65xx MICRO MAG

COMPUTING · SOFTWARE · HOBBY

DM 8,50 Nr. 22 Dezember 1981

Inhaltsverzeichnis

Sortieren mit dem AIM 3
LINED (AIM) 11
Prozeßtechnik mit Mikro-Computern (2) 22
Binär-BCD-Wandlung 27
Druckerausgabe auf parallele Schnittstelle 30
6505 Multiplikation, Division 31
Mischbilder vom PET und einer Video-Kamera 38
Berechnung von Pi mit großer Genauigkeit (CBM) 41
Adressierungsarten des 6809 42
CBM-Math 50
AIM Spezial (11) 54
Buchbesprechungen 55
Breite Monitorausgabe für CBM 8032 56
Editorial 56
BASIC-Formatierungen 57
Disk-APPEND 57
Jahresinhaltsverzeichnis ..... Heftmitte

Michael Zimmermann, 6102 Pfungstadt

Sortieren mit dem AIM

Für die Prozessoren der 65xx-Serie wurde bereits eine Reihe von Sortierprogrammen entwickelt (2), (3), (4). Ein Teil von Ihnen vollzieht die Sortierung ausschließlich in Maschinensprache (2),(3). Ein anderer Teil beruht mehr oder weniger auf BASIC (4) und ist damit an die Datenstrukturen dieser Sprache gebunden. Auf reine BASIC-Sortierprogramme soll hier nicht eingegangen werden, ihre Anzahl ist Legion. - Den bis jetzt veröffentlichten Programmen ist aber gemeinsam, daß die Sortierung ohne Benutzung externer Datenträger nur im Kernspeicher durchgeführt wird.

Mit dem vorliegenden Programm sollen die erweiterten Speichermöglichkeiten externer Datenträger dargestellt werden, insbesondere die der Audio-Cassetten. Dabei ist es ein besonderer Vorteil, daß die Kapazität nicht durch die Kapazität des Hauptspeichers beschränkt wird, sondern nahezu unbegrenzt ist, auch wenn dieses möglicherweise mit mehreren Durchläufen bei der Sortierung erkauft wird. Über den hier gewählten Betrieb mit Audio-Cassetten hinaus ist es selbstverständlich möglich, das dargestellten Konzept auch auf andere Speichermedien zu übertragen, z.B. auf Disketten. An geeigneter Stelle wird auf die erforderlichen Modifikationen, die nur die Ein- und Ausgabe betreffen, hingewiesen.

Bei der Erstellung eines allgemeinen Sortierprogrammes ist es eine naheliegende Idee, einmal nachzusehen, welche Verfahren bereits beschrieben sind und wie sie auf den vorliegenden Fall übertragen werden können. Die Literatur (1) zeigt eine Vielzahl von Verfahren auf, leider wird aber kein allgemeingültiger Ansatz für ein Sortieren mit nur zwei Laufwerken gegeben. Von dieser Konstellation für den Standard-AIM wollen wir uns bei dem zu beschreibenden Verfahren nicht lösen. Man muß also zur Selbsthilfe greifen und die Sortierung, vom Verfahren beginnend, selber entwickeln.

Wir wollen von Datensätzen fester Länge mit gleichbleibender Satzeinteilung ausgehen, d.h. der Sortierbegriff steht in jedem Satz der Datei immer an der gleichen Stelle relativ zum Satzanfang. Als maximale Satzlänge sollen 80 Byte entsprechend dem AIM-Format gewählt werden.

Auch hier sind Erweiterungen auf eine größere Satzlänge oder Benutzung eines variablen Formates möglich, die erforderlichen Änderungen werden an gegebener Stelle angedeutet. Eine Erweiterung über 255 Stellen hinaus dürfte größere Schwierigkeiten machen.

Für die Ein- und Ausgabe werden die entsprechenden ASP-Routinen verwendet (6), lediglich diese sind bei einem anderen Speichermedium durch entsprechende benutzereigene Programmteile zu ersetzen.

Das entwickelte Sortierverfahren ähnelt dem bubble-up-sort. Der Ablauf soll im folgenden beschrieben werden. Ausgangspunkt der Sortierung ist ein kontinuierlicher Sortierpuffer im Hauptspeicher. Dieser ist unterteilt in eine Reihe von Rastern, die jeweils die Länge der zu sortierenden Datensätze haben. Folgende klar voneinander getrennte Phasen sind für die Sortierung von Wichtigkeit:

1. Parametrierung
2. Initialisierung
3. Sortierphase
   3.1 Eingabephase
   3.2 Auswahlphase
   3.3 Ausgabephase
4. Resultat

1. Parametrierung

In der ersten Phase, der Parametrierung, werden die Ein- und Ausgabedatei, die Anfangsadresse des Sortierpuffers, die Anzahl und die Länge der Raster sowie die Positionen der Bytes des Sortierbegriffes festgelegt. Eine variable Zuordnung dient hier einer größtmöglichen Flexibilität des Sortierprogrammes selbst.

2. Die Initialisierung

In der Initialisierung wird Satz für Satz von der Eingabedatei gelesen und im Sortierpuffer abgelegt. Wenn die maximale Rasterzahl gefüllt ist, so wird in der Sortierphase mit der Auswahl fortgefahren.

Wird ein Ende der Eingabedatei erkannt, so hat der Bediener die Möglichkeit, eine weitere Eingabedatei zu spezifizieren. Wird dies nicht gewünscht, so wird der Rest des Sortierpuffers mit hex FF aufgefüllt, und es schließt sich die Auswahlphase an. - Grundsätzlich ist eine beliebige Anzahl von Eingabedateien verarbeitbar, aber nur eine Ausgabedatei.

3. Sortierphase

Die Sortierphase gliedert sich, wie bereits dargestellt, in die Eingabe-, Auswahl- und Ausgabephase.

3.1 Eingabephase

In der Eingabephase wird ein Satz von der Eingabedatei gelesen und in ein Rasterfeld des Sortierpuffers gestellt. Bei diesem Raster handelt es sich um jenes, aus dem in der vorausgegangenen Ausgabephase ein Datensatz herausgeschrieben wurde, so daß das Feld jetzt wieder verfügbar ist. Hierbei ist zu beachten, daß die Eingabephase unmittelbar nach der Initialisierung nicht angesprochen wird und wegen der dann bestehenden Füllung des Sortierpuffers auch gar nicht benutzt werden darf.

Wird bei einem Leseversuch ein Dateiende erkannt, so wird eine weitere Eingabedatei angefordert und ggfs. von dieser ein Satz eingelesen. Wird stattdessen die Eingabe vom Bediener beendet, so wird ein entsprechender Schalter gesetzt und für die anstehende sowie für alle nachfolgenden Datensatzanforderungen ein hex FF in das entsprechende Sortierraster eingespeichert.

3.2 Auswahlphase

In der Auswahlphase wird das Raster mit dem niedrigsten Sortierbegriff bestimmt und seine Rasternummer den folgenden Phasen der Ausgabe und der Eingabe zur Verfügung gestellt.

Gleichzeitig wird überprüft, ob durch diesen Satz die aufsteigende Folge der Ausgabedatei nicht zerstört wird. Liegt eine Störung vor, so wird für die Resultatphase ein Schalter gesetzt. - Ist der Sortierbegriff hex 'FF', so ist die Sortierung abgeschlossen. Dieser Satz wird natürlich nicht ausgegeben, vielmehr wird gleich zur Resultatphase verzweigt.

4. Resultatphase

In dieser Phase wird auf Grund der Schalterstellung überprüft, ob die Ausgabedatei mit aufsteigender Sortierfolge geschrieben wurde. Ist dies der Fall, so ist die Sortierung zu beenden. Ansonsten ist die gesamte Sortierung zu wiederholen, die Ausgabedatei dient dabei als Eingabedatei.

Aus der Beschreibung des Sortiervorganges erkennt man unschwer, daß alle speicherresidenten Vorgänge mit einem Minimum an Umspeicherung bewirkt werden. Insbesondere wurde kein Wert darauf gelegt, die Raster in eine physisch aufsteigende Folge zu bringen, ein Tatbestand, der sich vorteilhaft auf die Geschwindigkeit auswirkt. Eine möglichst kurze Sortierung wird erreicht, wenn ein einziger Sortiervorgang für eine ganze Datei ausreicht. Das wird erreicht, wenn die komplette Datei im Sortierpuffer Platz findet oder wenn zumindest der niedrigste Sortierbegriff bereits in der Initialisierungsphase eingelesen wird.

Ebenso ist ein Zusammenmischen mehrerer Eingabedateien möglich, wobei sinnvollerweise der kleinere Datenbestand als erster eingelesen wird.

Das eigentliche Sortierprogramm bedarf keiner weitergehenden Beschreibung. Der Ablauf entspricht dem dargestellten Verfahren und ist im Quellentext reichlich dokumentiert. Bei Änderung des Eingabemediums sind die Ein- und Ausgabe-Unterprogramme, die hier aus dem ASP entnommen wurden, vom Benutzer selber zu gestalten. Das gilt auch für vom Benutzer gewünschte variable oder größere Satzlängen. Die Rasterlänge ist dann auf den größtmöglichen Satz zu normieren. Ebenso muß das Feld LAST, das den letzten ausgegebenen Satz für Vergleichszwecke hält, ange-

paßt werden. Werden mehr als 16 Bytes für die Länge des Sortierbegriffes gewünscht, so ist SORTF entsprechend anzupassen.

Das Programm ist interaktiv gestaltet, alle erforderlichen Werte werden vom Benutzer abgefragt und sind, soweit numerisch, als Hexa-Werte einzugeben.

