, Z17, Z18) werden von der AIM-Platine entfernt. Auf beiden Leiterplatten ist zusätzlich folgende Verdrahtung erforderlich:

```
MC-65:  IC-8 (6502) Pin 35 an IC 14 (6349) Pin 1
        IC 7 (6532) Pin 16 an IC 7  (6532) Pin 24
        IC 7 (6532) Pin 24 an IC 8  (6502) Pin 36

AIM-65: Z 27 (74LS138) Pin 13 an Z 9 (6502) Pin 35
        U 1  (6520)    Pin 39 an Z 9 (6502) Pin 36
```

Huckepack

Die Verbindung von AIM-65 und MC-65 geschieht über den Prozessorsockel. Der 6502-Platz erhält auch auf der Lötseite der MC-65 Platine einen Sockel (IC8). Die Karte wird dann mit einem Zwischenstecker huckepack in die Prozessorfassung des AIM-65 gesteckt. Man kann sie aber auch getrennt montieren und ein Kabel mit 40-poligen DIL-Steckern verwenden. Dazu kommt auch auf die Lötseite der AIM-Platine ein Sockel unter den Prozessor. Die Verbindung soll möglichst kurz sein.

Vor dem Zusammenstecken bzw. vor dem Löten werden die Pins 1, 8 und 21 des Zwischensteckers oder der zusätzlichen Steckfassung entfernt. Die Stromversorung der MC-65 Karte erfolgt nicht über den Sockel, sondern durch Pin 1A (Masse) und Pin 32A (+5V) der DIN 41612 Steckverbindung. Pin 37 des Sockels wird ebenfalls weggelassen, denn die Takterzeugung geschieht auf der MC-65 Karte. Wer sicher gehen will, kann den Oszillator des AIM stillegen. Pin 37 sollte möglichst auf der AIM-Seite der Verbindung entfernt werden, um Hochfrequenzstörungen zu vermeiden.

Zur Ansteuerung von DESELECT und RAMHIGH des MC-65 werden die nicht belegten Pins 35 und 36 des 6502 benutzt. DESELECT ist über Pin 35 mit CSA auf der AIM-Platine verbunden. Dadurch wird der Bereich $Axxx des MC-65 ausgeblendet, und die I/O-Bausteine auf der AIM-Platine bleiben aktiv. Zum Ausblenden der ROMs ist der unbenutzte Anschluß CA2 des Display-PIA 6520 über Pin 36 mit RAMHIGH verbunden.

... und Software

Ein kurzes Programm (siehe Liste) besorgt die Umschaltung. Es wird mit Taste F1 gestartet. Drücken von 'B' lädt den Inhalt der ROMs ins RAM und schaltet auf RAM um. 'A' schaltet ohne Boot auf RAM und'O' zurück auf ROM. Nach einem Reset ist immer das ROM eingeschaltet.

Schlußbemerkung

In MICRO MAG sind schon öfter Vorschläge für Speichererweiterungen erschienen /3/, /4/, /5/, /6/. Die hier gezeigte Lösung liegt im Preis deutlich über reinen Selbstbaulösungen. Andere zur Zeit erhältliche 64K-Karten bieten jedoch bei gleichen Kosten weniger Vorteile. Der gezeigte Aufbau ist besonders für Leser gedacht, die weniger der Hardware zuneigen und eine industriell gefertigte Platine einem Drahtverhau vorziehen.

Beim Autor (Albrecht) ist die Karte seit einigen Wochen unter DISKMON /7/ in Betrieb. Vorteilhaft erscheint besonders die Möglichkeit, Utilities im Bereich B000-DFFF durch Bankumschlatung ständig zur Vergügung zu haben. Zur Ansteuerung eines Centronics-Druckers wird hier aus dem Monitorprogramm heraus auf ROM umgeschaltet, die Druckroutine abgearbeitet, und danach wieder in den alten Zustand (RAM oder ROM) zurückgeschaltet. Die parallel im RAM stehende Sprache (z.B. BASIC) kann dadurch wichtige Dienstprogramme problemlos mitbenutzen.

Tabelle 1: Stückliste zur MC-65 Karte als Speichererweiterung

IC 5, IC 9, IC 10	EPROM 2532 oder Rockwell-ROM
IC 11, IC 15	
IC 8	6502-CPU
IC 12	74 LS 12
IC 13	74 LS 04
IC 14	6349 (Adreßdecoder-PROM)
IC 18	74 LS 393
IC 19	74 LS 74
IC 20	TMS-4500-A-NL (Dynamic-Memory-Controller)
IC 21 ... IC 28	TMS-4164-15-NL (RAM 64 K * 1)
C1, C3, C6, C10	0,1 µF, RM 5 mm, 2,5 mm Breite, Wima MKS-2, 40 V
C13 ... C23	
C2, C4, C5, C7, C11	47 nF Keramik, niedr. Bauhöhe, RM 5 mm
C24, C25	33 µF/10V RM 5 mm
R10, R11, R12	470 Ohm
R25	1 kOhm
R3, R21	4,7 kOhm
D1	1 N 4148
Q	8-MHz-Quarz, HC 43 U (einlötbar)

Sockel mit gedrehten Kontakten für alle ICs
1 Zwischenstecker 40-pol. DIL oder
40-adriges Flachbandkabel mit 40-pol. DIL-Steckern
1 Platine MC-65-CPU (Regge-Elektronik, Fesenfeld 57, 2800 Bremen 1)

```
---------------------------------------------------------------
MC-65 als Speichererweiterung am AIM-65

0000          ;Boot und Umschaltung RAM/ROM
0000
---------------------------------------------------------------
Monitorunterprogramme

0000 COMIN          =$E1A1          ;Monitoreinsprung
0000 CRLOW          =$EA13          ;CR/LF ausgeben
0000 OUTPUT         =$E97A          ;ein Zeichen ausgeben
0000 READOUT        =$E973          ;Tastaturabfrage
0000
---------------------------------------------------------------
Variablen

0000 CRA            =$AC01          ;Display-PIA Control Register A
0000 ZEIGER         =$FE            ;Pointer für Boot
```

```
0000                    *=$200
0200
-------------------------------------------------------------
Hauptprogramm

0200 QUEST   2013EA JSR CRLOW        ;Display löschen
0203         A000   LDY #0
0205 QUEST1  B95102 LDA MES,Y        ;Meldung ausgeben
0208         207AE9 JSR OUTPUT
020B         C8     INY
020C         C014   CPY #20
020E         D0F5   BNE QUEST1
0210         2073E9 JSR READOUT      ;Tastatur abfragen
0213         C941   CMP #'A'         ;'A'=RAM ?
0215         F023   BEQ RAM          ;ja
0217         C94F   CMP #'O'         ;'O'=ROM ?
0219         F02B   BEQ ROM          ;ja
021B         C942   CMP #'B'         ;'B'=Boot ?
021D         D02F   BNE MONI         ;nein
021F
-------------------------------------------------------------
ROM ins RAM übertragen und RAM einschalten

021F BOOT    AD01AC LDA CRA          ;vor Boot auf ROM schalten
0222         0938   ORA #%00111000
0224         8D01AC STA CRA
0227         A0B0   LDY #$B0         ;ZEIGER mit $B000 laden
0229         84FF   STY ZEIGER+1
022B         A000   LDY #0
022D         84FE   STY ZEIGER
022F BOOT1   B1FE   LDA (ZEIGER),Y   ;und ROM in's RAM übertragen
0231         91FE   STA (ZEIGER),Y
0233         C8     INY
0234         D0F9   BNE BOOT1
0236         E6FF   INC ZEIGER+1
0238         D0F5   BNE BOOT1
023A         ;nach Boot auf RAM umschalten
023A
-------------------------------------------------------------
Umschaltung auf RAM

023A RAM     AD01AC LDA CRA
023D         29F7   AND #%11110111   ;setze CRA = XX110XXX, CA2 Low-Output
023F         0930   ORA #%00110000
0241         8D01AC STA CRA
0244         D008   BNE MONI
0246
-------------------------------------------------------------
Umschaltung auf ROM

0246 ROM     AD01AC LDA CRA
0249         0938   ORA #%00111000   ;setze CRA = XX111XXX, CA2 High-Output
024B         8D01AC STA CRA
024E
024E MONI    4CA1E1 JMP COMIN        ;Monitoreinsprung
0251
-------------------------------------------------------------
Textmeldung

0251 MES     424F   .BYT 'BOOT=B,RAM=A,ROM=O?'

0264
-------------------------------------------------------------
Anbindung an Taste <F1>

0264                    *=$10C
010C         4C0002 JMP QUEST
010F
010F                    .END
ERRORS=0000
```