Literaturhinweise

(1) Wedekind, 'Dateiorganisation', Berlin 1970
(2) Diverse, 'The First Book of KIM', Ostfildern 2, 1977, Seite 136
(3) Löhr, 'ALPHA-SORT, Flexibles Sortierprogramm', 65xx MICRO MAG Nr. 4, 1978
(4) Quindt, 'Ein Sortierprogramm für CBM 3001', 65xx MICRO MAG Nr. 16, 1980
(5) Zimmermann, 'ASP – Advanced Subroutine Package', 65xx MICRO MAG, Hefte Nr. 2 ff.

```
0000            ;--------------------
0000            ;SORT-PROGRAMM MIT EXTERNEN DATENTRAEGERN
0000            ;
0000            ;--------------------
0000            ;RAM-ADRESSEN PAGE ZERO
0000            ;
0000                    *=$0
0000 LAST               *=*+80          ;LETZTER SATZ DER AUSGABE
0050                    *=$80
0080 SORTF              *=*+$10         ;SORTIERFELD
0090 RAZA               *=*+1           ;RASTERZAHL
0091 RALE               *=*+1           ;RASTERLAENGE
0092 ANF                *=*+2           ;ANFANG SORTIERPUFFER  PUFFER
0094 AKT                *=*+2           ;AKTUELLE ADRESSE IN SORTIER
0096 MIN                *=*+2           ;ADRESSE KLEINSTES RASTER
0098 ARAZA              =$9D            ;AKTUELLE RASTERZAHL
0098 ENDFL              =$9E            ;ENDFLAG
0098 REDOFL             =$9F            ;REDO-FLAG
0098            ;--------------------
0098            ;AIM-ROM-ADRESSEN
0098            ;
0098 RSET               =$E0BF
0098 FROM               =$E7A3
0098 QM                 =$E7D4
0098 BLANK              =$E83E
0098 CRLOW              =$EA13
0098 RD2                =$EA5D
0098 OUTDP              =$EEFC
0098 TIBYTE             =$ED3B
0098            ;--------------------
0098            ;UNTERPROGRAMM-ADRESSEN DES SUBROUTINE PACKAGE
0098            ;
0098 WRITE              =$9C8C
0098 OPI                =$9CB2
0098 READ               =$9CB8
0098 OPO                =$9C86
0098 CLO                =$9C9F
0098            ;--------------------
0098            ;AIM-RAM-ADRESSEN
0098            ;
0098 ADDR               =$A41C
0098 INFLG              =$A412
0098 TAPTR              =$A436
0098 TAPTR2             =$A437
```

```
0098            ;---------------------
0098            ;ROUTINEN
0098            ;
0098                    *=$200
0200            20D702  JSR BEG         ;INITIALISIEREN
0203            D003    BNE MA20        ;UND MIT SORT WEITERMACHEN
0205 MA10
0205            206302  JSR INPUT       ;EINEN SATZ LESEN
0208 MA20                                                   FINDEN
0208            202202  JSR SORT        ;SORTIERN D.H.MINIMUM-SATZ
020B            209A02  JSR OUTPUT      ;UND MINIMUM-SATZ AUSGEBEN
020E            E000    CPX #00         ;WURDE LETZTER SATZ AUSGEGEBEN
0210            F0F3    BEQ MA10        ;NEIN - WEITER SORTIEREN
0212            249F    BIT REDOFL      ;SORTIERERGEBNISS PRUEFEN
0214            3004    BMI MA90
0216            A236    LDX #MSG08-MSG01    ;SORT RICHTIG SETZEN
0218            D002    BNE MA95
021A MA90                               ;SORT-WIEDERHOLUNG SETZEN
021A            A241    LDX #MSG09-MSG01
021C MA95
021C            206203  JSR PRT         ;ENDEMELDUNG DRUCKEN
021F            4CBFE0  JMP RSET        ;ZURUECK ZUM MONITOR
0222            ;---------------------
0222            ;SORTIEREN IN SPEICHER
0222            ;
0222 SORT
0222            A592    LDA ANF         ;PUFFERANFANG SETZEN IN
0224            8594    STA AKT         ;AKTUELLES RASTER
0226            8596    STA MIN         ;UND MINIMUM-RASTER
0228            A593    LDA ANF+1
022A            8595    STA AKT+1
022C            8597    STA MIN+1
022E            A590    LDA RAZA        ;AKTUELLE RASTERZAHL LADEN
0230            859D    STA ARAZA
0232            ;--------------------
0232            ;
0232 S010
0232            205502  JSR INCR        ;AKTUELLES RASTER ERHOEHEN
0235            D001    BNE S020        ;UND AUF PUFFERENDE PRUEFEN
0237            60      RTS             ;BEI PUFFERENDE SORTIERDURCHLAUF
0238                                                        FERTIG
0238            ;---------------------
0238            ;VERGLEICHEN
0238            ;
0238 S020
0238            38      SEC
0239            ;ANFANGSBEDINGUNG FUER SBC/VERGLEICH SETZEN
0239            A20F    LDX #$0F        ;LAENGE DES SORTIERFELDES SETZEN
023B            ;--------------------
023B            ;
023B S030                                                   HOLEN
023B            B480    LDY SORTF,X     ;RASTERPOSITION AUS SORTIERFELD
023D            C0FF    CPY #$FF        ;UNINTERESSANTE POSITION
023F            F004    BEQ S040        ;KEIN VERGLEICH
0241            B196    LDA (MIN),Y     ;VERGLEICH EINER POSITION IN
0243            F194    SBC (AKT),Y     ;MINIMUM UND AKTUELLEM RASTER
0245 S040
0245            CA      DEX             ;VERMINDERN SORTIERFELDPOINTER
0246            10F3    BPL S030
0248            ;SORTIERFELD NOCH NICHT DURCH WEITER VERGLEICHEN
```

```
0248          90E8    BCC SO10
024A          ;AKTUELLES RASTER GROESSER MINIMUM - WEITERMACHEN
024A          ---------------------
024A          KLEINER GEFUNDEN
024A
024A          A594    LDA AKT          ;AKTUELLES RASTER UMSETZEN
024C          8596    STA MIN          ;IN MINIMUM
024E          A595    LDA AKT+1
0250          8597    STA MIN+1
0252          18      CLC
0253          90DD    BCC SO10
0255          ---------------------
0255          INCREMENT AKTUELLEN POINTER
0255
0255 INCR
0255          18      CLC
0256          A594    LDA AKT          ;ZUM AKTUELLEN RASTER
0258          6591    ADC RALE         ;RASTERLAENGE ADDIEREN
025A          8594    STA AKT
025C          9002    BCC INC90
025E          E695    INC AKT+1
0260 INC90
0260          C69D    DEC ARAZA        ;AKTUELLE RASTERZAHL VERMINDERN
0262          60      RTS
0263          ---------------------
0263          EINGABE EINES SATZES IN DEN PUFFER
0263
0263 INPUT
0263          A000    LDY #0           ;POINTER AUF RASTERANFANG SETZEN
0265          249E    BIT ENDFL        ;TEST ENDE DER EINGABEPHASE
0267          3027    BMI IN80         ;EINGABE ZU END - BLOCK LOESCHEN
0269          A591    LDA RALE
026B          ;LESE-UNTERPROGRAMM MIT PARAMETERN LAENGE UND
026B          8D7B02  STA PRALE
026E          A596    LDA MIN          ;ADRESSE MINIMUM-RASTER VERSORGEN
0270          8D7C02  STA PMIN
0273          A597    LDA MIN+1
0275          8D7D02  STA PMIN+1
0278          20B89C  JSR READ         ;LESEN NAECHSTER BLOCK IN MINIMUM-
027B PRALE    00      .BYT 0                                        RASTER
027C PMIN     0000    .WOR 0
027E          9019    BCC IN90         ;KEIN EOF - BLOCKEINGABE BEENDEN
0280          20B29C  JSR OPI          ;BEI EOF NAECHSTES FILE ANFORDDERN
0283          AD12A4  LDA INFLG NAECHSTES FILE GEWUENSCHT
0286          C954    CMP #'T'
0288          F0D9    BEQ INPUT        ;JA - ERSTEN SATZ NEUES FILE LESEN
028A          A9FF    LDA #$FF         ;NEIN - ENDFLAG SETZEN
028C          859E    STA ENDFL
028E          D0D3    BNE INPUT        ;UND INPUT FUER GELOESCHTES RASTER
0290          ---------------------                          WIEDERHOLE
0290          EINSETZEN EINES HIGH-VALUE-BLOCKES
0290
0290 IN80
0290          A9FF    LDA #$FF         ;MINIMUM-RASTER MIT HIGH-VALUE ALS
0292          9196    STA (MIN),Y      ;LOESCHZEICHEN FUELLEN
0294          C8      INY
0295          C491    CPY RALE
0297          D0F7    BNE IN80
```

```
0299 IN90
0299          60      RTS              ;ENDE SATZEINGABE
029A          ----------------------
029A          AUSGABE EINES SATZES
029A
029A OUTPUT
029A          A491    LDY RALE         ;RASTERLAENGE LADEN
029C          88      DEY
029D OU10
029D          A9FF    LDA #$FF
029F          ;PRUEFEN OB MINIMUM-RASTER BEREITS GELOESCHT
029F          D196    CMP (MIN),Y
02A1          D00A    BNE OU30         ;NICHT GELOESCHT - SORTIERUNG
02A3          88      DEY                GEHT WEITER
02A4          10F7    BPL OU10
02A6          209F9C JSR CLO
02A9          ;GELOESCHT - SORTIERUNG BEENDET - AUSGABEDATEI
02A9          A2FF    LDX #$FF
02AB          ;SCHLIESSEN - END-KENNZEICHEN SETZEN
02AB          D029    BNE OU90         ;UND OUTPUT BEENDEN
02AD OU30                                          VERGLEICH
02AD          38      SEC              ;ANFANGSBEDINGUNGEN FUER SBC/
02AE          A20F    LDX #$0F         ;LAENGE SORTIERFELD SETZEN
02B0 OU40
02B0          B480    LDY SORTF,X      ;PRUEFEN OB AUSGABESATZ GROESSER
02B2          3005    BMI OU45         ;AUSGABESATZ          ALS VORHERIG
02B4          B196    LDA (MIN),Y
02B6          F90000 SBC LAST,Y
02B9 OU45
02B9          CA      DEX
02BA          10F4    BPL OU40                              RICHTIG
02BC          B004    BCS OU50         ;SATZ IST GROESSER - SORTFOLGE
02BE          A9FF    LDA #$FF         ;SATZ IST KLEINER
02C0          859F    STA REDOFL                    IN LETZTER SATZ
02C2 OU50                              ;MINIMUM-RASTER UMSPEICHERN
02C2          A000    LDY #0           ;FUER AUSGABE UND NAECHSETN
02C4 OU60                               VERGLEICH
02C4          B196    LDA (MIN),Y
02C6          990000 STA LAST,Y
02C9          C8      INY
02CA          C491    CPY RALE
02CC          D0F6    BNE OU60
02CE OU80                              ;SATZ AUSGEBEN MIT UNTERPROGRAMM
02CE          208C9C JSR WRITE
02D1          4E      .BYT 78
02D2          0000    .WOR LAST
02D4          A200    LDX #0
02D6 OU90
02D6          60      RTS
02D7          ----------------------
02D7          INITIALISIERUNG
02D7
02D7 BEG
02D7          A200    LDX #0           ;ANFANGSTEXT AUSGEBEN
02D9          206203 JSR PRT
```

```
02DC            ;---------------------
02DC            ;SORTIERPUFFER ANFORDERN
02DC            ;
02DC  BEG10     ;
02DC            20A3E7 JSR FROM         ;ADRESSE FUER PUFFERANFANG ANFORD.
02DF            B0FB   BCS BEG10        ;BEI FEHLER WIEDERHOLEN
02E1            AD1CA4 LDA ADDR         ;ADRESSE UMSPEICHERN IN ANFANGSADR
02E4            8592   STA ANF
02E6            AD1DA4 LDA ADDR+1
02E9            8593   STA ANF+1
02EB            ;---------------------
02EB            ;RASTERZAHL ANFORDERN
02EB            ;
02EB  BEG20     ;
02EB            A20F   LDX #MSG02-MSG01
02ED            206203 JSR PRT
02F0            205DEA JSR RD2          ;BYTE ALS RASTERZAHL ANFORDERN
02F3            B0F6   BCS BEG20
02F5            8590   STA RAZA         ;UND IN RASTERZAHL ABLEGEN
02F7            ;---------------------
02F7            ;RASTERLAENGE ANFORDERN
02F7            ;
02F7  BEG30     ;
02F7            A21B   LDX #MSG03-MSG01
02F9            206203 JSR PRT
02FC            205DEA JSR RD2          ;BYTE ALS RASTERLAENGE ANFORDERN
02FF            B0F6   BCS BEG30
0301            8591   STA RALE         ;UND IN RASTERLAENGE ABLEGEN
0303            ;---------------------
0303            ;LOESCHE SORTIERFELD
0303            ;
0303            A20F   LDX #$0F         ;LADEN SORTIERFELDLAENGE
0305  BEG40     ;
0305            A9FF   LDA #$FF         ;UND LOESCHZEICHEN
0307            9580   STA SORTF,X      ;LOESCHZEICHEN ABSPEICHERN IN
0309            CA     DEX                          SORTIERFELDD
030A            10F9   BPL BEG40
030C            ;---------------------
030C            ;SORTIERFELDD AUFBAUEN
030C            ;
030C            A229   LDX #MSG04-MSG01
030E            206203 JSR PRT
0311            A200   LDX #00           ;POINTER AUF SORTIERFELD LOESCHEN
0313  BEG42     ;
0313            205DEA JSR RD2          ;BYTE ALS SORTIERPOSITION ANFORD
0316            900D   BCC BEG45        ;RICHTIGES BYTE - WEITER MIT ABSP
0318            C90D   CMP #$0D         ;BEI CR SORTIERFELDEINGABE BEENDEN
031A            F013   BEQ BEG50
031C            20D4E7 JSR QM           ;FALSCHE EINGABE - FRAGEZEICHEN
031F            203EE8 JSR BLANK
0322            18     CLC
0323            90EE   BCC BEG42        ;UND EINGABE WIEDERHOLEN
0325  BEG45     ;
0325            9580   STA SORTF,X      ;SORTIERPOSITION IN SORTFELD ABL.
0327            203EE8 JSR BLANK        ;LEERZEICHE ALS TRENNUNG AUSG
032A            E8     INX              ;SORTIERFELDPOINTER ERHOEHEN
032B            E010   CPX #$10         ;SORTIERFELD VOLL
032D            D0E4   BNE BEG42        ;NEIN - WEITER MIT NAECHSTER POS
                                        ;JA - SORTIERFELDEINGABE BEENDEN
```

```
032F          2013EA JSR CRLOW
0332          ---------------------
0332          OPEN AUS- UND EINGABEDATEIEN
0332
0332          20869C JSR OPO          ;AUSGABEDATEI EROEFFNEN
0335          20B29C JSR OPI          ;EINGABEDATEI EROEFFNEN
0338          A900   LDA #0 LOESCHEN VON
033A          859E   STA ENDFL        ;ENDE-KENNZEICHEN
033C          859F   STA REDOFL       ;WIEDERHOLUNGS-KENNZEICHEN
033E          A250   LDX #80

0340 AN10
0340          9500   STA LAST,X      ;VERGLEICHSPUFFER FUER LETZTEN
0342          CA     DEX                                    SATZ
0343          10FB   BPL AN10
0345          A590   LDA RAZA         ;RASTERZAHL IN AKTUELLE RASTERZAHL
0347          859D   STA ARAZA
0349          A592   LDA ANF          ;ANFANGSRASTER IN AKTUELLES RASTER
034B          8594   STA AKT
034D          A593   LDA ANF+1
034F          8595   STA AKT+1
0351          ---------------------
0351          ANFANGSFUELLUNG DES SORTIERPUFFERS
0351
0351 AN20
0351          A594   LDA AKT          ;AKTUELLES RASTER IN MINIMUM-
0353          8596   STA MIN                          RASTER UEBERNEHM
0355          A595   LDA AKT+1
0357          8597   STA MIN+1
0359          206302 JSR INPUT        ;SATZ LESEN
035C          205502 JSR INCR         ;AKTUELLES RASTER ERHOEHEN
035F          D0F0   BNE AN20         ;PUFFERENDE NOCH NICHT ERREICHT
0361          60     RTS              ;PUFFER VOLL - INITIALISIERUNG
0362          ---------------------                  BEENDET
0362          PRINT MESSAGE
0362
0362 PRT
0362          BD6F03 LDA MSG01,X      ;MESSAGE-ZEICHEN LDAEN
0365          A8     TAY              ;UND IN Y RETTEN
0366          20FCEE JSR OUTDP        ;ZEICHE AUSGEBEN
0369          E8     INX              ;ZEICHENPOINTER ERHOEHEN
036A          C000   CPY #$00         ;VERGLEICH AUF ENDE-BIT IM ZEICHEN
036C          10F4   BPL PRT          ;NACHRICHT NOCH NICHT ZU ENDE
036E          60     RTS              ;ENDE ERREICHT - RUECKSPRUNG
036F          ---------------------
036F          MESSAGE-POOL
036F
036F MSG01    4149   .BYT 'AIM 1 SORT 1.1',$8D
037D          8D
037E MSG02    0D     .BYT $0D,'RASTERZAHL',$8D
037F          5241
0389          8D
038A MSG03    0D     .BYT $0D,'RASTERLAENGE',$8D
038B          5241
0397          8D
0398 MSG04    0D     .BYT $0D,'SORTIERFELD',$8D
0399          534F
03A4          8D
```

```
03A5 M8B08    534F    .BYT 'SORT READY',$8D
03AF          8D
03B0 M8B09    5245    .BYT 'REDO SORT',$8D
03B9          8D
03BA                  .END
```

#

Ing. grad. Horst Steder, 6000 Frankfurt

LINED (AIM)

Dieses für den AIM 65/PC 100 geschriebene Programm setzt einen TV-Monitor oder ein Terminal voraus. Es stellt dem Text-Editor des AIM zahlreiche zusätzliche Funktionen zur Verfügung, die seine Leistungsfähigkeit für die Bearbeitung von Texten nachhaltig verbessern.

Wer schon einmal in einem 1500 Zeilen langen Source-Text nach einem bestimmten 'BNE *+5' gesucht oder mühsam mit Zurhilfenahme des Tape Textblöcke verschoben hat, der wird nach kurzer Zeit dieses Dienstleistungsprogramm nicht mehr missen mögen.

Die ersten Versuche für dieses Programm wurden vor mehr als einem Jahr durchgeführt. Nach mehr als 30 Zwischenversionen ergab sich die hier vorgelegte Form. Es sollte folgenden Anforderungen genügen:

1) Automatische Zeilennummern-Vergabe mit ebenfalls automatischem Renumbering bei Änderungen. Zeilennummern sind z.B. für die Blockmove-Routine ein Muß.
2) Die erzeugten Zeilennummern müssen virtuell sein, um ohne Probleme mit dem Assembler arbeiten zu können und um den ohnehin schmalbrüstigen Display-Buffer nicht noch zusätzlich zu belasten.
3) Es sollte kein zusätzlicher RAM-Bereich außerhalb der von Monitor und Standard-Editor bereits beanspruchten Bereiche für Zwischenspeicherung notwendig werden.

Außerdem lag noch die Forderung vor, dieses Programm im Rahmen einer Monitor-Erweiterung in den vorhandenen EPROM-Bereich hineinzubringen. Das führte zu einer sehr ausgedehnten Verwendung von AIM-Routinen und zu teilweise extremen 'byte crunching'. Zudem wurde bei den Unterroutinen stark modularisiert, um diese möglichst universell für andere Routinen verwenden zu können. Ich hoffe aber, daß mit Hilfe der 'Gebrauchsanleitung' und der ausführlichen Kommentierung eigene Erweiterungen leicht durchzuführen sind. - Bei dieser Gelegenheit möchte ich mich bei den Mitstreitern an diesem Projekt bedanken, die mich durch ihre ständige konstruktive Unzufriedenheit über das Erreichte zwangen, immer wieder als notwendig erachtete Verbesserungen durchzuführen.

```
*****************************************************
*** ERKLAERUNG LINE - EDITOR VON HORST STEDER    05/80   ***
*****************************************************
              AUFRUF LINE - EDITOR MIT <F3>
*****************************************************
FOLGENDE BEFEHLE HABEN EXAKT DIE GLEICHE WIRKUNG WIE IM
TEXT-EDITOR DES AIM 65 :

              T,B,F,C,U,D,I,K, SPACE

*****************************************************
ABWEICHEND SIND :
================
<Q> - QUIT EDITOR MIT CURRENT LINE IN DEN STANDARD - EDITOR
      STATT IN DEN MONITOR.

*****************************************************
```

```
<R> = LIEST GENAU WIE DER STANDARD-EDITOR EIN,VERGIBT ABER
      AUTOMATISCH DIE RICHTIGE ZEILENNUMMER.
      ES IST NUR DIE EINGABE VOM KB VORGESEHEN, ALSO KEINE
      ABFRAGE NACH INPUT DEVICE (IN =).
      WENN IN DEN TEXT VOM TAPE ODER USER EINGEGEBEN WERDEN
      SOLL:
      AUF GEWUENSCHTE CURRENT LINE GEHEN
      MIT <Q> IN DEN STANDARD-EDITOR SPRINGEN
      WIE UEBLICH <R> AUFRUFEN.
      AUS DEM <R>-COMMAND KANN NUR (M. <ESC> ODER 2 X <CR>)
      IN DEN MONITOR GESPRUNGEN WERDEN. ABER MIT <F3> ER-
      SCHEINT DIE RICHTIG NUMERIERTE FOLGENDE ZEILE.
*************************************************************
<L> = WIRKT GENAU WIE IM STANDARD-EDITOR. OUTPUT-DEVICE IST
      ABER NUR DAS TERMINAL ODER TV-DISPLAY, HIERDURCH WIRD
      MIT SICHERHEIT VERHINDERT, DASS DIE ERZEUGTEN ZEILEN-
      NUMMERN AUS VERSEHEN AUF DEM BAND ABGESPEICHERT WERDEN
      WENN TEXT-TEILE AUF TAPE GESPEICHERT WERDEN SOLLEN:
      WIE OBEN AN DER CURRENT LINE MIT <Q> IN DEN AIM-EDITOR
      DANN <L> WIE GEWOHNT VERWENDEN.

*************************************************************

-------------------------------------------------------------
ZUSAETZLICHE BEFEHLE, DIE IM STANDARD-EDITOR NICHT VORHANDEN
SIND :

IN DEN [ ] DIE VOM ANWENDER EINZUGEBENDEN PARAMETER :
-------------------------------------------------------------

</> = [ZEILEN-NR]
      ZEIGT DIE ZEILE MIT DER GEWUENSCHTEN NUMMER. KEIN
      VORHERIGES SETZEN AUF DIE TOP-LINE MIT <T> NOETIG!
*************************************************************
<M> FROM= [ZEILEN-NR.] TO= [ZEILEN-NR.] / [ANZAHL DER ZEILEN]
      VERSCHIEBT EINEN BLOCK VON BIS ZU 99 ZEILEN BELIEBIG
      INNERHALB DES TEXTES. HIERZU NOCH FOLGENDE HINWEISE :

A. BEI EINER VERSCHIEBUNG NACH UNTEN WIRD DER BLOCK HINTER
DER * TO * - ZEILENNUMMER EINGEFUEGT , BEI VERSCHIEBUNG NACH
OBEN WIRD DER BLOCK VOR DIE SPEZIFIZIERTE ZEILE GESETZT.
HIERDURCH IST EIN VERSCHIEBEN VOR DIE OBERSTE UND HINTER DIE
LETZTE ZEILE GEWAEHRLEISTET.

B. WIRD DIE VERSCHIEBUNG HINTER EINE NICHT MEHR EXISTIERENDE
ZEILENNUMMER GEWAEHLT (Z.B. TO = 2000 BEI EINER TEXTLAENGE
VON 500 ZEILEN), SO WIRD NUR DIE LETZTE ZEILE ANGEZEIGT UND
DANN IN DIE KOMMANDOSCHLEIFE GESPRUNGEN.

C. WIRD AUS VERSEHEN DIE VERSCHIEBUNG AUF DIE GLEICHE ZEILE
GEWAEHLT, ( Z.B. FROM 200 TO 200), SO ERSCHEINT EIN "?"
UND ES ERFOLGT EBENFALLS EIN RUECKSPRUNG IN DIE KOMMANDO-
SCHLEIFE..

D. DIE ANZAHL DER ZEILEN MUSS IMMER ZWEISTELLIG EINGEGEBEN
WERDEN, Z.B. BEI 5 ZEILEN : 05  (KEIN <DEL> MOEGLICH!)

E. BEI <CR> ODER <SPC> WIRD NUR EINE ZEILE VERSCHOBEN.

BEISPIEL : VERSCHIEBUNG VON 5 ZEILEN VON 0 BEGINNEND HINTER
DIE ZEILE 400 :

*<M> FROM = 0 TO = 400 / 05.
#     ......  PROMPT FUER START
    400....   ZEILENNUMMER BLEIBEN GLEICH...



    400....   DA AUTOMATISCHES RENUMERIEREN ERFOLGT
    400....
    400....
    400....
END

DIE VERSCHOBENEN ZEILEN WERDEN AUF DEM DISPLAY ANGEZEIGT.

ACHTUNG: BEI LANGEN SOURCE-TEXTEN KANN DIE VERSCHIEBUNG PRO
```

```
ZEILE EINIGE SEKUNDEN DAUERN. DAHER WIRD DER BEGINN MIT DEM
PROMPT "*" ANGEZEIGT, DAS ENDE MIT "END".
================================================================
<S> FROM= [ZEILEN-NR] TO= [ZEILEN-NR]
      LOESCHT EINEN BLOCK VON ZEILE X BIS ZEILE Y.
      HIER IST NOCH EINE SICHERHEIT GEGEN UNBEABSICHTIGTES
      LOESCHEN EINGEBAUT, WIE IM FOLGENDEN BEISPIEL GEZEIGT:

<S> FROM= 200 TO= 210
SURE ? N   (ODER JEDES ZEICHEN AUSSER "Y")
END

<S> FROM= 100 TO= 110
SURE? Y
100..... GELOESCHTE ZEILEN WERDEN NOCHMAL ANGEZEIGT...
100....ZEILEN-NR BLEIBT IMMER GLEICH, DA AUTOMATISCHES
100....RENUMERIEREN ERFOLGT
100....
    101....NAECHSTE ZEILE ..

================================================================
<1> = FINDET UND ZEIGT NACHEINANDER ALLE ZEILEN , IN DENEN
      DER DEFINIERTE STRING (AUCH ALS SUBSTRING WIE IM
      F-COMMAND VORKOMMT.
      SUCHT AUTOMAMATISCH VON DER TOP-LINE AB.
================================================================
<2> = FINDET UND LISTET ALLE ZEILEN, IN DENEN DER DEFINIERTE
      >> ABGESCHLOSSENE <<  STRING = LABEL VORKOMMT.
      ZEILEN, IN DENEN DAS LABEL MEHRFACH VORKOMMT,WERDEN
      AUCH MEHRFACH ANGEZEIGT.
================================================================
<3> = AENDERT EIN LABEL UEBERALL IN EIN NEU DEFINIERTES.
      BEDIENUNG WIE BEIM <C> - COMMAND.
      BEISPIEL :  <3> ABC  TO= XYZ
      DAS LABEL ABC WIRD ZU XYZ GEAENDERT, DAS LABEL ABCD
      JEDOCH NICHT, DA ABC HIER NUR SUBSTRING IST. DADURCH
      WIRD SICHERGESTELLT, DASS NUR DAS GEWUENSCHTE LABEL
      UEBERALL GEAENDERT WIRD, NATUERLICH AUCH IN ALLEN
      MOEGLICHEN KOMBINATIONEN WIE: ABC+3 , #ABC+DDD  ODER
      (ABC),Y  USW.

HIER IST EBENFALLS EINE SICHERUNG GEGEN DIE FAST
IRREPARABLE LOESCHEUNG EINES LABELS IM GESAMTEN SOURCE-
TEXT BEIM VERSEHENTLICHEN DRUECKEN DER <CR>-TASTE
NACH DEM PROMPT "TO=" EINGEBAUT;
IN DIESEM FALL WIRD NUR DAS ERSTE GEFUNDENE LABEL GELOESCHT,
DANN ERSCHEINT ABER "SURE?"...
================================================================
<4> WIE <2>, ABER AB CURRENT LINE STATT VON DER TOP LINE.
================================================================
<5> WIE <3>, ABER AB CURRENT LINE STATT VON DER TOP LINE.
================================================================
<6> = DELETE COMMENT IN CURRENT LINE
      LOESCHT ALLE ZEICHEN, DIE NACH EINEM SEMIKOLON (;)
      IN DER ZEILE EXISTIEREN INKL. DES SEMIKOLONS SELBST.
      ES WIRD AUCH GEPRUEFT, OB SICH VOR DEM ";" EIN SPACE
      BEFINDET UND LOESCHT DIESES GGF. DIE GEAENDERTE ZEILE
      WIRD NOCH EINMAL ANGEZEIGT.
================================================================
<7> FROM= [ZEILEN-NR] TO= [ZEILEN-NR]
      LOESCHT ALLE KOMMENTARE IM ANGEGEBENEN ZEILENBEREICH.
      ZUR SICHERHEIT ERSCHEINT HIER EBENFALLS NACH DER PARA-
      METER-EINGABE DER PROMPT "SURE?", DER MIT "Y" BEANT-
      WORTET WERDEN MUSS, BEVOR DIE DELETE-ROUTINE BEGINNT.

HINWEIS : BEI NUR - KOMMENTAREN WIRD DIE GANZE ZEILE DURCH
EIN SPACE ERSETZT. DADURCH VERAENDERT SICH DIE ZEILENZAHL
NICHT.

ACHTUNG: DELETE - COMMENTS <6> UND <7> LOESCHEN ALLE ZEICHEN
HINTER EINEM SEMIKOLON.BEI BEFEHLEN WIE  LDA #';'  ODER BEI
TABELLEN MIT SEMIKOLON IST ALSO VORSICHTSHALBER STATT ";"
DER HEXA - WERT vON $3B EINZUSETZEN !!!
================================================================
<0> = ADD COMMENT IN CURRENT LINE
      FUEGT AN EINE ZEILE DEN GEWUENSCHTEN KOMMENTER MIT
      SEMIKOLON UND EINEM SPACE AN.
```

```
      BEISPIEL:
=</>=144
    0144  LDA PNTR
=<0>TO=DAS IST EIN KOMMENTAR


    0144  LDA PNTR ;DAS IST EIN KOMMENTAR
*****************************************************************

HINWEIS : BEI ALLEN ROUTINEN, DIE DIE BEARBEITUNG VON ZEILEN
BLOECKEN ERLAUBEN, KANN GRUNDSAETZLICH DURCH DRUECKEN VON
SPACE DER VORGANG ANGEHALTEN, BZW. DURCH ESCAPE DER VORGANG
DURCH RUECKSPRUNG IN DEN MONITOR ABGEBROCHEN WERDEN.
GILT ALSO FUER  -- S,M,L,1,2,3,4,5,7 --
*****************************************************************

                  ---  VIEL SPASS !  ---     H.STEDER

END=<
```

```
0000           0000 ; ***********************************************
0000           0001 ; ***********************************************
0000           0002 ; ***          =====   L I N E D   =====        ***
0000           0003 ; ***      LINE-# EDITOR FOR THE AIM-65         ***
0000           0004 ; ***  VERS. 3.6                      03/81    ***
0000           0005 ; ***                  COPYRIGHT BY H.STEDER    ***
0000           0006 ; ***********************************************
0000           0007 ; @@@  KOMMERZIELLE VERWERTUNG VORBEHALTEN   @@@
0000           0008 ; ***********************************************
0000           0009
0000           0010 ;THIS PROGRAM IS AN EXTENSION OF THE AIM-65 EDITOR
0000           0011 ;AND USES A LARGE NUMBER OF THE EDITOR- AND MONITOR
0000           0012 ;ROUTINES. IT FEATURES ADDITIONAL CAPABILITIES BE-
0000           0013 ;SIDES THE DYNAMIC LINE NUMBERING AS: FIND A LINE BY
0000           0014 ;IT'S NUMBER, DELETE A BLOCK OF LINES, MOVE A