Anton Ruepp, CH-8153 Rümlang

Soft Spy

für AIM 65

In zeitkritischen Assemblerprogrammen läßt sich der single step-Betrieb nicht anwenden. Wenn sich ein Programm aufhängt oder wenn es auf etwas wartet, was jetzt nicht eintrifft, bleibt meist nur noch eine Rettung - die Reset-Taste.. Damit ist aber dem Entwickler nicht viel geholfen, die Spuren sind verwischt, und er hat keine Möglichkeit herauszufinden, wo das Programm im Kreis gelaufen ist. - Viele Wege führen nach Rom, hier ein interessanter:

Mit der vorgestellten Hilfsroutine kann der Computer zur Herausgabe seiner Geheimnisse bewegt werden. Auf Knopfdruck präsentiert er seinen Prozessorstatus, den Programmzählerstand, den Akku und das X- und Y-Register. Aufgrund dieser Daten läßt sich dann das auf Abwegen befindliche Programm rasch eingrenzen. Es wird hier der Umstand ausgenutzt, daß die CPU bei einem Interrupt die momentane Adresse und den Prozessorstatus auf den Stack rettet.

Das Vorgehen bei Programmen ohne Interrupts:

Wenn die Hilfsroutine geladen ist, wird von Hand der IRQ-Vektor in Adresse A404 auf den Anfang der Routine gestellt. Beispiel: Die Routine beginnt bei 9Ed6. Dann mittels M-Kommando des Monitors den Inhalt von A004 auf D6 und von A405 auf 9E ändern. - Allfällige SEI-Befehle müssen aus dem zu untersuchenden Programm entfernt werden. Dafür wird ganz am Anfang ein CLI eingesetzt. - Programm starten und bei Bedarf die IRQ-Leitung (am Expansion Stecker, Pin 4) kurz auf Masse tippen. Jetzt erscheinen PS PC A X und Y auf der Anzeige.

Bei Programmen mit Interrupts:

Hier muß auf den NMI-Interrupt ausgewichen werden. Der Vektor liegt dann in A402/A403. Das Programm muß deswegen nicht geändert werden. Pin 6 des Expansionssteckers aber nur ganz kurz an Masse tippen.

Das Musterprogramm belegt in der Zeropage die Plätze 0-6. Es ist relozierbar.

```
*************************************************************************
SOFT SPY
000000                  0001 ;PS, PC, AKKU UND REGISTER AUSGEBEN
000000                  0002 ;MITTELS MANUELLEM INTERRUPT
000000                  0003 ;IRQ PIN 4 / NMI PIN 6 EXPANSION ST.
*************************************************************************
ZEROPAGE
000000                  0005         *=0
000000                  0006 BUF     *=*+3 +TEMPORAER PS, PC
000003                  0007 SAVA    *=*+1            ;SAVE A
000004                  0008 SAVX    *=*+1            ;SAVE X
000005                  0009 SAVY    *=*+1            ;SAVE Y
*************************************************************************
MONITORROUTINEN
000006                  0011 CRLOW   =$EA13           ;DISPLAY CLEAR
000006                  0012 BLANK2  =$E83B           ;2 BLANKS
000006                  0013 NUMA    =$EA46           ;2 HEXZAHLEN AUSG
*************************************************************************
PROGRAMM
000006                  0015         *=$7900
007900  85 03           0016 INTER   STA SAVA         ;RETTE AKKU
007902  86 04           0017         STX SAVX         ;RETTE X
007904  84 05           0018         STY SAVY         ;RETTE Y
007906  68              0019         PLA              ;HOLE STATUS VOM
007907  85 00           0020         STA BUF          ;STACK, VERSORGE IHN
007909  68              0021         PLA              ;LOWBYTE DER ADRESSE
00790A  85 01           0022         STA BUF+1        ;VERSORGE ES
```

```
00790C  68          0023           PLA               ;DITO HIGHBYTE
00790D  85 02       0024           STA BUF+2
00790F  48          0025           PHA               ;RESTORE STACK
007910  A5 01       0026           LDA BUF+1
007912  48          0027           PHA               ;RESTORE STACK
007913  A5 00       0028           LDA BUF
007915  48          0029           PHA               ;RESTORE STACK
007916              0030 ;STACK KOPIERT, JETZT ANZEIGEN
007916  20 13 EA    0031           JSR CRLOW         ;CLEAR DISPLAY
007919  A5 00       0032           LDA BUF
00791B  20 46 EA    0033           JSR NUMA          ;PS AUSGEBEN
00791E  20 3B E8    0034           JSR BLANK2
007921  A5 02       0035           LDA BUF+2
007923  20 46 EA    0036           JSR NUMA          ;ADH AUSGEBEN
007926  A5 01       0037           LDA BUF+1
007928  20 46 EA    0038           JSR NUMA          ;ADL AUSGEBEN
00792B  20 3B E8    0039           JSR BLANK2
00792E  A5 03       0040           LDA SAVA
007930  20 46 EA    0041           JSR NUMA          ;AKKU AUSGEBEN
007933  20 3B E8    0042           JSR BLANK2
007936  A5 04       0043           LDA SAVX
007938  20 46 EA    0044           JSR NUMA          ;X AUSGEBEN
00793B  20 3B E8    0045           JSR BLANK2
00793E  A5 05       0046           LDA SAVY
007940  20 46 EA    0047           JSR NUMA          ;Y AUSGEBEN
007943  A5 03       0048           LDA SAVA          ;RESTORE AKKU
007945  A6 04       0049           LDX SAVX          ;UND X
007947  A4 05       0050           LDY SAVY          ;FERNER Y
007949  40          0051           RTI               ;ZUR. INS PROGR.
00794A              0052           .END
```

Aus der Branche

GTE-Mikroprozessoren 65SC802 und 65SC816 sowie CMOS-Peripheriebausteine: Die in dieser Zeitschrift in ihrer Leistungsfähigkeit und Programmierung schon besprochenen CMOS-Mikroprozessoren sollen etwa im Juli 1985 in Serie lieferbar sein. Für CMOS-Lösungen sind schon lieferbar die Interfacebausteine GS65SC21 PIA, x22 VIA, x32 RIOT und x51 ACIA, so daß zusammen mit CMOS-RAMs und EPROMs die Möglichkeit gegeben ist, äußerst stromsparende Systeme aufzubauen.

ZI GmbH, in 8033 Martinsried: Diese Firma stellt Geräte für die Zeiterfassung und Zugangskontrolle mit CPU 6502 her. Auf der Hannover-Messe wurde bereits ein Computer mit CPU G65SC816 gezeigt, ausgestattet mit 5 Bänken zu je 64 KB CMOS-RAM. Es war beeindruckend, die CPU in ihrer Geschwindigkeit bei Suchbefehlen über die Bänke hinweg und bei Block-Moves zu beobachten. Eine Speicherbank ist transportiert, ehe man die Return-Taste richtig losgelassen hat.

VALVO, I2C-Bus Monitor. Der serielle I2C-Bus ist inzwischen in vielen Geräte-Entwicklungen mit Mikrocontrollern eingesetzt worden. Neben den Mikrocontrollern der Familie MAB 84000, in denen die I2C-Bus Steuerlogik integriert ist, steht jetzt schon eine ganze Reihe von Peripherieschaltungen zur Verfügung. Als Monitor für den Bus steht jetzt die Platine MCT-SIO bei den Distributoren zur Verfügung (Bestellnummer 293 0042), die die Entwicklung von Hard- und Software für den Bus unterstützen soll. Sie hat wie damals der KIM-1 ein hexadezimales Tastenfeld, 8 Steuertasten, eine 16stellige 7 Segment-LCD und 11 LED's für die Anzeige interner Flags.

Commodore und Atari: Die wesentlichen Merkmale der neuen in Hannover vorgestellten Computer sind im Editorial dieses Heftes enthalten. Eine erste kleine Serie des PC128 von Commodore soll im Mai 1985 auf den Markt kommen. Die regelmäßige Liefermöglichkeit ist für etwa Juli angekündigt. Juli s gilt auch für die interessanten ST-Typen von Atari.