0000           0015 ;BLOCK OF LINES ANYWHERE WITHIN THE SOURCE TEXT,
0000           0016 ;FIND ALL DEFINED STRINGS, FIND ALL DEFINED LABELS,
0000           0017 ;AND CHANGE ALL DEFINED LABELS TO A NEW ONE.
0000           0018 ;ALSO: DELETE COMMENTS IN SINGLE AND MULTIPLE LINES,
0000           0019 ;AND ADD COMMENT IN CURRENT LINE.
0000           0020 ;IS PART OF THE FAST-ASSEMBLER PACKAGE IN :MONEX 2.3:
0000           0021
0000           0022 ; *** ZERO-PAGE POINTERS AND VARIABLES ***
0000           0023         *=$DB
00DB           0024 LINUM   *=*+2            ;LINE-# COUNTER
00DD           0025 SAVPOS  *=*+1            ;SAVE POINTER FOR SEARCH
00DE           0026 CRSFLG  *=*+1            ;CROSS REF FLAG
00DF           0027 NOWLN   *=*+2            ;EDITOR POINTER
00E1           0028         *=$E9
00E9           0029 OLDLEN  *=*+1            ;STANDARD EDITOR VARIABLES
00EA           0030 LENGTH  *=*+1
00EB           0031 STRING  *=*+1
00EC           0032 CNTR1   *=*+1            ;# OF LINES (MOVE+LIST), ALSO FLAG
00ED           0033 INTLEN  *=*+1            ;INTERIM STRING LENGTH SAVE
00EE           0034 POSFLG  *=*+1            ;POSITIVE FLAG (FOR OFFSET)
00EF           0035 CMPVAL  *=*+2            ;TO COMPARE WITH CURRENT LINE-#
00F1           0036 CMDVCT  *=*+2            ;LINED COMMAND VECTOR
00F3           0037 OFFSET  *=*+2            ;# OF BYTES TO WORK WITH
00F5           0038 SVCNT   *=*+2            ;SAVE OFFSET HERE
00F7           0039 LEN1    *=*+1            ;SAVE NEW STRING LENGTH
00F8           0040 OLEN1   =SVCNT           ;SAVE OLD STRING LENGTH
00F8           0041 COMFLG  =LEN1            ;FLAG FOR COMMENT DELETE
00F8           0042
00F8           0043 ; *** PAGE-ONE POINTERS AND BUFFER ***
00F8           0044 BUFFER  =$120            ;GEN PURPOSE BUFFER
00F8           0045
00F8           0046 ; *** MONITOR POINTERS AND ROUTINES ***
00F8           0047 ; INCLUDE ALL REFERENCES TO THE COMPLETE MONITOR-
00F8           0048 ; EXPANSION AND ARE NOT LISTED TO SAVE LIST SPACE.
00F8           0146         .OPT LIS
00F8           0147
00F8           0148         *=$112           ; <F3> FUNCTION KEY
0112 4C1050    0149         JMP ERR0         ;START WITH INITIALIZING
0115           0150
```

```
0115            0151          *=$5000           ;+++ START OF LINED
5000            0152
5000 20BCFE     0153 LINED    JSR PATCH8        ;GET COMMAND IN < >
5003 A216       0154          LDX #22           ;FOR 23 COMMANDS
5005 DD2950     0155 ECMD1    CMP CMDTB,X       ;FIND CMD IN TABLE
5008 F00F       0156          BEQ ECMD2         ;CMD FOUND, PROCESS IT
500A CA         0157          DEX               .
500B 10F8       0158          BPL ECMD1
500D 20D4E7     0159 ERR00    JSR QM            ;SHOW "?", GO BACK
5010 20F0E9     0160 ERR0     JSR CRLF
5013 A2FF       0161 ERR1     LDX #$FF          ;RESET STACK POINTER
5015 9A         0162          TXS
5016 D8         0163          CLD
5017 D0E7       0164          BNE LINED         ;WAIT FOR NEXT CMD
5019 8A         0165 ECMD2    TXA               ;GET VECTOR FOR CMD
501A 0A         0166          ASL A
501B AA         0167          TAX
501C BD4050     0168          LDA EDCMD,X       ;STORE IT FOR INDIRECT JMP
501F 85F1       0169          STA CMDVCT
5021 BD4150     0170          LDA EDCMD+1,X
5024 85F2       0171          STA CMDVCT+1
5026 6CF100     0172          JMP (CMDVCT)      ;EXECUTE COMMAND
5029            0173
5029 494B       0174 CMDTB    .BYT 'IKFLSQ/UDM TBCR'
5038 3132       0175          .BYT '12345670'
5040            0176
5040 8C50       0177 EDCMD    .WOR INSERT,DLTLIN,FNDCHR,LSTOUT,DLTBLK
5042 E650       0177
5044 2B51       0177
5046 6E50       0177
5048 EE50       0177
504A 8750       0178          .WOR EXIT65,FNDLIN,UPLIN,DWNLIN,MOVBLK
504C 9A50       0178

504E B250       0178
5050 C850       0178
5052 8A51       0178
5054 AC50       0179          .WOR CURLIN,TOPLIN,BOTTOM,CHNGLN,READIN
5056 8D50       0179
5058 9250       0179
505A 4C51       0179
505C 7651       0179
505E 3151       0180          .WOR FNDSTR,FNDLBL,CHANGE,FNDLBL+3,CHANGE+3
5060 4051       0180
5062 5851       0180
5064 4351       0180
5066 5B51       0180
5068 0151       0181          .WOR DLTCMT,DLTCMS,ADDCMT
506A 0B51       0181
506C 2551       0181
506E            0182
506E            0183 ; LIST DESIRED NUMBER OF LINES
506E 20EB53     0184 LSTOUT JSR GTCNT         ;GET # OF LINES TO LIST
5071 20F753     0185 LST01  JSR FDCHAR+3      ;SHOW NEXT LINE
5074 201C53     0186        JSR UPN01         ;ONE LINE DOWN
5077 2090E7     0187        JSR DONE          ;ALL DONE?
507A F005       0188        BEQ LSTEND        ;YES, STOP
507C 20E9F8     0189        JSR ATBOT         ;AT TEXT END ?
507F 90F0       0190        BCC LST01         ;NO, STAY IN LOOP
5081 200B53     0191 LSTEND JSR DOWLN         ;BACK UP ONE LINE
5084 4C1350     0192        JMP ERR1          ;BACK TO NEW CMD W/O CR/LF
5087            0193
5087            0194 ; EXIT WITH CURRENT LINE TO AIM-EDITOR
5087 20F0E9     0195 EXIT65 JSR CRLF
508A 4C89F7     0196        JMP IN03A         ;TO AIM-65 EDITOR
508D            0197
508D            0198 ; FIND AND DISPLAY TOP LINE
508D 206E54     0199 TOPLIN JSR TOPLN         ;GET LINE #0000
5090 101A       0200        BPL CURLIN        ;DISPLAY LINE (JMP ALWAYS)
5092            0201
5092            0202 ; FIND AND DISPLAY BOTTOM LINE
5092 A999       0203 BOTTOM LDA #$99          ;PRESET MAX LINE-# TO 9999
5094 85EF       0204        STA CMPVAL
5096 85F0       0205        STA CMPVAL+1
5098 D006       0206        BNE FIND1         ;SEARCH FOR LAST LINE
509A            0207
509A            0208 ; FIND A LINE BY IT'S LINE-NUMBER
509A 20AEEA     0209 FNDLIN JSR ADDIN         ;GET DESIRED LINE-#
```

```
509D 20D353  0210         JSR TRANS        ;ADDR --> CMPVAL
50A0 203BE8  0211 FIND1   JSR BLANK2
50A3 207D53  0212         JSR FNDMVL       ;FIND DESIRED LINE
50A6 20F0E9  0213 OUTPT1  JSR CRLF
50A9 20E553  0214         JSR NEWLIN
50AC 20FD53  0215 CURLIN  JSR SHOWLN+3     ;SHOW LINE WITH LINE-#
50AF 4C1350  0216         JMP ERR1         ;BACK TO NEXT CMD W/O CR/LF
50B2         0217
50B2         0218 ; GO ONE LINE UP
50B2 20D8F8  0219 UPLIN   JSR ATTOP        ;ALREADY AT TOP?
50B5 B0F5    0220         BCS CURLIN       ;YES,SHOW TOP LINE
50B7 200B53  0221         JSR DOWLN        ;NO, GO ONE LINE UP
50BA B0F0    0222         BCS CURLIN       ;SHOW LINE
50BC         0223
50BC         0224 ; INSERT A NEW LINE
50BC 20E553  0225 INSERT  JSR NEWLIN       ;BLANKS FOR NICE FORMAT
50BF 203BE8  0226         JSR BLANK2
50C2 206DF7  0227         JSR INL          ;INSERT NEW LINE
50C5 20E553  0228         JSR NEWLIN
50C8         0229
50C8         0230 ; GO ONE LINE DOWN
50C8 201C53  0231 DWNLIN  JSR UPNO1        ;ONE LINE DOWN
50CB 90DF    0232         BCC CURLIN       ;SHOW LINE
50CD A000    0233 UPNO    LDY #0
50CF 20E9F8  0234         JSR ATBOT        ;ALREADY AT BOTTOM?
50D2 B003    0235         BCS *+5          ;YES, IS END
50D4 4C13F7  0236         JMP UP1          ;NO, GO ONE LINE DOWN
50D7 200B53  0237         JSR DOWLN        ;ONE LINE UP...
50DA 20F0E9  0238         JSR CRLF
50DD 20FA53  0239         JSR SHOWLN       ;TO SHOW LAST VALID LINE
50E0 2004F7  0240 TXTEND  JSR ENDMSG       ;SHOW "END"
50E3 4C1050  0241         JMP ERR0         ;BACK TO NEXT CMD
50E6         0242
50E6         0243 ; DELETE A SINGLE LINE
50E6 20FD53  0244 DLTLIN  JSR SHOWLN+3     ;SHOW LINE
50E9 209C53  0245         JSR KILLIN       ;DELETE LINE
50EC F0BB    0246         BEQ OUTPT1+3     ;JMP ALWAYS
50EE         0247
50EE         0248 ; DELETE A BLOCK OF LINES
50EE 20E851  0249 DLTBLK  JSR GTINFO       ;GET PARAMETERS
50F1 20FD53  0250 DLTLN1  JSR SHOWLN+3     ;SHOW LINE
50F4 2007E9  0251         JSR RCHEK        ;IF INTERRUPT WANTED
50F7 209C53  0252         JSR KILLIN       ;DELETE LINE
50FA 20A653  0253         JSR DECOFF       ;DECREMENT # OF LINES
50FD B0F2    0254         BCS DLTLN1       ;STILL LINES TO DELETE
50FF D0A5    0255 DLTLN2  BNE OUTPT1       ;JMP ALWAYS
5101         0256
5101         0257 ; DELETE COMMENT IN CURRENT LINE
5101 A900    0258 DLTCMT LDA #0
5103 85F7    0259         STA COMFLG       ;FLAG FOR SINGLE LINE
5105 20F751  0260         JSR DLTC0        ;FIND AND DELETE COMMENT
5108 4C1350  0261         JMP ERR1         ;SHOW CHANGED LINE
510B         0262
510B         0263 ; DELETE COMMENTS IN A BLOCK OF LINES
510B 20E851  0264 DLTCMS  JSR GTINFO       ;GET PARAMETERS
510E A901    0265         LDA #1
5110 85F7    0266         STA COMFLG       ;FLAG FOR MULTIPLE LINES
5112 20FD53  0267 DLTS1   JSR SHOWLN+3     ;SHOW LINE
5115 2007E9  0268         JSR RCHEK        ;IF INTERRUPT WANTED
5118 20F751  0269         JSR DLTC0        ;SEARCH FOR COMMENTS
511B 201C53  0270         JSR UPNO1        ;ONE LINE DOWN
511E 20A653  0271         JSR DECOFF       ;DECREMENT # OF LINES
5121 B0EF    0272         BCS DLTS1        ;STILL LINES TO CHECK
5123 D0DA    0273         BNE DLTLN2
5125         0274
5125         0275 ; ADD A COMMENT IN LINE
5125 204952  0276 ADDCMT  JSR ADCOM        ;ADD COMMENT WITH SPACE AND ';'
5128 4CA950  0277         JMP OUTPT1+3     ;SHOW CHANGED LINE
512B         0278
512B         0279 ; FIND A CHARACTER STRING IN TEXT
512B 20F453  0280 FNDCHR  JSR FDCHAR       ;FIND AND DISPLAY LINE
512E 4C1350  0281         JMP ERR1         ;BACK FOR NEW COMMAND
5131         0282
5131         0283 ; FIND ALL DEFINED STRINGS
5131 206E54  0284 FNDSTR  JSR TOPLN        ;GO TO TOP LINE
5134 20F453  0285         JSR FDCHAR       ;FIND 1ST STRING AND SHOW IT
5137 208652  0286 FNDS1   JSR FC2
```

```
513A 20F753   0287          JSR FDCHAR+3     ;ALSO SHOW ALL OTHERS
513D 4C3751   0288          JMP FNDS1
5140          0289
5140          0290 ; FIND ALL DEFINED LABELS
5140 206E54   0291 FNDLBL JSR TOPLN        ;GO TO TOP
5143 200354   0292          JSR FNDALL       ;TEST IF LABEL
5146 201954   0293 FNDL1  JSR FNDA         ;SHOW ALL LINES WITH LABEL...
5149 4C4651   0294          JMP FNDL1        ;UNTIL END OF SOURCE
514C          0295
514C          0296 ; CHANGE A CHARACTER STRING IN LINE
514C 20B2F8   0297 CHNGLN JSR CFLG         ;SET C CMD FLAG
514F 20F453   0298          JSR FDCHAR       ;FIND CORRECT LINE
5152 20CC52   0299          JSR CHN1
5155 4CA650   0300          JMP OUTPT1       ;DISPLAY CHANGED LINE