Wiesemann Mikrocomputertechnik in Wuppertal: Der Herausgeber erhielt einen 50seitigen Katalog mit der Beschreibung von Interfaces für CBM, C64, Atari auf die wesentlichen Schnittstellen IEEE488, C64 seriell, Centronics, RS232/V24.

Editorial

Der Herausgeber besuchte am Eröffnungstage die Hannover-Messe. Die Zeit reichte gerade, um die wichtigsten Messestände zu besuchen, nicht aber, um auch an den angesetzten Pressekonferenzen teilzunehmen. Von der Lage in der Branche folgende Eindrücke:

Der Wettbewerb bei Computern ist schärfer geworden. Es hat bereits 1983/84 Firmenzusammenbrüche gegeben oder Rückzüge vom Markt für Personal- und Heimcomputer. Das letzte Quartal 1984 und der Jahresbeginn 1985 waren für viele Anbieter so hart, daß ganze Fabriken stillgelegt wurden, daß man hier und da Kurzarbeit wählte und auch Modelle einstellte, z.B. den IBM PC jr. Bei den billigen Heimcomputern (Klasse des erfolgreichen C-64) scheint die Bergspitze des mengenmäßigen Absatzes überschritten zu sein. Es soll in dieser Klasse weltweit einen großen Überhang noch unverkaufter Geräte geben.

Bei den größeren Personal- oder Arbeitsplatzcomputern geht der Trend ganz offensichtlich zu IBM und den 'kompatiblen', wobei zu erwarten ist, daß hier noch weitere Mitanbieter einen Rückzieher machen müssen, weil hart gekämpft wird.

Bemerkenswert ist, daß Commodore, vormals mit der Serie 8032 hierzulande sehr erfolgreich, sein Nachfolgemodell 8296 in Hannover gar nicht mehr so sehr herausstellte. In der Klasse der Personal Computer scheint man mehr auf die kompatiblen PC 10 und PC 20 zu setzen. Darüber soll im Herbst die Serie 900 (mit CPU Z8000) gesetzt werden, die unter einer UNIX-Version mehrplatzfähig ist. Dieses System wurde bereits auf der Messe gezeigt. - Man begeht in der Reihe der PC 10, PC 20 und er 900 also den Weg der verbreiteten Betriebssysteme für Maschinen mit 16 Bit.

Ein 'Mischling' ist der bei Commodore auf der Messe ebenfalls gezeigte PC 128. Er wird drei Betriebsarten haben: Den C-64-Modus, PC 128 Modus mit erweitertem BACIC 7.0 und schließlich die Betriebsart CP/M. Die beiden ersten sind also Commodore-Betriebsarten. Mit einer auch eingebauten CPU Z80A (4 MHz) sucht man dann Anschluß an ein fremdes Betriebssystem, nämlich CP/M. Als Endverbraucherpreise werden genannt je DM 1190,- (inkl. MWST) für den PC 128 und seine Floppy 1571 sowie DM 1298,- für einen RGB-Datenmonitor (farbig). Der Computer hat 128 KB RAM-Speicher. Ob er richtig im Markt liegen wird, kann man erst nach Auslieferung der Serie im Juli beurteilen, denn es muß auch noch ein Blick auf den Wettbewerb geworfen werden. - Man geht aber wohl nicht Fehl in der Annahme, daß der PC 128 in der unteren Klasse der letzte Rechner mit einem Commodore-Betriebssystem sein wird. Und trotz vieler lobenswerter Merkmale dürfte er die Stückzahlen des C-64 angesichts zunehmender Marktsättigung nicht mehr erreichen, auch wenn er im Preis deutlich unter DM 1000 taucht.

Für viel Wirbel dürfte in nächster Zeit Atari sorgen. Es dürfte bekannt sein, daß Jack Tramiel und viele andere Commodore-Manager zu Atari überwechselten. Diese Firma hat nach schweren Rückschlägen sicher nicht die gleiche finanzielle Kraft wie andere. Der auf der Messe vorgestellte 520 ST als Spitzengerät schwingt jedoch deutlich die Keule im Wettbewerb: Computerconsole mit CPU 68000, 512 KB RAM, 192 KB ROM, 32 KB Video-RAM, farbtüchtig bis 640x200 pixels, maximal 640x400 pixels monochrom, inkl. Foppy mit 500 KB Speichervolumen zu DM 2795,- (Endverbraucherpreis). Dazu muß man natürlich noch den notwendigen Bildschirm hinzurechnen. Eine Maus ist wohl im Preis schon inbegriffen. Das gezeigte File-Managerprogramm ähnelt dem des McIntosh und dem der LISA. Sprachen und weitere Softwarepakete wurden auf der Messe als 'kommend' bezeichnet, ebenso deutsche Sonderzeichen auf Bildschirm und Tastatur sowie deutsche Bedienerführung und Dokumentationen.

Dieses Gerät wird auf jeden Fall den McIntosh angreifen, der nicht farbtüchtig aber teurer ist, der nur einen kleinen Monitor-Schirm hat und dessen Speicher für heutige Begriffe zu klein ist, wenn man von weiteren Beschwerden von Benutzern einmal absieht. - Und der Atari wird auch Interessenten eines PC 128 von Commodore die Frage stellen, ob sich etwa DM 400 Mehrausgabe für die eigene Nutzenvorstellung lohnen. - Beim Betriebssystem, TOS genannt, dürfte es sich um ein firmeneigenes handeln, das dann nicht kompatibel ist.

Nach diesem Blick auf wichtige Anbieter kann man wohl wie folgt zusammenfassen: Bei fertigen Computern erleben wir mehr und mehr den Übergang auf Systeme mit 16 Bit, entweder auf der Intel/IBM-Linie mit 8086/8088 oder 68xxx (Apple, Atari). 68xxx wird ferner in großem Umfang

für Steuerungen vorgesehen, so daß für diese Familie und ihre Anbieter die Marktbasis für Weiterentwicklungen vor allem in Ergänzungsbausteinen gegeben ist. - Der 8 Bit-Computer ist damit nicht tot, er hat noch viele unerfüllte Aufgaben in Steuerungen zu erledigen, in Sachgebieten, die bisher noch nicht angefaßt wurden. Vor allem sind damit die Single Chippers mit 8 Bit und möglichst CMOS-Technik nicht in Ihrem Vormarsch in die Steuerungen behindert.

In letzter Zeit tritt häufig das Wort 'Benutzeroberfläche' auf. Gemeint ist, wie sich der Computer dem Anwender für die Hantierung präsentiert, letztlich also das Betriebssystem und seine Interaktionen mit dem Benutzer. So sollte ein Textsystem sich einer Schreibdame nicht als Computer mit umständlichen Startprozeduren präsentieren, es sollte nach dem Einschalten fertig für die Texteingabe sein. - Bei den Betriebssystemen sehen wir, von Ausnahmen noch abgesehen, eine Linie, die vom CP/M über CP/M-68K und MS-DOS bis hin zu UNIX (Mehrbenutzersysteme) in den verschiedenen Versionen reicht. Die höheren Stufen bedeuten aber zugleich, daß in ihnen auch die Sprache 'C' erreichbar ist, um die man sich kümmern muß, wenn man beruflich an der weiteren Entwicklung teilnehmen will.

Bei künftigen Erneuerungen im Computerbestand wird man also sehr darauf achten müssen, welche Chancen der Anbieter und sein Modell für das Überleben haben, so daß man mit einer weiteren Unterstützung rechnen kann.

Bei Hardware wird ferner darauf zu achten sein, daß sie an den beiden wichtigen Interfaces des Benutzers ergonomisch richtig gestaltet ist. Das Bild muß flackerfrei, gut lesbar und augenfreundlich sein. Der Schirm sollte entspiegelt und schwenkbar sein. Bei der Tastatur ist zu fordern, daß die Tasten möglichst an der erwarteten Stelle liegen, prellfrei und ohne ungewollte Berührung von Nachbartasten erreichbar sind. Der Versuch auf verschiedenen Tastaturen auf der Hannover-Messe zeigte, daß wir sogar bei Terminals für den Berufsalltag von diesem Ziel noch weit entfernt sind. Und man achte auch auf deutsche Sonderzeichen auf dem Bildschirm und auf deutsche Tastaturbelegung, sowie auf die entsprechende Ausgabe dieser Sonderzeichen auf dem Drucker. Als Anwender haben wir das Recht, auf der Erfüllung dieser fast mühelos zu lösender Kleinigkeiten zu bestehen.