5158          0301
5158          0302 ; CHANGE ALL DEFINED LABELS
5158 206E54   0303 CHANGE JSR TOPLN        ;GO TO TOP
515B 20B2F8   0304          JSR CFLG         ;SET CHANGE FLAG
515E 200354   0305          JSR FNDALL       ;FIND FIRST LABEL
5161 20CC52   0306          JSR CHN1         ;CHANGE IT
5164 20F753   0307          JSR FDCHAR+3     ;DISPLAY CHANGED LABEL
5167 207454   0308          JSR SFTYCK       ;SAFETY CHECK ("SURE?")
516A 201954   0309 CHAN1  JSR FNDA         ;FIND NEXT LABEL
516D 203D54   0310          JSR CHNG5        ;CHANGE IT
5170 20F753   0311          JSR FDCHAR+3     ;DISPLAY IT
5173 4C6A51   0312          JMP CHAN1        ;BACK UNTIL END
5176          0313
5176          0314 ; READ NEW LINES INTO TEXT
5176 20E553   0315 READIN JSR NEWLIN       ;BLANKS FOR NICE FORMAT
5179 203BE8   0316          JSR BLANK2
517C 206DF7   0317          JSR INL          ;READ LINES INTO EDITOR
517F 20FA53   0318          JSR SHOWLN
5182 20E553   0319          JSR NEWLIN
5185 201C53   0320          JSR UPNO1
5188 90EC     0321          BCC READIN       ;(EXIT ONLY TO MONITOR)
518A          0322
518A          0323 ; MOVE A BLOCK OF LINES WITHIN SOURCE TEXT
518A 202F53   0324 MOVBLK JSR GETBLK       ;GET ORIGIN & DEST. OF BLOCK
518D 20EB53   0325          JSR GTCNT        ;GET # OF LINES TO BE MOVED
5190 20B653   0326          JSR CTRNS        ;SAVE # OF LINES
5193 20CAE7   0327          JSR PROMPT       ;PROMPT WITH "*" AND START
5196 20F0E9   0328          JSR CRLF
5199 207453   0329 MVBLK1 JSR SVEOFF       ;SAVE OFFSET
519C 20A653   0330          JSR DECOFF       ;DECREMENT OFFSET VALUE
519F 207D53   0331          JSR FNDMVL       ;FIND ORIGIN LINE
51A2 208A53   0332          JSR TRNSLN       ;TRANSFER LINE TO DIBUFF
51A5 209C53   0333          JSR KILLIN       ;DELETE LINE IN SOURCE TEXT
51A8 A5EE     0334 MVBLK2 LDA POSFLG       ;IS OFFSET POS OR NEG ?
51AA F005     0335          BEQ MVBLK3       ;IS POSITIVE
51AC 200B53   0336          JSR DOWLN        ;IS NEGATIVE, ONE LINE UP
51AF B003     0337          BCS *+5          ;JMP ALWAYS
51B1 201C53   0338 MVBLK3 JSR UPNO1        ;GO ONE LINE DOWN
51B4 20A653   0339          JSR DECOFF       ;DECREMENT OFFSET VALUE
51B7 B0EF     0340          BCS MVBLK2       ;NOT YET AT END
51B9 A000     0341          LDY #0           ;DEST. FOUND, INSERT LINE
51BB 84E9     0342          STY OLDLEN
51BD A4ED     0343          LDY INTLEN       ;GET LENGTH OF NEW STRING
51BF 20A153   0344          JSR KILLIN+5     ;AND INSERT LINE IN TEXT
51C2 20F753   0345          JSR FDCHAR+3     ;SHOW LINE WITH LINE-#

51C5 A5EC     0346          LDA CNTR1        ;DECREMENT LINE COUNT
51C7 F8       0347          SED
51C8 38       0348          SEC
51C9 E901     0349          SBC #1
51CB 85EC     0350          STA CNTR1
51CD D8       0351          CLD
51CE D003     0352          BNE MVBLK4       ;STILL LINES TO MOVE
51D0 4CE050   0353          JMP TXTEND       ;DONE, SHOW "END", BACK FOR CMD
51D3 A5EE     0354 MVBLK4 LDA POSFLG       ;OFFSET POS OR NEG?
51D5 F0C2     0355          BEQ MVBLK1       ;POSITIVE, BACK TO LOOP
51D7 F8       0356          SED              ;NEGATIVE, INCREMENT ORIGIN
51D8 18       0357          CLC
51D9 A5EF     0358          LDA CMPVAL
51DB 6901     0359          ADC #1
51DD 85EF     0360          STA CMPVAL
51DF A5F0     0361          LDA CMPVAL+1
51E1 6900     0362          ADC #0
51E3 85F0     0363          STA CMPVAL+1
```

```
51E5 D8      0364          CLD
51E6 9081    0365          BCC MVBLK1       ; ...THEN BACK TO LOOP
51E8         0366
51E8         0367 ; *******   SUBROUTINES   ********
51E8         0368
51E8 202F53  0369 GTINFO JSR GETBLK       ;GET START & END OF BLOCK
51EB 207453  0370          JSR SVEOFF      ;SAVE # OF LINES IN BLOCK
51EE 207D53  0371          JSR FNDMVL      ;FIND START LINE
51F1 20F0E9  0372          JSR CRLF
51F4 4C7854  0373          JMP SFTYCK+4    ;SAFETY CHECK ("SURE?")
51F7         0374
51F7 A000    0375 DLTC0   LDY #0
51F9 84EC    0376          STY CNTR1       ;CLEAR COMMENT-ONLY FLAG
51FB B1DF    0377 DLTC1   LDA (NOWLN),Y   ;SEARCH FOR A SEMICOLON
51FD C90D    0378          CMP #$D         ;IF 'CR' COMES FIRST...
51FF F043    0379          BEQ ECMT        ;THEN EXIT
5201 C93B    0380          CMP #$3B
5203 F003    0381          BEQ DLC1        ;FOUND A SEMICOLON
5205 C8      0382          INY
5206 D0F3    0383          BNE DLTC1
5208 C003    0384 DLC1    CPY #3          ;IF POSITION OF ';' IS < 3...
520A B006    0385          BCS DLC10
520C A901    0386          LDA #1          ;THEN SET CMT-ONLY FLAG..
520E 85EC    0387          STA CNTR1       ;TO REPLACE COMMENT LINE...
5210 D008    0388          BNE DLC11       ;WITH A SINGLE SPACE
5212 88      0389 DLC10   DEY             ;DECREMENT POINTER
5213 B1DF    0390          LDA (NOWLN),Y   ;CHECK WHETHER A SPACE..
5215 C920    0391          CMP #$20        ;PRECEDES THE SEMICOLON
5217 F001    0392          BEQ *+3         ;YES
5219 C8      0393          INY             ;NO, INCREMENT POINTER AGAIN
521A 98      0394 DLC11   TYA             ;ADJUST POINTER (NOWLN)
521B 202AF9  0395          JSR ADDA
521E A000    0396          LDY #0          ;START WITH COMMENT..
5220 B1DF    0397 DLTC2   LDA (NOWLN),Y   ;AS IF IT WERE A NEW LINE
5222 C90D    0398          CMP #$D
5224 F003    0399          BEQ DLC2
5226 C8      0400          INY
5227 D0F7    0401          BNE DLTC2
5229 84E9    0402 DLC2    STY OLDLEN      ;SAVE COMMENT LENGTH
522B 2082F8  0403          JSR CFLG
522E A4EC    0404          LDY CNTR1       ;COMMENT-ONLY LINE?
5230 F005    0405          BEQ DLC21       ;NO, GO ON
5232 A920    0406          LDA #' '        ;YES, REPLACE IT...
5234 9937A4  0407          STA DIBUFF-1,Y  ;WITH A SINGLE SPACE
5237 20A153  0408 DLC21   JSR KILLIN+5    ;DELETE COMMENT ONLY
523A A5EC    0409          LDA CNTR1
523C D003    0410          BNE *+5         ;IF FLAG SET, DON'T...
523E 20E3F6  0411          JSR DOW1        ;ADJUST LINE POINTER
5241 4CFD53  0412          JMP SHOWLN+3    ;SHOW CHANGED LINE
5244         0413
5244 A5F7    0414 ECMT    LDA COMFLG      ;MULTI OR SINGLE LINE?
5246 F04F    0415          BEQ FC4-3       ;SINGLE, -->ERROR
5248 60      0416          RTS             ;MULTI, GET NEXT LINE
5249         0417
5249 A001    0418 ADCOM   LDY #1          ;SET OLDLEN TO 1
524B 84E9    0419          STY OLDLEN
524D B1DF    0420 ADCM    LDA (NOWLN),Y   ;FIND LENGTH OF LINE
524F C8      0421          INY
5250 C90D    0422          CMP #$D         ;END OF LINE ?
5252 D0F9    0423          BNE ADCM        ;NOT YET
5254 88      0424          DEY
5255 88      0425          DEY
5256 B1DF    0426          LDA (NOWLN),Y   ;GET LAST CHAR IN LINE
5258 8D38A4  0427          STA DIBUFF      ;AND STORE IT AS FIRST...
525B 98      0428          TYA             ;OF NEW STRING IN BUFFER
525C 48      0429          PHA             ;SAVE ADJUSTED LENGTH
525D 202AF9  0430          JSR ADDA        ;ADD LENGTH TO NOWLN
5260 2082F8  0431          JSR CFLG        ;SET CHANGE FLAG
5263 A920    0432          LDA #' '        ;INSERT A SPACE...
5265 8D39A4  0433          STA DIBUFF+1    ;IN NEXT BUFFER LOC
5268 A93B    0434          LDA #$3B        ;THEN A SEMICOLON
526A 8D3AA4  0435          STA DIBUFF+2
```

```
526D A005     0436          LDY #5          ;DISPLAY 'TO='...
526F 20AFE7   0437          JSR KEP         ;TO GET COMMENT STRING
5272 A003     0438          LDY #3          ;ADJUST BUFFER POINTER
5274 4CF052   0439          JMP CHN20       ;JUMP TO INSERTION ROUTINE
5277          0440
5277 A000     0441 FCHAR    LDY #0
5279 84DE     0442          STY CRSFLG      ;CLEAR CROSS-REF FLAG
527B 205FE9   0443 FC1      JSR RDRUB       ;GET CHARACTER STRING
527E C90D     0444          CMP #$D         ;REUSE OLD ARGUMENT?
5280 D009     0445          BNE FC3
5282 C000     0446          CPY #0          ;IS 'CR' THE FIRST CHAR ?
5284 D005     0447          BNE FC3
5286 201C53   0448 FC2      JSR UPND1       ;YES, NEXT LINE DOWN
5289 9015     0449          BCC FC5         ;ALWAYS
528B C90D     0450 FC3      CMP #$D         ;DONE
528D F00B     0451          BEQ FC4
528F 99EB00   0452          STA STRING,Y    ;STORE IN STRING BUFFER
5292 C8       0453          INY
5293 C014     0454          CPY #20         ;MAX LENGTH OF BUFFER
5295 D0E4     0455          BNE FC1
5297 4C0D50   0456          JMP ERR00
529A 8C29A4   0457 FC4      STY STIY+2      ;COUNT OF CHARACTERS
529D 20F0E9   0458          JSR CRLF
52A0 A000     0459 FC5      LDY #0
52A2 84DD     0460          STY SAVPOS
52A4 A4DD     0461 FC6      LDY SAVPOS
52A6 A200     0462          LDX #0
52A8 B1DF     0463 FC7      LDA (NOWLN),Y
52AA D003     0464          BNE FC8         ;NOT YET AT END
52AC 4CE050   0465          JMP TXTEND      ;STRING NOT FOUND, SHOW "END"
52AF 24DE     0466 FC8      BIT CRSFLG      ;CROSS-REF FLAG SET?
52B1 1004     0467          BPL NOCRS       ;NO
52B3 C93B     0468          CMP #$3B        ;YES, START OF COMMENT?
52B5 F0CF     0469          BEQ FC2         ;YES, SKIP AND GET NEXT LINE
52B7 C90D     0470 NOCRS    CMP #$D         ;END OF LINE?
52B9 F0CB     0471          BEQ FC2
52BB D5EB     0472          CMP STRING,X    ;COMP WITH CHAR IN STRING
52BD F005     0473          BEQ FC9
52BF E6DD     0474          INC SAVPOS
52C1 4CA452   0475          JMP FC6
52C4 C8       0476 FC9      INY
52C5 E8       0477          INX
52C6 EC29A4   0478          CPX STIY+2      ;DONE?
52C9 D0DD     0479          BNE FC7
52CB 60       0480          RTS
52CC          0481
52CC 203CE9   0482 CHN1     JSR READ        ;IS <CR> IF OK
52CF C90D     0483          CMP #$D
52D1 F009     0484          BEQ CHN2
52D3 208652   0485          JSR FC2         ;TRY NEXT ONE
52D6 20F753   0486          JSR FDCHAR+3
52D9 4CCC52   0487          JMP CHN1
52DC AD29A4   0488 CHN2     LDA STIY+2      ;GET CHAR COUNT
52DF 85F5     0489          STA OLEN1
52E1 85E9     0490          STA OLDLEN      ;GET READY FOR REPLACEMENT
52E3 A5DD     0491          LDA SAVPOS      ;PNTR TO BEG OF STRING
52E5 48       0492          PHA
52E6 202AF9   0493          JSR ADDA        ;ADD TO NOWLN
52E9 A005     0494          LDY #5          ;PRINT "TO"
52EB 20AFE7   0495          JSR KEP
52EE A000     0496          LDY #0
52F0 207AF7   0497 CHN20    JSR IN02        ;GET NEW STRING & REPLACE
52F3 A5EA     0498          LDA LENGTH