Man wird künftig auch mehr Wert auf große Speicher legen, besonders auch für die Programmentwicklung. Dem Autor stehen z.B. auf seinem Textsystem 3 Textbänke mit 64 KB zur Verfügung, das entspricht etwa 2200 Zeilen mit 80 Zeichen. Bei ausreichender Kommentierung und gegliederter Anordnung des Quelltextes entspricht das in Assembler nur etwa 4 KB Programm. Viele gedankliche Gebilde sind jedoch umfangreicher, und da ist es vorteilhaft, wenn man mühelos zwischen den Textstellen, die in die Maschine geladen sind, umblättern kann. So sind aus der Sicht des Entwicklers und Programmierers 512 KB Textspeicher doch schon sehr erstrebenswert.

Textsysteme und Anregungen zur Fortentwicklung derselben sind daher trotz verschiedener früher veröffentlichter Arbeiten nach wie vor 'in'. Dieses Heft enthält dementsprechend einen Screeneditor, der unter den erschwerten Bedingungen der seriellen Datenübertragung arbeitet. Cursorbewegungen müssen dabei in Beziehung zu den Textstellen gebracht werden. Der Herausgeber meint, daß das Programm recht einfach auch für andere Befehlssätze als für 65C02 umzusetzen ist. Bei einer Implementierung sollte man überlegen, ob man nicht gleich auch die in diesem Heft beschriebenen Methoden zum beschleunigten Stringsuchen einbaut. - Der Editor und die grundsätzliche Beschreibung eines DOS von Andreas Zilker entsprechen der vom Herausgeber angestrebten Linie, nicht nur die kleineren verbessernden Hantierungen und Veränderungen zu beschreiben, sondern auch größere Zusammenhänge, und diesen dann auch den notwendigen Platz einzuräumen. - In diesem Sinne sind die Leser weiterhin gebeten, ihre Beobachtungen und Arbeiten einem breiteren Leserpublikum vorzustellen. Neben Themen wie in der Vergangenheit sollte man künftig auch gerne über Einchipper und den Betrieb neuer Interfacebausteine berichten, auch über die Benutzung der Betriebssysteme und Ergänzungspakete, die unter ihnen laufen.

Bücher

Hilf, W; Nausch, A.: M68000 Familie, Teil 2, Anwendung und 68000-Bausteine. te-wi Verlag, München 1984, ca. 350 S., ISBN 3-921803-30-6, DM 59. Seit Februar 1985 ist dieses Buch nun endlich

lieferbar. Der erste Band der Autoren wurde in Heft 37 besprochen und für gut befunden. Dieses Urteil darf man auch auf den zweiten Teil ausdehnen. Er hat folgende Hauptabschnitte: Assembler-Programmiertechnik mit M68000-Befehlen, M68000 Peripheriebausteine, Die nächste Generation (68008, 68010, 68012, 68020), Ein Single Board System, Anhänge mit Daten. Innerhalb des erstgenannten Abschnittes finden wir Programmierungsbeispiele für Code-Wandlung, Mathematik, Sortier- und Suchprogramme und verschiedene Utilities. Die Programme sind dabei mit Flußdiagrammen und Schemazeichnungen erklärt und mit einer kommentierten Assembler-Liste versehen. Wer noch mehr Software studieren will, der findet beim Single-Board-Computer nicht nur einen Schaltplan, sondern auch ein ganzes Monitor-Programm dafür nebst Erklärungen und Assembler-Liste auf etwa 80 Seiten. Der Designer wird sich über die verständliche Darstellung der Peripheriebausteine und der neueren CPU-Typen freuen. Bei letzteren findet man dann auch die Erklärung der neuen Befehle, des virtuellen Speichers und der virtuellen Maschine. - Der Rezensent meint, daß auch dieses Buch in die Hand eines jeden gehört, der sich für die Hardware oder für die Assemblerprogrammierung interessiert. Man findet hier Hinweise, die den Datenblättern manchmal nur mühsam oder gar nicht zu entnehmen sind.

Highmore, D.; Page, L.: Der sensible C64. te-wi Verlag, München 1984, 130 S., ISBN 3-921 803-45 -4, DM 29,80. Dieses Buch eröffnet die Softwarenutzung aller technologischen Eigenheiten des beliebten C64: Tastatureingaben, benutzer-definierte Zeichen, Floppy Disks, Sprite-Grafiken, mehrfarbige Darstellungen usw. Die kurzen Programmsequenzen sollen zur Demonstration dienen, wie man es macht, um sie dann in entsprechender Form in eigene Programme zu übernehmen.

Valentine, R.: C-64 Programm-Sammlung. te-wi Verlag, München 1984, 200 S. ISBN 3-921 803-26 -2, DM 29,80. Das Buch enthält etwa 40 Lehr-, Spiel- und Nutzprogramme, die jeweils auch textlich erklärt sind.

Waite, M.; Morgan, Ch. L.: Grafik mit dem IBM PC. Bei McGraw-Hill, Hamburg 1985, 372 S., ISBN 3-89028-024-2, DM 69,50. Wir finden in diesem Buch folgende Haupt-Themen: Grafik—Hard- und Software des PC, detaillierte Hinweise zum Text- und Grafik-Modus, Techniken der 2D- und 3D-Darstellung, Grafiksprache GDL, Ausfüllen von Flächen, GET und PUT für Bewegungen, Definition eigener Zeichen, IBM-Grafiksplatine und spezielle Programmiertechniken. Das Buch enthält 86 verständlich erklärte Programme, die auch auf Diskette erhältlich sind und deren Bildschirmausgabe auf den farbig gedruckten Seiten des Anhanges dargestellt ist. Zusammen mit erklärenden Schema-Zeichnungen macht das Buch einen sehr soliden Eindruck, und es dürfte für Besitzer eines PC oder eines 'kompatiblen' Rechners sehr nützlich sein, wenn auch Grafik genutzt werden soll.

Engelen, W. von: Electron- und Acorn B-Anwenderhandbuch. Bei McGraw-Hill, Hamburg 1985, 296 S. ISBN 3-89028-023-4, DM 45,--. Für die bezeichneten BBC-Mikrocomputer dient das Buch mit mehr als 150 Programmen als Programmierbuch und andererseits auch als Lehrbuch, weil es mit zahlreichen Programmierverfahren und mit dem Befehlssatz, auch für Grafik und Tonerzeugung, bekannt macht.

Goodman, D.: Apple IIc Anwenderhandbuch. Bei McGraw-Hill, Hamburg 1985, 162 S., DM 39,80, ISBN 3-89028-027-7. Dieses Buch dient vor allem der Orientierung über die Leistungsfähigkeit dieses Computers in Hard- und Software. Es ist weniger ein Buch für die Hantierung oder Programmierung. Man sollte es studieren, wenn die Anschaffung eines entsprechenden Computermodells in Betracht gezogen wird oder wenn man es schon besitzt.

Hubertus Arndt, 8916 Unternühlhausen

Mitgliederverwaltung

Ein Programm z.B. für Vereine

Das nachfolgende Programm hat einen Umfang von 10 KB. Es ist auf einem AIM-65 mit 28 KB RAM nebst Floppy und Drucker geschrieben worden. BASIC-Dialekt ist das bekannte Microsoft-BASIC. Wie bei allen Programmen sind die rechnerspezifischen Ein- und Ausgaberoutinen die Ausnahme davon, welche die Daten auf Floppy und Drucker schreiben bzw. von der Floppy lesen. Die Adaption auf andere Rechner macht damit grundsätzlich keine Schwierigkeiten, wenn dort auch die nötige Speicherkapazität gewährleistet ist. Zur E/A weiter unten zusätzliche Hinweise.

Um dieses Programm benutzen zu können, müssen zuerst die Mitgliederdaten auf die Diskette geschrieben werden, da das Hauptprogramm so ausgelegt ist, daß bei einer Bearbeitung unter 'Menue' erst die Daten von der Floppy gelesen werden. Der Datenanlage dient das Formatierungsprogramm. Es bildet aus den im Dialog erfragten Daten einen String von 154 Zeichen je Mitglied. Aus der Aufgabenstellung ist klar, daß man die Grundideen der Programme auf ähnliche Aufgabenstellungen übertragen kann, seien es nun Adreß- oder ähnliche Bestandsdateien.

Ein Datenblock (Datensatz) hat folgende Felder:

Name und Vorname	max. 30 Zeichen
Geburtsdatum	10
Postleitzahl	4
Wohnort	20
Straße	20
Telefon	10
Bankleitzahl	10
Kontonummer	10
Geldinstitut	30
Einzugsermächtigung	4
Aktiv/passiv	4

Der Autor benutzt die Programme für einen Fußballverein mit ungefähr 200 Mitgliedern. Der Dateiname in Zeile 570 ist der verkürzte Vereinsname. - Dem Programm stehen 18 KB RAM zur Verfügung. In dieser Umgebung ist einzusehen, daß nicht alle Mitgliederdaten von A-Z im Speicher Platz haben können. Deswegen wurden sie in zwei Blöcke aufgeteilt, nämlich für A-M im File SFUA-M und entsprechend in File SUFN-Z für Mitgliedernamen von N-Z. Es ist klar, daß man bei Verwendung anderer Filenamen das in allen Programmteilen berücksichtigen muß.

Nach der Anlage dieser Dateien im Formatierungsprogramm hat der Anwender im Hauptprogramm der Verwaltung folgende Möglichkeiten:

Einmalige Neuaufnahme eines Mitgliedes. Es wird der komplette Datenblock erfragt und danach in das richtige File eingebunden und alphabetisch einsortiert.
Aufnahme mehrerer Mitglieder. Es werden Datenblöcke erfaßt und soviele Mitglieder angenommen, wie in die gleiche Kategorie des Filenames passen.

Löschen von Datensätzen, auch physisch auf der Diskette.

Die Suchroutine erlaubt die Eingabe einer beliebigen Buchstaben- oder Zahlenkombination, welche dann in beiden Files gesucht wird.

Datenblöcke können bearbeitet/verändert werden.

Ein Datenteil (z.B. Postleitzahl, Bankleitzahl) wird bei allen Mitgliedern verändert, welche das gleiche Merkmal in der Vorgängerdatei haben.

Die Druckroutine erzeugt eine formatierte alphabetische Mitgliederliste nebst Zahl der Mitglieder und Aufteilung auf aktive und passive.

Rückkehr in die BASIC-Eingaberoutine.

Besonderheiten betreffen die Ein- und Ausgaben. Hier wurde das GWK-BASIC benutzt. Die Eingaberoutine findet sich im Verwaltungsprogramm in den Zeilen 570 bis 600:

INDEVUF	entspricht dem Öffnen eines Files
B$	Variable für den Filename
1	Drive Nr.
INPUTLINE A$(I)	liest einen String von der Floppy
INDEVC	entspricht dem Schließen eines Files

Die Ausgaberoutine findet sich in den Zeilen 640 bis 700:

OUTDEVUF	entspricht dem Öffnen eines Files
B$	Variable für Filename
1	entspricht der Laufwerksnummer
OUTDEVUFC	entspricht dem Schließen eines Files

Weitere selbständige Programme arbeiten mit den Datenfiles, und zwar für Einzugsermächtigungen und die formatierte Ausgabe von Adreßaufklebern. Sie mögen das Programmpaket abrunden. Es gibt sicher andere Lösungsmöglichkeiten. Hier sollte für vergleichbare Aufgaben ein Paket als Anregung geschnürt werden.

FORMATIERUNGSPROGRAMM

(Anlage.der Datei)

```
 10 JSR$B215
 20 D=100
 30 DIMA$(D)
 40 PRINTCHR$(24)
100 Z=0
110 Z=Z+1
120 INPUT"NAME VORNAME";C$
130 INPUT"GEBURTSDATUM";D$
140 INPUT"POSTLEITZAHL";E$
150 INPUT"WOHNORT      ";F$
160 INPUT"STRASSE      ";G$
170 INPUT"TELEPHON     ";H$
180 INPUT"BANKLEITZAHL";I$
190 INPUT"KONTONUMMER  ";J$
200 INPUT"GELDINSTITUT";K$
210 INPUT"EINZUGSSERMAECHTIGUNG (JA/NEIN)";L$
220 INPUT"AKTIV/PASSIV";M$
300 B$=" "
310 IFLEN(C$)<30THENC$=C$+B$:GOTO310
320 IFLEN(D$)<10THEND$=D$+B$:GOTO320
330 IFLEN(E$)<4 THENE$=E$+B$:GOTO330
340 IFLEN(F$)<20THENF$=F$+B$:GOTO340
350 IFLEN(G$)<20THENG$=G$+B$:GOTO350
360 IFLEN(H$)<10THENH$=H$+B$:GOTO360
370 IFLEN(I$)<10THENI$=I$+B$:GOTO370
380 IFLEN(J$)<10THENJ$=J$+B$:GOTO380
390 IFLEN(K$)<30THENK$=K$+B$:GOTO390
400 IFLEN(L$)<4 THENL$=L$+B$:GOTO400
410 IFLEN(M$)<6 THENM$=M$+B$:GOTO410
500 A$(Z)=C$+D$+E$+F$+G$+H$+I$+J$+K$+L$+M$
510 PRINT" "
520 PRINT"  "
530 PRINT"NAECHSTES MITGLIED (J/N)?"
540 GETB$
550 PRINTCHR$(24)
560 IFB$="J"THEN110
565 PRINT"WELCHEN NAMEN SOLL DIE NEUE DATEI ERHALTEN?"
566 PRINT"  "
570 PRINT"SFUA-M      SFUN-Z  "
575 INPUTB$
580 OUTDEVUF,B$,1
590 FORA=1TOZ
600 PRINTA$(A)
610 NEXT
620 A$(A+1)="END"
630 PRINTA$(A+1)
640 OUTDEVUFC
650 PRINTCHR$(24)
660 PRINT"DATEN SIND ABGESPEICHERT WORDEN !!!!!!"
670 END
```