52F5 85F7     0499          STA LEN1
52F7 205454   0500          JSR SAVDIB      ;SAVE STRING
52FA 68       0501 CHN21    PLA
52FB AA       0502          TAX
52FC F006     0503          BEQ CHN4
52FE 201DF9   0504 CHN3     JSR SUB         ;RESTORE NOWLN
5301 CA       0505          DEX
5302 D0FA     0506          BNE CHN3
5304 60       0507 CHN4     RTS
5305          0508
5305 A900     0509 CLRLIN   LDA #0          ;RESET LINE-# TO 0
5307 85D8     0510          STA LINUM
5309 F020     0511          BEQ LNA1
5308          0512
```

```
530B 20E3F6  0513 DOWLN   JSR DOW1         ;BACK UP ONE LINE
530E F8      0514         SED              ;DECREMENT LINE COUNTER
530F 38      0515         SEC
5310 A5DB    0516         LDA LINUM
5312 E901    0517         SBC #1
5314 85DB    0518         STA LINUM
5316 A5DC    0519         LDA LINUM+1
5318 E900    0520         SBC #0
531A B00F    0521         BCS LNA1
531C         0522
531C 20CD50  0523 UPNO1   JSR UPNO         ;MOVE ONE LINE DOWN
531F F8      0524 LINADD  SED              ;INCREMENT LINE COUNTER..
5320 18      0525         CLC              ;IN DECIMAL MODE
5321 A5DB    0526         LDA LINUM
5323 6901    0527         ADC #1
5325 85DB    0528         STA LINUM
5327 A5DC    0529         LDA LINUM+1
5329 6900    0530         ADC #0
532B 85DC    0531 LNA1    STA LINUM+1
532D D8      0532 LNA2    CLD
532E 60      0533         RTS
532F         0534
532F 20A3E7  0535 GETBLK  JSR FROM         ;GET ORIGIN LINE-#
5332 48      0536         PHA              ;SAVE IT
5333 AD1DA4  0537         LDA ADDR+1
5336 48      0538         PHA
5337 2038E8  0539         JSR BLANK2
533A 20A7E7  0540         JSR TO           ;GET DESTINATION LINE-#
533D 20D353  0541         JSR TRANS        ;IS IT BEYOND LAST LINE?
5340 207D53  0542         JSR FNDMVL       ;IF SO, SHOW LAST LINE...
5343         0543                          ;AND GO BACK FOR NEW CMD
5343 20D0F8  0544         JSR RESNOW       ;ALL OK, TRANSFER TO NOWLN
5346 68      0545         PLA              ;GET ORIGIN
5347 AA      0546         TAX
5348 68      0547         PLA
5349 20D653  0548         JSR TRNS1        ;ORIGIN TO CMPVAL
534C F8      0549 DISPLC  SED
534D A200    0550         LDX #0           ;CALCULATE OFFSET AND CHECK...
534F 86EE    0551         STX POSFLG       ;WHETHER POSITIVE OR NEGATIVE
5351 38      0552         SEC              ;IF POSITIVE --> POSFLG=0
5352 A5DF    0553         LDA NOWLN        ;IF NEGATIVE --> POSFLG=1
5354 E5EF    0554         SBC CMPVAL
5356 85F3    0555         STA OFFSET
5358 A5E0    0556         LDA NOWLN+1
535A E5F0    0557         SBC CMPVAL+1
535C 85F4    0558         STA OFFSET+1
535E B00D    0559         BCS POS          ;IS POSITIVE, CHECK IF ZERO
5360 E6EE    0560         INC POSFLG       ;IS NEGATIVE, SET FLAG...
5362 8A      0561         TXA              ;AND RECALCULATE...
5363 38      0562         SEC              ;ABSOLUTE OFFSET VALUE
5364 E5F3    0563         SBC OFFSET
5366 85F3    0564         STA OFFSET
5368 8A      0565         TXA
5369 E5F4    0566         SBC OFFSET+1
536B 85F4    0567         STA OFFSET+1
536D 05F3    0568 POS     ORA OFFSET       ;IS OFFSET VALUE ZERO?
536F D0BC    0569         BNE LNA2         ;NO
5371 4C0D50  0570         JMP ERR00        ;YES,-->"?", BACK TO NEW CMD
5374         0571
5374 A5F3    0572 SVEOFF  LDA OFFSET       ;SAVE OFFSET VALUE
5376 85F5    0573         STA SVCNT
5378 A5F4    0574         LDA OFFSET+1
537A 85F6    0575         STA SVCNT+1
537C 60      0576 RTX     RTS
537D         0577
537D 206E54  0578 FNDMVL  JSR TOPLN        ;GET LINE #0000
5380 20C353  0579 FNDM1   JSR CMPLIN       ;DESIRED LINE FOUND ?
5383 90F7    0580         BCC RTX          ;FOUND LINE, RETURN
5385 201C53  0581         JSR UPNO1        ;NO, ONE LINE DOWN
5388 90F6    0582         BCC FNDM1        ;JMP ALWAYS
538A         0583
538A A000    0584 TRNSLN  LDY #0           ;TRANSFER LINE TO DIBUFF
538C B1DF    0585 TRNSL1  LDA (NOWLN),Y
538E 9938A4  0586         STA DIBUFF,Y
5391 C8      0587         INY
5392 C90D    0588         CMP #$0D
5394 D0F6    0589         BNE TRNSL1
```

```
5396 88       0590          DEY
5397 84ED     0591          STY INTLEN       ;SAVE STRING LENGTH
5399 84E9     0592          STY OLDLEN
5398 60       0593          RTS
539C          0594
539C 20B6F8   0595 KILLIN   JSR KIFLG        ;SET DELETE FLAG
539F A000     0596          LDY #0
53A1 84EA     0597          STY LENGTH
53A3 4C3FF9   0598          JMP REPLAC       ;DELETE LINE
53A6          0599
53A6 F8       0600 DECOFF   SED              ;DECREMENT OFFSET VALUE
53A7 38       0601          SEC
53A8 A5F5     0602          LDA SVCNT
53AA E901     0603          SBC #1
53AC 85F5     0604          STA SVCNT
53AE A5F6     0605          LDA SVCNT+1
53B0 E900     0606          SBC #0
53B2 85F6     0607          STA SVCNT+1
53B4 D8       0608          CLD
53B5 60       0609          RTS
53B6          0610
53B6 AD19A4   0611 CTRNS    LDA COUNT        ;TRANSFER # OF LINES
53B9 85EC     0612 CNTS1    STA CNTR1
53BB C92C     0613          CMP #$2C         ;WAS IT A <SPACE>?
53BD D013     0614          BNE RET          ;NO, RETURN
53BF A901     0615          LDA #1           ;YES, REPLACE WITH 1
53C1 D0F6     0616          BNE CNTS1
53C3          0617
53C3 A5DB     0618 CMPLIN   LDA LINUM        ;COMPARE CURRENT LINE-#..
53C5 C5EF     0619          CMP CMPVAL       ;WITH DESIRED LINE-#
53C7 F002     0620          BEQ CMP1
53C9 38       0621 CMP0     SEC              ;SET CARRY IF NO MATCH
53CA 60       0622          RTS
53CB A5DC     0623 CMP1     LDA LINUM+1
53CD C5F0     0624          CMP CMPVAL+1
53CF D0F8     0625          BNE CMP0
53D1 18       0626          CLC              ;CLEAR CARRY IF MATCH
53D2 60       0627 RET      RTS
53D3          0628
53D3 AE1DA4   0629 TRANS    LDX ADDR+1       ;TRANSFER DES. LINE-#
53D6 85EF     0630 TRNS1    STA CMPVAL
53D8 86F0     0631          STX CMPVAL+1
53DA 60       0632          RTS
53DB          0633
53DB A5DC     0634 OUTLIN   LDA LINUM+1      ;DISPLAY LINE-#
53DD A6DB     0635          LDX LINUM
53DF 2042EA   0636          JSR WRAX
53E2 4C3BE8   0637          JMP BLANK2
53E5          0638
53E5 203BE8   0639 NEWLIN   JSR BLANK2       ;START NEW LINE
53E8 4C3BE8   0640          JMP BLANK2
53EB          0641
53EB 2037E8   0642 GTCNT    JSR PSLS         ;SHOW "/" AND GET..
53EE 2085E7   0643          JSR LCNT         ;# OF LINES TO BE..
53F1 4CF0E9   0644          JMP CRLF         ;WORKED ON
53F4          0645
53F4 207752   0646 FDCHAR   JSR FCHAR        ;FIND STRING AND...
53F7 2007E9   0647          JSR RCHEK        ;DISPLAY IT WITH INTERRUPT..
53FA 20E553   0648 SHOWLN   JSR NEWLIN       ;POSSIBILITY VIA RCHEK
53FD 20DB53   0649          JSR OUTLIN       ;SHOW LINE-# & LINE
5400 4C27F7   0650          JMP PLNE
5403          0651
5403 207752   0652 FNDALL   JSR FCHAR        ;FIND ALL DEFINED LABELS
5406 20DE54   0653          JSR FNDAL+3
5409 B0EC     0654          BCS FDCHAR+3     ;JMP ALWAYS
540B          0655
540B 20BF52   0656 FNDAL    JSR FC9-5        ;SEARCH FOR NEXT MATCH
540E 202F54   0657          JSR ENDCHK       ;TEST FOR CORRECT END
5411 90F8     0658          BCC FNDAL
5413 201E54   0659          JSR FCHK         ; ..AND BEGIN
5416 90F3     0660          BCC FNDAL
5418 60       0661          RTS
5419          0662
5419 20DB54   0663 FNDA     JSR FNDAL
541C B0D9     0664          BCS FDCHAR+3     ;JMP ALWAYS
541E          0665
```

```
541E 98       0666 FCHK   TYA              ;CHECK CORRECT BEGIN...
541F ED29A4   0667        SBC STIY+2       ;OF STRING (LABEL)
5422 A8       0668        TAY
5423 201DF9   0669        JSR SUB
5426 202F54   0670        JSR ENDCHK
5429 08       0671        PHP
542A 2028F9   0672        JSR ADDA-2
542D 28       0673        PLP
542E 60       0674        RTS
542F          0675
542F A20E     0676 ENDCHK LDX #14          ;THIS ROUTINE CHECKS...
5431 B1DF     0677        LDA (NOWLN),Y    ;WHETHER THE STRING IS...
5433 DD9854   0678 ENDCK  CMP CHRTBL,X     ;CORRECTLY TERMINATED
5436 F004     0679        BEQ RTCK         ;IF SO, CARRY IS SET
5438 CA       0680        DEX              ;OTHERWISE CARRY IS CLEAR
5439 10F8     0681        BPL ENDCK
543B 18       0682        CLC
543C 60       0683 RTCK   RTS
543D          0684
543D 206154   0685 CHNGS  JSR RSTDIB       ;RESTORE NEW LABEL
5440 A5F5     0686        LDA OLEN1        ;RESTORE LENTGHS OF LABELS
5442 85E9     0687        STA OLDLEN
5444 A5F7     0688        LDA LEN1
5446 85EA     0689        STA LENGTH
5448 A5DD     0690        LDA SAVPOS       ;START CHANGE
544A 48       0691        PHA
544B 202AF9   0692        JSR ADDA
544E 203FF9   0693        JSR REPLAC
5451 4CFA52   0694        JMP CHN21
5454          0695
5454 A6F7     0696 SAVDIB LDX LEN1         ;SAVE NEW LABEL
5456 CA       0697        DEX
5457 BD38A4   0698 SAVD1  LDA DIBUFF,X
545A 9D2001   0699        STA BUFFER,X
545D CA       0700        DEX
545E 10F7     0701        BPL SAVD1
5460 60       0702        RTS
5461          0703
5461 A6F7     0704 RSTDIB LDX LEN1         ;RESTORE NEW LABEL
5463 CA       0705        DEX



5464 BD2001   0706 RSTD1  LDA BUFFER,X
5467 9D38A4   0707        STA DIBUFF,X
546A CA       0708        DEX
546B 10F7     0709        BPL RSTD1
546D 60       0710        RTS
546E          0711
546E 20BCF8   0712 TOPLN  JSR TOPNO        ;GET TOP LINE
5471 4C0553   0713        JMP CLRLIN       ;RESET LINE COUNTER
5474          0714
5474 A5F7     0715 SFTYCK LDA LEN1         ;CHECK IF A "DELETE"...
5476 D0C4     0716        BNE RTCK         ;OF LABELS IN SOURCE...
5478 A205     0717        LDX #5           ;IS REALLY DESIRED
547A BD9254   0718 SFTY1  LDA MSG,X
547D 2002EF   0719        JSR OUTDP1       ;DISPLAY "SURE?"
5480 CA       0720        DEX
5481 10F7     0721        BPL SFTY1
5483 2073E9   0722        JSR REDOUT       ;GET "Y/N" AND ECHO
5486 C959     0723        CMP #'Y'         ;IF "YES", GO ON
5488 08       0724        PHP
5489 20F0E9   0725        JSR CRLF
548C 28       0726        PLP
548D F0AD     0727        BEQ RTCK
548F 4CE050   0728        JMP TXTEND       ;...ELSE TERMINATE
5492          0729
5492 203F     0730 MSG    .BYT ' ?ERUS'
5498 0D       0731 CHRTBL .BYT $D,$20,$3B,'#=()+,-<>"?.'
5499 20       0731
549A 3B       0731
5498 233D     0731
54A7          0732
54A7          0733        .END

ERRORS= 0000
```