Hauptprogramm zur Verwaltung

```
5 REM FLOPPY UND DRUCKER INITIALISIEREN
10 POKE$9FBB,$00
20 JSR$8215
30 JSR$93DC
40 D=150
50 DIMA$(D),B(D)
55 REM LOESCHE BILDSCHIRM
60 PRINTCHR$(24)
65 REM MENUEAUSWAHL
```

```
70 PRINT"N=NEUAUFNAHME EINFACH"
80 PRINT"M=NEUAUFNAHME MEHRFACH"
90 PRINT"L=LOESCHEN"
100 PRINT"S=STICHWORTSUCHE"
110 PRINT"A=AENDERN EINZELN"
120 PRINT"G=AENDERN GESAMT"
130 PRINT"D=DRUCKEN"
140 PRINT"E=ENDE PROGRAMM"
150 GETB$
160 IFB$="N"THENB=1:GOTO250
170 IFB$="L"THENB=2:GOTO250
180 IFB$="S"THENB=3:GOTO250
190 IFB$="A"THENB=4:GOTO250
200 IFB$="D"THENB=5:GOTO250
210 IFB$="M"THENB=6:GOTO250
220 IFB$="G"THENB=7:GOTO250
230 IFB$="E"THENEND
240 GOTO70
250 ONBGOSUB270,730,1000,1370,2350,3020,3320
260 GOTO60
265 REM ROUTINE ZUR EINMALIGEN MITGLIEDERAUFNAHME
270 PRINTCHR$(24)
280 GOSUB300
290 GOTO540
300 INPUT"NAME VORNAME:";C$
310 INPUT"GEBURTSDATUM:";D$
320 INPUT"POSTLEITZAHL:";E$
330 INPUT"WOHNORT      :";F$
340 INPUT"STRASSE      :";G$
350 INPUT"TELEPHON     :";H$
360 INPUT"BANKLEITZAHL:";I$
370 INPUT"KONTONUMMER :";J$
380 INPUT"GELDINSTITUT:";K$
390 INPUT"EINZUGSERMAECHTIGUNG (JA/NEIN):";L$
400 INPUT"AKTIV/PASSIV:";M$
410 B$=" "
420 IFLEN(C$)<30THENC$=C$+B$:GOTO420
430 IFLEN(D$)<10THEND$=D$+B$:GOTO430
440 IFLEN(E$)<4THENE$=E$+B$:GOTO440
450 IFLEN(F$)<20THENF$=F$+B$:GOTO450
460 IFLEN(G$)<20THENG$=G$+B$:GOTO460
470 IFLEN(H$)<10THENH$=H$+B$:GOTO470
480 IFLEN(I$)<10THENI$=I$+B$:GOTO480
490 IFLEN(J$)<10THENJ$=J$+B$:GOTO490
500 IFLEN(K$)<30THENK$=K$+B$:GOTO500
510 IFLEN(L$)<4THENL$=L$+B$:GOTO510
520 IFLEN(M$)<6THENM$=M$+B$:GOTO520
530 RETURN
535 REM SFUA-M UND SFUN-Z WERDEN ALS FILENAME BENUTZT UND SIND
536 REM VOM ANWENDER FREI WAEHLBAR
540 IFLEFT$(C$,1)<="M"THENB$="SFUA-M"
550 IFLEFT$(C$,1)>="N"THENB$="SFUN-Z"
560 I=1
565 REM DIE NAECHSTE ZEILE OEFFNET DIE FLOPPY 1 UND LIEST DEN
566 REM FILE MIT DEM NAMEN IN B$
570 INDEVUF,B$,1
580 INPUTLINEA$(I)
590 IFA$(I)<>"END"THENI=I+1:GOTO580
600 INDEVC
610 A$(I)=C$+D$+E$+F$+G$+H$+I$+J$+K$+L$+M$
620 GOSUB4200
630 FORA=1TO1000:NEXT
640 OUTDEVUF,B$,1
650 FORA=1TOI
660 PRINTA$(A)
670 NEXT
680 A$(A+1)="END"
690 PRINTA$(A+1)
700 OUTDEVUFC
710 GOSUB4460
720 RETURN
725 REM ROUTINE ZUM LOESCHEN EINES MITGLIEDES
```

```
730 PRINTCHR$(24)
740 PRINT"WELCHES MITGLIED WOLLEN SIE LOESCHEN (NAME VORNAME)?"
750 PRINT"  "
760 INPUTC$
770 IFLEFT$(C$,1)<="M"THENB$="SFUA-M"
780 IFLEFT$(C$,1)>="N"THENB$="SFUN-Z"
790 I=1
800 INDEVUF,B$,1
810 INPUTLINEA$(I)
820 IFA$(I)<>"END"THENI=I+1:GOTO810
830 INDEVC
840 J=1:K=LEN(C$)
850 IFLEFT$(A$(J),K)=C$THEN880
860 IFA$(J)="END"THENPRINTCHR$(24):GOTO970
870 J=J+1:GOTO850
880 A$(J)=A$(J+1)
890 IFA$(J+1)="END"THEN910
900 J=J+1:GOTO880
910 OUTDEVUF,B$,1
920 FORA=1TOJ+1
930 PRINTA$(A)
940 NEXT
950 OUTDEVUFC
960 GOTO4460
970 PRINT"MITGLIED NICHT VORHANDEN!!"
980 FORI=1TO2000:NEXT
990 RETURN
995 REM ROUTINE ZUM SUCHEN NACH EINEM STICHWORT (NAME,POSTLEITZAHL USW.)
1000 PRINTCHR$(24)
1010 I=1:D$="SFUA-M"
1020 INDEVUF,D$,1
1030 INPUTLINEA$(I)
1040 IFA$(I)<>"END"THENI=I+1:GOTO1030
1050 INDEVC
1060 PRINTCHR$(24)
1070 J=0:INPUT"STICHWORT";B$:K=LEN(B$)
1080 J=J+1:IFA$(J)="END"THENPRINT"######################":J=0:GOTO1100
1090 GOTO1080
1100 J=J+1:IFLEFT$(A$(J),K)=B$THENZ=1:GOTO1200
1110 IFA$(J)<>"END"THEN1100
1120 J=0
1130 J=J+1:IFRIGHT$(A$(J),K)=B$THENZ=2:GOTO1200
1140 IFA$(J)<>"END"THEN1130
1150 J=0
1160 J=J+1
1161 Q=INSTR(B$,1,A$(J))
1162 IFQ=0GOTO1165
1163 Z=3:GOTO1200
1165 IFA$(J)<>"END"THEN1160
1166 GOTO1290
1200 OUTDEVD
1210 C$=" ":PRINT"  "
1220 PRINTLEFT$(A$(J),30)
1230 PRINTMID$(A$(J),31,10)
1240 PRINTMID$(A$(J),41,4);C$;MID$(A$(J),45,20);C$;MID$(A$(J),65,20)
1250 PRINTMID$(A$(J),85,10)
1260 PRINTMID$(A$(J),95,10);C$;MID$(A$(J),105,10);C$;MID$(A$(J),115,30)
1270 PRINTMID$(A$(J),145,4);C$;C$;RIGHT$(A$(J),6)
1280 OUTDEVC:ONZGOTO1100,1130,1160
1290 IFRIGHT$(D$,1)="Z"THENRETURN
1300 I=1:D$="SFUN-Z"
1310 INDEVUF,D$,1
1320 INPUTLINEA$(I)
1330 IFA$(I)<>"END"THENI=I+1:GOTO1320
1340 INDEVC
1350 PRINTCHR$(24)
1360 J=0:GOTO1100
1365 REM ROUTINE ZUM AENDERN DER DATEN EINES MITGLIEDES
1370 PRINTCHR$(24)
1380 PRINT"VON WELCHEM MITGLIED WOLLEN SIE DATEN AENDERN ( NAME VORNAME)?"
1390 PRINT"  "
1400 INPUTC$
```

```
1410 IFLEFT$(C$,1)<="M"THENF$="SFUA-M"
1420 IFLEFT$(C$,1)>="N"THENF$="SFUN-Z"
1430 I=1
1440 INDEVUF,F$,1
1450 INPUTLINEA$(I)
1460 IFA$(I)<>"END"THENI=I+1:GOTO1450
1470 INDEVC:PRINTCHR$(24)
1480 F=I-1
1490 J=0:K=LEN(C$)
1500 J=J+1:IFLEFT$(A$(J),K)=C$THEN1550
1510 IFA$(J)<>"END"THEN1500
1520 PRINT"MITGLIED NICHT VORHANDEN!!"
1530 FORI=1TO2000:NEXT
1540 RETURN
1550 PRINTCHR$(24)
1560 GOSUB1590
1570 GOSUB1670
1580 GOTO1810
1590 C$=" ":PRINTCHR$(24)
1600 PRINTLEFT$(A$(J),30)
1610 PRINTMID$(A$(J),31,10)
1620 PRINTMID$(A$(J),41,4);C$;MID$(A$(J),45,20);C$;MID$(A$(J),65,20)
1630 PRINTMID$(A$(J),85,10)
1640 PRINTMID$(A$(J),95,10);C$;MID$(A$(J),105,10);C$;MID$(A$(J),115,30)
1650 PRINTMID$(A$(J),145,4);C$;C$;RIGHT$(A$(J),6)
1660 RETURN
1670 PRINT"  "
1680 PRINT"WAS WOLLEN SIE AENDERN: N=NAME VORNAME"
1690 PRINT"                        G=GEBURTSDATUM"
1700 PRINT"                        P=POSTLEITZAHL"
1710 PRINT"                        W=WOHNORT"
1720 PRINT"                        S=STRASSE"
1730 PRINT"                        T=TELEPHON"
1740 PRINT"                        B=BANKLEITZAHL"
1750 PRINT"                        K=KONTONUMMER"
1760 PRINT"                        I=GELDINSTITUT"
1770 PRINT"                        E=EINZUGSERMAECHTIGUNG"
1780 PRINT"                        A=AKTIV/PASSIV"
1790 PRINT"  "
1800 RETURN
1810 GETB$
1820 C$=" "
1830 IFB$="N"THENPRINTCHR$(24):GOTO1950
1840 IFB$="G"THENPRINTCHR$(24):GOTO2010
1850 IFB$="P"THENPRINTCHR$(24):GOTO2030
1860 IFB$="W"THENPRINTCHR$(24):GOTO2050
1870 IFB$="S"THENPRINTCHR$(24):GOTO2070
1880 IFB$="T"THENPRINTCHR$(24):GOTO2090
1890 IFB$="B"THENPRINTCHR$(24):GOTO2110
1900 IFB$="K"THENPRINTCHR$(24):GOTO2130
1910 IFB$="I"THENPRINTCHR$(24):GOTO2150
1920 IFB$="E"THENPRINTCHR$(24):GOTO2170
1930 IFB$="A"THENPRINTCHR$(24):GOTO2190
1940 GOTO1810
1950 PRINT"ALTE DATEN: ";LEFT$(A$(J),30)
1960 PRINT"  "
1970 INPUT"NEUE DATEN";B$
1980 IFLEN(B$)<30THENB$=B$+C$:GOTO1980
1990 A$(J)=B$+RIGHT$(A$(J),124)
2000 GOTO2310
2010 A=31:B=10:C=30:D=114:GOSUB2250
2020 GOTO2310
2030 A=41:B=4:C=40:D=110:GOSUB2250
2040 GOTO2310
2050 A=45:B=20:C=44:D=90:GOSUB2250
2060 GOTO2310
2070 A=65:B=20:C=64:D=70:GOSUB2250
2080 GOTO2310
2090 A=85:B=10:C=84:D=60:GOSUB2250
2100 GOTO2310
2110 A=95:B=10:C=94:D=50:GOSUB2250
2120 GOTO2310
```

```
2130 A=105:B=10:C=104:D=40:GOSUB2250
2140 GOTO2310
2150 A=115:B=30:C=114:D=10:GOSUB2250
2160 GOTO2310
2170 A=145:B=4:C=144:D=6:GOSUB2250
2180 GOTO2310
2190 PRINT"ALTE DATEN:" ";RIGHT$(A$(J),6)
2200 PRINT"  "
2210 INPUT"NEUE DATEN";B$
2220 IFLEN(B$)<6THENB$=B$+C$:GOTO2220
2230 A$(J)=LEFT$(A$(J),148)+B$
2240 GOTO2310
2250 PRINT"ALTE DATEN: ";MID$(A$(J),A,B)
2260 PRINT"  "
2270 INPUT"NEUE DATEN";B$
2280 IFLEN(B$)<BTHENB$=B$+C$:GOTO2280
2290 A$(J)=LEFT$(A$(J),C)+B$+RIGHT$(A$(J),D)
2300 RETURN
2310 GOSUB1590
2315 FORV=1TO500:NEXT
2320 I=F
2330 B$=F$
2332 PRINTCHR$(24)
2334 PRINT"WENN SIE IN DER DATEI ";B$;" WEITERE AENDERUNGEN VORNEHMEN WOLLEN,"
2336 PRINT"GEBEN SIE 'JA' EIN, SONST GEBEN SIE 'NEIN' EIN."
2338 INPUTV$
2340 IFV$<>"JA"THEN620
2342 PRINT" ":PRINT"  "
2344 PRINT"DER NAME DES NAECHSTEN MITGLIEDES LAUTET:"
2346 INPUTC$
2348 GOTO1490
2349 REM ROUTINE ZUM AUSDRUCKEN EINER MITGLIEDERLISTE
2350 PRINTCHR$(24)
2360 INPUT"HEUTIGES DATUM ";B$
2370 C$="SFUA-M"
2380 GOSUB2790
2390 OUTDEVUP
2400 PRINTCHR$($0E);"INT. SFU-MITGLIEDERLISTE VOM ";B$
2410 PRINT"  "
2420 PRINT"  "
2430 PRINT"  "
2440 PRINT"  "
2450 OUTDEVC
2460 C$="AKTIV "
2470 GOSUB2850
2480 J=G
2490 C$="SFUN-Z"
2500 GOSUB2790
2510 C$="AKTIV "
2520 GOSUB2850
2530 K=G
2540 C$="SFUA-M"
2550 GOSUB2790
2560 E=I-1
2570 C$="PASSIV"
2580 GOSUB2850
2590 L=G
2600 C$="SFUN-Z"
2610 GOSUB2790
2620 F=I-1
2630 C$="PASSIV"
2640 GOSUB2850
2650 M=G
2660 OUTDEVUP
2670 PRINT"  "
2680 PRINT"  "
2690 PRINT"  "
2700 PRINT"  "
2710 PRINT"  "
2720 PRINTCHR$($0E);"GESAMTMITGLIEDERZAHL : ";E+F
2730 PRINT"  "
2740 PRINTCHR$($0E);"AKTIVE MITGLIEDER    : ";J+K
```

```
2750 PRINT"   "
2760 PRINTCHR$(&0E);"PASSIVE MITGLIEDER   : ";L+M
2770 OUTDEVC
2780 RETURN
2790 I=1
2800 INDEVUF,C$,1
2810 INPUTLINEA$(I)
2820 IFA$(I)<>"END"THENI=I+1:GOTO2810
2830 INDEVC:PRINTCHR$(17)
2840 RETURN
2850 OUTDEVUP
2860 G=0
2870 FORA=1TOI-1
2880 IFRIGHT$(A$(A),6)<>C$THEN2990
2890 PRINTLEFT$(A$(A),30);SPC(4);MID$(A$(A),31,10);SPC(21);RIGHT$(A$(A),6)
2900 PRINT"   "
2910 PRINTMID$(A$(A),41,4);SPC(5);MID$(A$(A),45,20);SPC(5);MID$(A$(A),65,20);
2915 PRINT SPC(11);MID$(A$(A),85,10)
2920 PRINT"   "
2930 PRINTMID$(A$(A),95,10);SPC(5);MID$(A$(A),105,10);SPC(9);
2935 PRINT MID$(A$(A),115,30);SPC(1);MID$(A$(A),145,4)
2940 PRINT"   "
2950 PRINT"   "
2960 PRINT"   "
2970 PRINT"   "
2980 G=G+1
2990 NEXT
3000 OUTDEVC
3010 RETURN
3015 REM ROUTINE ZUR MEHRMALIGEN MITGLIEDERAUFNAHME
3020 PRINTCHR$(24)
3030 PRINT"ZU WELCHER TEILDATEI GEHOEREN DIE NEUEN MITGLIEDER?"
3040 PRINT"SFUA-M=A UND SFUN-Z=N"
3050 GETB$
3060 IFB$="A"THEND$="SFUA-M":GOTO3080
3070 D$="SFUN-Z"
3080 I=1
3090 INDEVUF,D$,1
3100 INPUTLINEA$(I)
3110 IFA$(I)<>"END"THENI=I+1:GOTO3100
3120 INDEVC
3130 PRINTCHR$(24)
3140 GOSUB300
3150 A$(I)=C$+D$+E$+F$+G$+H$+I$+J$+K$+L$+M$
3160 PRINT"NAECHSTES MITGLIED (Y/N)?"
3170 GETB$
3180 IFB$<>"Y"THEN3200
3190 I=I+1:GOTO3140
3200 GOSUB4200
3210 FORA=1TO1000:NEXT
3220 OUTDEVUF,D$,1
3230 FORA=1TOI
3240 PRINTA$(A)
3250 NEXT
3260 A$(A+1)="END"
3270 PRINTA$(A+1)
3280 OUTDEVUFC
3290 B$=D$
3300 GOSUB4460
3310 RETURN
3315 REM ROUTINE, WELCHE EINE AENDERUNG BEI JEDEM MITGLIED VORNIMMT
3316 REM BEI WELCHEM DER ZUAENDERNDE WERT ENTHALTEN IST
3320 PRINTCHR$(24)
3330 GOSUB1670
3340 GETB$
3350 IFB$="N"THENA=1:GOTO3450
3360 IFB$="G"THENA=2:GOTO3450
3370 IFB$="P"THENA=3:GOTO3450
3380 IFB$="W"THENA=4:GOTO3450
3390 IFB$="S"THENA=5:GOTO3450
3400 IFB$="T"THENA=6:GOTO3450
3410 IFB$="B"THENA=7:GOTO3450
```

```
3420 IFB$="K"THENA=8:GOTO3450
3430 IFB$="I"THENA=9:GOTO3450
3440 GOTO3340
3450 PRINTCHR$(24)
3460 PRINT"WIE LAUTEN DIE ALTEN DATEN?"
3470 B$=" "
3480 INPUTH$
3490 IFA=1THENB=30:GOTO3580
3500 IFA=2THENB=10:GOTO3580
3510 IFA=3THENB=4:GOTO3580
3520 IFA=4THENB=20:GOTO3580
3530 IFA=5THENB=20:GOTO3580
3540 IFA=6THENB=10:GOTO3580
3550 IFA=7THENB=10:GOTO3580
3560 IFA=8THENB=10:GOTO3580
3570 IFA=9THENB=30:GOTO3580
3580 IFLEN(H$)<BTHENH$=H$+B$:GOTO3580
3590 PRINT"WIE LAUTEN DIE NEUEN DATEN?"
3600 INPUTC$
3610 IFLEN(C$)<BTHENC$=C$+B$:GOTO3610
3620 E$="SFUA-M"
3630 I=1
3640 INDEVUF,E$,1
3650 INPUTLINEA$(I)
3660 IFA$(I)<>"END"THENI=I+1:GOTO3650
3670 INDEVC
3680 ONAGOTO3690,3730,3770,3810,3850,3890,3930,3970,4010
3690 FORD=1TOI
3700 IFLEFT$(A$(D),30)=H$THENA$(D)=C$+RIGHT$(A$(D),124)
3710 NEXT
3720 GOTO4050
3730 FORD=1TOI
3740 IFMID$(A$(D),31,10)=H$THENA$(D)=LEFT$(A$(D),30)+C$+RIGHT$(A$(D),114)
3750 NEXT
3760 GOTO4050
3770 FORD=1TOI
3780 IFMID$(A$(D),41,4)=H$THENA$(D)=LEFT$(A$(D),40)+C$+RIGHT$(A$(D),110)
3790 NEXT
3800 GOTO4050
3810 FORD=1TOI
3820 IFMID$(A$(D),45,20)=H$THENA$(D)=LEFT$(A$(D),44)+C$+RIGHT$(A$(D),90)
3830 NEXT
3840 GOTO4050
3850 FORD=1TOI
3860 IFMID$(A$(D),65,20)=H$THENA$(D)=LEFT$(A$(D),64)+C$+RIGHT$(A$(D),70)
3870 NEXT
3880 GOTO4050
3890 FORD=1TOI
3900 IFMID$(A$(D),85,10)=H$THENA$(D)=LEFT$(A$(D),84)+C$+RIGHT$(A$(D),60)
3910 NEXT
3920 GOTO4050
3930 FORD=1TOI
3940 IFMID$(A$(D),95,10)=H$THENA$(D)=LEFT$(A$(D),94)+C$+RIGHT$(A$(D),50)
3950 NEXT
3960 GOTO4050
3970 FORD=1TOI
3980 IFMID$(A$(D),105,10)=H$THENA$(D)=LEFT$(A$(D),104)+C$+RIGHT$(A$(D),40)
3990 NEXT
4000 GOTO4050
4010 FORD=1TOI
4020 IFMID$(A$(D),115,30)=H$THENA$(D)=LEFT$(A$(D),114)+C$+RIGHT$(A$(D),10)
4030 NEXT
4040 GOTO4050
4050 I=I-1
4060 GOSUB4200
4070 FORF=1TO8000:NEXT
4080 OUTDEVUF,E$,1
4090 FORG=1TOI
4100 PRINTA$(G)
4110 NEXT
4120 A$(G+1)="END"
```

```
4130 PRINTA$(G+1)
4140 OUTDEVUFC
4150 B$=E$
4160 GOSUB4460
4170 FORZ=1TO5000:NEXT
4180 IFE$="SFUA-M"THENE$="SFUN-Z":GOTO3630
4190 RETURN
4195 REM SORTIERROUTINE
4200 L=1:N=I:B(L)=N+1:M=1
4210 J=B(L)
4220 F=M-1
4230 IFJ-M<3THEN4390
4240 M1=INT((F+J)/2)
4250 F=F+1
4260 IFF=JTHEN4330
4270 IFA$(F)<=A$(M1)THEN4250
4280 J=J-1
4290 IFF=JTHEN4330
4300 IFA$(J)>=A$(M1)THEN4280
4310 A$(0)=A$(F):A$(F)=A$(J):A$(J)=A$(0)
4320 GOTO4250
4330 IFF>=M1THENF=F-1
4340 IFJ=M1THEN4360
4350 A$(0)=A$(F):A$(F)=A$(M1):A$(M1)=A$(0)
4360 L=L+1
4370 B(L)=F
4380 GOTO4210
4390 IFJ-M<2THEN4420
4400 IFA$(M)<A$(M+1)THEN4420
4410 A$(0)=A$(M):A$(M)=A$(M+1):A$(M+1)=A$(0)
4420 M=B(L)+1
4430 L=L-1
4440 IFL>0THEN4210
4450 RETURN
4455 REM ROUTINE UM EINE FILE PHYSISCH VON DER FLOPPY ZU LOESCHEN
4460 POKE$9E00,1
4470 FORD=1TOLEN(B$)
4480 N$=MID$(B$,D,1)
4490 F(D)=ASC(N$)
4500 POKE$9E00+D,F(D)
4510 NEXT
4520 JSR$BB89
4530 POKE$9E00,1
4540 JSR$BBEC
4550 RETURN

1 REM INITIALISIEREN VON DRUCKER UND FLOPPY
10 POKE$9FBB,$00
20 JSR$8215                    (Drucken von Adreßaufklebern)
30 JSR$93DC
40 D=100
50 DIMA$(D)
60 PRINTCHR$(24)
65 PRINT"SIND DIE ADRESSAUFKLEBER SCHON IM DRUCKER EINGELEGT J/N ?"
66 GETA$
67 IFA$="N"THEN65
70 C$="SFUA-M":GOSUB110
80 C$="SFUN-Z":GOSUB110
100 END
110 INDEVUF,C$,1
120 A=1
130 INPUTLINEA$(A)
140 IFA$(A)<>"END"THENA=A+1:GOTO130
150 INDEVC
160 OUTDEVUP
170 FORI=1TOA-1
180 PRINT"HR./FR./FRL."
190 PRINT"   "
200 PRINTLEFT$(A$(I),23)
210 PRINT"   "
220 PRINTMID$(A$(I),65,20)
230 PRINTMID$(A$(I),41,4);SPC(1);MID$(A$(1),45,20)
240 PRINT"   "
250 PRINT"   "
260 PRINT"   "
270 NEXT
280 OUTDEVC
290 RETURN
```