#

Bernhard Kokula, 6700 Ludwigshafen

Prozeßtechnik mit Mikro-Computern (2)

Teil 2: Meßwert-Aufbereitung und Filterung

Der erste Teil beschrieb ein low cost A/D-Modul in Hard- und Software. Die Eingangsdaten im Meßumfang von 1 Byte wurden in Registern (HEXREG) abgelegt, die den Meßstellen zugeordnet sind. Die Meßstellenauswahl besorgt ein Pointer im X-Register. Die folgenden Module FILTER und HEXDEZ benutzen X ebenfalls als Meßstellenindex.

Bei höherer Genauigkeitsanforderung ist es ein Leichtes, außer dem Highbyte des Timers UT1CH auch das Lowbyte (UT1CL) zu verwerten. Die Meßzeit fällt ja als 16-Bit-Wort an. Allerdings bringt 1 Byte bereits eine Auflösung von 0,4%, also, was soll's?

Dieser Teil behandelt die Module

FILTER	ein programmierbares Tiefpassfilter
HEXDEZ	Hexadezimal-Umwandlung.

Die Module sind jedes für sich assemblierbar, die von ihnen benutzten Variablen sind ihnen unmittelbar zugeordnet. Zur Steuerung des Assemblers werden die Variablen- und die Programmblöcke jeweils durch Anweisungen wie VAR2=*, VAR3=* bzw. durch PRO2=*, PRO3=* abgeschlossen, die den augenblicklichen Stand des Zuordnungszählers aufnehmen. Die später hinzugelinkten Module können damit auf den jeweils erreichten Stand des Variablenbereiches bzw. des Programmbereiches Bezug nehmen und ihre Variablen dem Variablenbereich direkt (und ungestückelt) anschließen, ebenso Programme dem Programmbereich.

Das Modul 'TEST' darf für die Erprobung der beiden oben genannten Module nicht fehlen. Es erfordert die Eingaben:

Meßstellennummer	(00 bis 07 erlaubt), RETURN = REPEAT
Filterfaktor	(00, 03, 07, 0F .. auch 1F, 3F ....FF)
Neu-Hexdate	(00 - FF)

Um diesen Teil klein zu halten, wurde eine Prüfung auf gültige Daten nicht durchgeführt. Er liefert:

NEUHEX/HEXREG (nach Filter) = DEZREG.

Damit der gesamte Teil assemblierfähig ist, wurden ihm der Variablenblock aus Teil 1 sowie die Programm-Endadresse vorangestellt (siehe Listing: Variablenblock, dann *=PRO1, d.h. Anknüpfen an das Ende des Programmes aus Teil 1).

Sehr anschaulich wird der Test, wenn man die Analogausgabe 'ANTEST' aus Teil 1 noch vor dem JSR RCHEK einbaut, z.B. als JSR ANTEST. Da ANTEST das X-Register ändert, muß dieses gerettet werden. Bei dieser Art des Aufrufes (JSR) ist der Teil ANTEST weiterhin mit einem RTS abzuschließen, was in Teil 1 noch nicht vorgesehen war. Der Parameter ERST ist auf =0 zu setzen, er bestimmt den ersten der beiden analog auszugebenden Meßwerte.

Das Modul FILTER führt eine Mittelwertbildung durch, und zwar zwischen dem NEUHEXwert und dem zugehörigen Altwert im HEXREG nach der Formel:

$$\frac{\text{NEUHEX} + (n \times \text{HEXREG})}{n+1}$$

n kann dabei sein	00 entspr.	100% Neuwert, keine Filterung
	01	50%
	03	25%
	07	12,5%
	0F	6,25% (1F, 3F, 7F funktioniert auch)

Jeder Meßstelle kann einer dieser FILterFAKtoren individuell zugeordnet werden. Spätere Module werden sie verändern und FILTER als Integrator benutzen.

Das Modul HEXDEZ wandelt Daten aus den HEXREGistern in Dezimalwerte. Diese benötigen je 2 Byte für den Zahlenbereich 0000 - 0255. Der Assembler belegt wieder ja nach ZYKLUS (=Meßstellenzahl) die richtige Menge, und zwar zuerst alle LODEZ, sodann die HIDEZ. Dies ist nicht ganz fein,erspart es aber, das X-Register während der Zugriffe auf die DEZREGister zu verdoppeln. Es geht, liest sich nur unbequem.

Die HEX/DEZ-Wandlung ist von einfachster Art und nicht gerade schnell. Dafür kann man sie einfach verstehen: während HEXREG hexadezimal zu Null herunter gezählt wird, wird in den DEZREG im Dezimalmodus je Schleife eine '1' hinzuaddiert. Schneller geht es nach den Beispielen in Heft 2 und 3 dieser Zeitschrift (S. 29 bzw. 19 - Dr. Rüdiger), nach FUNKSCHAU 18/1979, S. 90, nach BYTE 8/1981, S. 413 (R. A. Young) oder nach der Routine in diesem Heft.

```
0003            PROZESSTECHNIK TEIL2
0003
0003            ;                                   10.11.81/KOKULA
0003            ;
0003            ;++++++++++ 'ALTE ZEROPAG-& VARIABLEN AUS TEIL 1'
0003 ZYKLUS             =8              ;FUER 1-8 MESSTELLEN
0003                    *=0
0000 HEXREG             *=*+ZYKLUS
0008 SHIFT              *=*+1           ;MESST.ABFRAGE-MASKE
0009 VAR1               =*              ;LINK ZUR WEITERVERWENDUNG
0009 PRO1               =$02BF          ;PROGRAMMENDE AUS TEIL 1
0009            ;
0009            ;
0009            ;
0009            ;++++++++++ 'PROGRAMMTEIL FILTER'
0009            ;
0009            ;TIEFPASS FILTER , FAKTOR VERAENDERLICH
0009            ;BEKOMMT BEIM AUFRUF DEN NEUWERT IM <A>
0009            ;HOLT SICH DEN ALTWERT UEBER <X>-MESSTELLEN-POINTER
0009            ;GIBT ERGEBNIS IM <A> ZURUECK
0009            ;<X> WIRD NICHT VERAENDERT, <Y> WIRD VERAENDERT
0009            ;
0009            ;---- 'NEUE ZEROPAGE-& VARIABLEN'
0009 FAK100=
0009                    0               ;FILTERFAKTOR: =100% NEUWERT
0009 FAK50              = 1             ;              = 50%     "
0009 FAK25              = 3             ;              = 25%     "
0009 FAK12              = 7             ;              = 12,5%   "
0009 FAK6               = 15            ;              = 6,25%   "
0009
0009                    *=VAR1          ;ZEROPAGE BELEGUNG AUS TEIL 1
0009 HISUM              *=*+1           ;HIBYTE SUMME
000A FILFAK             *=*+ZYKLUS      ;FILTEH FAKTOR JEDER MESSTELLE
0012 DIVFAK             *=*+1           ;TEMP. DIVISIONSFAKTOR
0013 UPDOWN             *=*+1           ;AENDERUNGSRICHTUNG
0014 VAR2               =*
0014            ;
0014            ;---- NUR ALS BEISPIEL GEDACHT !!!
0014                    *=FILFAK        ;EINSETZEN DER FAKTOREN
000A            00      .BYT FAK100     ;MESSTELLE 1, 100% NEUWERT
000B            01      .BYT FAK50      ;          2, 50%
000C            03      .BYT FAK25      ;U.S.W.
000D            07      .BYT FAK12,FAK6,FAK6,FAK100,FAK100
000E            0F
000F            0F
0010            00
0011            00
```

```
0012                    *=PRO1          ;PROGRAMMENDE AUS TEIL 1
02BF FILTER   A000      LDY #0
02C1          8409      STY HISUM       ;CLEAR TEMP. CARRY-SAVE
02C3          8413      STY UPDOWN      ;UP/DOWN-KORREKTUR
02C5          B40A      LDY FILFAK,X    ;HOLE FILTERFAKTOR
02C7          F01E      BEQ FI5         ;100% WEITERGEBEN
02C9          8412      STY DIVFAK      ;WIRD AUCH DIVISIONSFAKTOR
02CB          D500      CMP HEXREG,X    ;UP OR DOWN?
02CD          B002      BCS FI1
02CF          C613      DEC UPDOWN      ;WAR '0'
02D1                    ;
02D1                    ;---- ADDIERE ALTWERT ZU <A> <FILFAK>MAL
02D1 FI1      38        SEC             ;BEI 'NEU > ALT'
02D2          2413      BIT UPDOWN      ;NEU > ALT?
02D4          1001      BPL FI2         ;JA
02D6          18        CLC             ;BEI 'NEU < ALT'
02D7 FI2      7500      ADC HEXREG,X    ;ADDIERE ALTWERT ZU <A>
02D9          9002      BCC FI3         ;KEIN UEBERLAUF
02DB          E609      INC HISUM       ;FANG UEBERLAUF
02DD FI3      88        DEY
02DE          D0F1      BNE FI1         ;NOCHMAL?
02E0                    ;
02E0                    ;---- DIVIDIER SCHLEIFE
02E0 FI4      4609      LSR HISUM       ;LADE CARRY
02E2          6A        ROR A           ;TEIL <A>+CARRY DURCH 2
02E3          4612      LSR DIVFAK      ;/2
02E5          D0F9      BNE FI4         ;NOCHMAL?
02E7 FI5      60        RTS             ;ERGEBNIS LIEGT IM <A>
02E8 PRO2               =*
02E8                    ;++++++++++ 'PROGRAMMTEIL HEX/DEZ-WANDLER'
02E8                    ;
02E8                    ;<X>-MESSTELLEN-POINTER STEUERT DATENTRANSPORT
02E8                    ;DATENQUELLE SIND DIE 'HEXREG'
02E8                    ;DATENZIEL WERDEN DIE 'DEZREG'
02E8                    ;VERAENDERT WERDEN <A> & <Y>
02E8                    ;
02E8                            *=VAR2
0014 DEZREG                     *=*+ZYKLUS+ZYKLUS    ;JE 2 BYTE
0024 VAR3                       =*
0024                    ;
0024                            * =PRO2
02E8 HEXDEZ   A900      LDA #0
02EA          951C      STA DEZREG+ZYKLUS,X    ;CLEAR ZIELREGISTER-HI
02EC          B400      LDY HEXREG,X    ;HOLE HEX-DATE
02EE          F00C      BEQ HE3         ;0 BLEIBT 0
02F0          F8        SED             ;DECIMAL MODUS
02F1                    ;
02F1 HE1      18        CLC
02F2          6901      ADC #1
02F4          9002      BCC HE2         ;KEIN UEBERTRAG
02F6          F61C      INC DEZREG+ZYKLUS,X    ;HUNDERTER-UEBERTRAG
02F8 HE2      88        DEY             ;HEXDATE RUNTERZAEHLEN
02F9          D0F6      BNE HE1         ;NOCHMAL?
02FB          D8        CLD
02FC HE3      9514      STA DEZREG,X
02FE          60        RTS
02FF PRO3               =*              ;PROGRAMM ENDE TEIL 2
02FF                    ;
```