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

Neu: Assembler für 65802/816, für R65C02 und für 6502

2 Pass Assembler mit einstellbaren Befehlssätzen für jede CPU zur Vermeidung von Irrtümern. Geschrieben in 6502-Code, damit auf vorhandenen Systemen einsetzbar. Generierbar für beliebige Adreßbereiche. Standardmäßig lieferbar für AIM 65 und kompatible (für SYSTEM 65 a. Anfr.). Für andere Fälle dokumentierte Sprungleiste auf externe Routinen des E/A-Systems. Die Syntax folgt dem Rockwell-Assembler. Symboltafel alphabet. geordnet mit Zeilenangabe, wo definiert. Formatierte Assembler-Liste mit 80 Zch/Z, wie in Heft 41 dokumentiert. Kommandofiles möglich, die Quelltexte von verschiedenen Eingabemedien verbinden. Damit zügige Durchläufe mit Library-Funktion. Assemblerung mit Offset in beliebige Bereiche als Vorlage für EPROMs. Für AIM 65, inkl. Dokumentation, Preis inkl. MWSt DM 300,--

Cross-Assembler für 6804

in Vorbereitung. Lieferbar etwa Ende März 1985. Bitte anfragen

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

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) 712 40 18

MICRO MAG

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

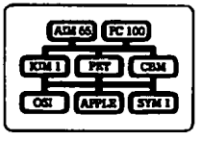
MICRO MAG (vormals 65xx MICRO MAG) erscheint zweimonatlich, jeweils Mitte Februar, April usw.. COPYRIGHT 1984 by Roland Löhr. Alle Rechte vorbehalten, auch die des auszugsweisen Nachdruckes, der Übersetzung, der fotomechnischen Wiedergabe und die der Verbreitung auf magnetischen und sontigen Trägern. Von den veröffentlichten Programmen, Schaltungen und Angaben wird ohne eine Gewährleistung von hier aus angenommen, daß sie fehlerfrei und frei von den Schutzrechten Dritter sind. Beiträge, die nicht besonders gekennzeichnet sind, stammen vom Herausgeber. Offsetdruck: L & L Druckservice, Hamburg 73

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 und 16 der Zeitschrift. Es erfolgt keine Neuauflage.

Lieferbar: 'Das Buch 7-13 des 65xx MICRO MAG' zu DM 42,– sowie die Hefte 14, 15 und 17-41 zu DM 7,80/St. (ab 10 St. in einer Sendung: DM 6,–/St.).

FORTH User's Manual (Lieferung, solange vorrätig)
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.