Die Subroutine FILTER ist im Programmteil MUX555 (Heft 20) nach EOR#$FF aufzurufen.

```
02FF            ;
02FF            ;
02FF            ;++++++++++ 'PROGRAMMTEIL TEST'
02FF            ;
02FF            ;FORDERT EINGABE VON:
02FF            ;MESSTELLENNUMMER <00-ZYKLUS>
02FF            ;FILTERFAKTOR     <00/01/03/07/0F> AUCH 1F/3F/7F
02FF            ;HEXDATE          <00-FF>
02FF            ;ERLAUBT MIT 'CR' ALTE DATEN WEITERZUVERARBEITEN.
02FF            ;'ESC' STOPPT PROGRAMM
02FF            ;KEINE PRUEFUNG AUF RICHTIGE EINGABE !!!
02FF                    *=$010C         ;USER START <F1>
010C            4CFF02  JMP TEST
010F            ;
010F                    * =PRO3
02FF TEST       203EE8  JSR BLANK
0302            205DEA  JSR RBYTE       ;HOLE <X>=MESSTELLE
0305            B012    BCS TE2         ;REPEAT
0307            AA      TAX
0308            2037E8  JSR PSL1        ;'/'
030B            205DEA  JSR RBYTE       ;HOLE FILTERFAKTOR
030E            950A    STA FILFAK,X
0310            20D8E7  JSR EQUAL
0313            205DEA  JSR RBYTE       ;HOLE NEUE HEXDATE
0316            8D4503  STA NEUHEX
0319            ;
0319            ;---- TESTERGEBNIS AUSGEBEN
0319 TE2        20F0E9  JSR CRLF        ;NEW LINE
031C            AD4503  LDA NEUHEX      ;INPUT DATE
031F            48      PHA
0320            2046EA  JSR NUMA        ;ZEIGE SIE
0323            2037E8  JSR PSL1        ;'/'
0326            68      PLA
0327            20BF02  JSR FILTER      ;TESTE FILTER
032A            9500    STA HEXREG,X
032C            2046EA  JSR NUMA        ;ZEIGE FILTER DATE
032F            20E802  JSR HEXDEZ      ;TESTE HEX/DEZWANDLER
0332            20D8E7  JSR EQUAL
0335            B51C    LDA DEZREG+ZYKLUS,X   ;DEZIMAL-HIBYTE
0337            2046EA  JSR NUMA
033A            B514    LDA DEZREG,X    ;LO
033C            2046EA  JSR NUMA
033F            2007E9  JSR RCHEK
0342            4CFF02  JMP TEST
0345            ;
0345 NEUHEX             *=*+1           ;SAVE TEMPORAER INPUT-HEXDATE
0346            ;
0346            ;
0346            ;---- AIM EQUATES
0346 BLANK              =$E83E
0346 CRLF               =$E9F0
0346 EQUAL              =$E7D8
0346 NUMA               =$EA46
0346 PSL1               =$E837
0346 RBYTE              =$EA5D
0346 RCHEK              =$E907
0346                    .END
```

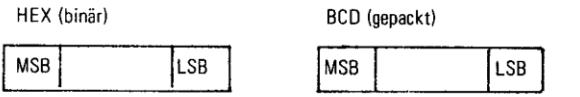
#

StDir. Peter Rix, 2350 Neumünster

Binär - BCD-Wandlung

1. Grundsätzliche Bemerkungen

Eine üblicherweise hexadezimal interpretierte Binärzahl wird in eine gepackte BCD-Zahl umgewandelt. Die Zahl ist positiv und ganz, sie steht wie angegeben in natürlicher Reihenfolge der Ziffern in zwei Feldern:



Zwischen den Feldlängen N und M gilt die Beziehung: $M \geqslant \frac{\ln 16}{\ln 10} N \cong 1.21\ N$

Das führt zu folgender Tabelle von Feldlängen:

N	1	2	3	4	5	6	8	10	12	14	16	18	20	24	28	32
M	2	3	4	5	7	8	10	13	15	17	20	22	25	29	34	39

Eine Zahl Z läßt sich in binärer Darstellung auf folgende Arten beschreiben:

1.) $Z = b_{i-1}b_{i-2}b_{i-3} \ \cdots\ b_2b_1b_0$

2.) $Z = b_{i-1}.2^{i-1} + b_{i-2}.2^{i-2} + \ldots\ b_2.2^2 + b_1.2 + b_0$

3.) $Z = (((((b_{i-2}.2 + b_{i-2}).2 + b_{i-3}).2 + ..)\ .\ b_2).2 + b_1).2 + b_0$

In der ersten Beschreibung steht die Zahl als Folge von Binärziffern im Speicher. Die zweite Form beschreibt die Zahl als Summe ihrer Stellenwerte. Die dritte Form ist eine für die numerische Rechnung besonders zweckmäßige Produktdarstellung, die schematisch ausgewertet werden kann und unter dem Namen 'Horner-Schema' bekannt ist.

Aus dem Horner-Schema ist bereits das Grundprinzip einer effektiven Umwandlung in eine BCD-Zahl abzulesen. Man addiert sukzessive eine Binärziffer (beginnend mit der signifikantesten Ziffer) und verdoppelt das jeweilige Zwischenergebnis. Führt man diese Operationen ***dezimal (!)*** aus, erhält man als Endergebnis die gesuchte BCD-Zahl.

2. Algorithmus

1. Lösche BCD-Feld
2. Lade Zähler mit I=8 x N
3. Verschiebe HEX-Feld 1 Bit nach rechts und dabei meist signifikantes Bit ins Carry
4. Addiere BCD-Feld mit Carry dezimal zu sich selber
5. Dekrementiere Zähler
6. Falls Zählerinhalt noch nicht Null, Fortsetzung bei 3
7. Ende, wenn Zählerinhalt Null.

Zentraler Teil des Algorithmus ist Schritt 4. Die Addition des Feldes mit sich selber bewirkt eine Multiplikation mit 2, durch den Carry-Inhalt wird gleichzeitig die nächste Binärziffer addiert.

Der Algorithmus dürfte als optimal hinsichtlich Codieraufwand und Laufzeit anzusehen sein. Schneller arbeiten nur Tabellenverfahren, die für Kleinsysteme kaum in Frage kommen. Die Laufzeit ist von der umzuwandelnden Zahl unabhängig, sie hängt nur von der Feldlänge N bzw. M ab.

3. HXBCD, universeller Algorithmus für den Prozessor 6502

Der Algorithmus ist als Unterprogramm mit frei verschiebbarem Maschinencode geschrieben. Die Feldlänge ist von N=1 bis N=32 ohne Programm-Neuentwurf einsetzbar. Im Sonderfall N=32 ist I=256 als 0 in das 8-Bit-Indexregister Y zu laden.

Wenn sich die Variablen in der Zeropage befinden, hat das Programm eine Länge von 35 Byte. Die Laufzeiten in Abhängigkeit von der Feldlänge (einschl. Rücksprung in das Hauptprogramm) sind in der Tabelle gelistet.

4. Modifikationen

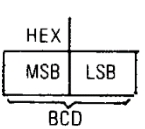
Der universelle Algorithmus ist für die bei Adreßumrechnungen und ähnlichen einfachen Aufgaben auftretenden Feldlängen (N=1 oder N=2) noch verkürzbar, weil Löschungen und Additionen hier nicht in Schleifen programmiert werden müssen. Da ein Verzicht auf Schleifen auch zu einer Laufzeitverkürzung führt, werden für N=1 und N=2 angepaßte Programmvarianten angegeben. HEX- und BCD-Felder überlappen einander wie angegeben und reduzieren dadurch auch den Speicherplatzbedarf für die Variablen. Gegenüber dem Universal-Algorithmus verkürzen sich die Ausführungszeiten noch einmal um mehr als die Hälfte.

```
0000 HEXBCD          =$300
0000
0000 N               =16
0000 I               =128
0000 M               =20
0000
0000                 *=$10
0010 HEX             *=*+N
0020 BCD             *=*+M
0034
0034                 *=HEXBCD
0300          F8     SED              ;DEZIMAL-MODUS SETZEN
0301          A900   LDA #00
0303          A213   LDX #M-1
0305 AAA      9520   STA BCD,X        ;LOESCHEN
0307          CA     DEX              ;BCD-FELD
0308          10FB   BPL AAA
030A          A080   LDY #I           ;BITZAEHLER LADEN
030C BBB      A20F   LDX #N-1
030E CCC      3610   ROL HEX,X        ;LINKSSCHIEBEN
0310          CA     DEX              ;HEX-FELD
0311          10FB   BPL CCC
0313          A213   LDX #M-1
0315 DDD      B520   LDA BCD,X        ;VERDOPPELN BCD-FELD
0317          7520   ADC BCD,X        ;MIT CARRY-ADDITION
0319          9520   STA BCD,X
031B          CA     DEX
031C          10F7   BPL DDD
031E          88     DEY
031F          D0EB   BNE BBB NAECHSTES BIT
0321          D8     CLD              ;DEZIMAL-MODUS LOESCHEN
0322          60     RTS
0323                 .END
```

Tabelle: Ausführungszeiten für 1 MHz-CPU

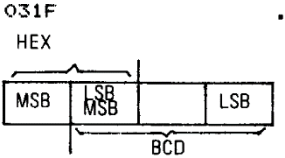
N	M	T (msek)	
1	2	0,454	
2	3	1,327	
3	4	2,648	für
4	5	4,417	Programm
5	7	7,323	
6	8	10,124	HEXBCD
8	10	17,070	
10	13	27,177	
12	15	37,979	
14	17	50,573	
16	20	67,144	
18	22	83,596	
20	25	104,566	
24	29	146,970	
28	34	200,363	
32	39	262,011	

```
0000 HXBCD1          =$300
0000
0000                 *=$10
0010 HEX
0010 BCD             *=*+2
0012
0012                 *=HXBCD1
0300          F8     SED
0301          A900   LDA #00
0303          8511   STA BCD+1
0305          A008   LDY #8
0307          18     CLC
0308 BBB      A511   LDA BCD+1
030A          6511   ADC BCD+1
030C          8511   STA BCD+1
030E          2610   ROL HEX
0310          88     DEY
0311          10F5   BPL BBB
0313          D8     CLD
0314          60     RTS
0315                 .END
```



HXBCD1
für Feldlängen N=1, M=2
Ausführungszeit 0,195 msek

```
0000 HXBCD2          =$300
0000
0000                 *=$10
0010 HEX             *=*+1
0011 BCD             *=*+3
0014
0014                 *=HXBCD2
0300          F8     SED
0301          A900   LDA #00
0303          8512   STA BCD+1
0305          8513   STA BCD+2
0307          A010   LDY #16
0309          18     CLC
030A BBB      A513   LDA BCD+2
030C          6513   ADC BCD+2
030E          8513   STA BCD+2
0310          A512   LDA BCD+1
0312          6512   ADC BCD+1
0314          8512   STA BCD+1
0316          2611   ROL BCD
0318          2610   ROL HEX
031A          88     DEY
031B          10ED   BPL BBB
031D          D8     CLD
031E          60     RTS
031F                 .END
```



HXBCD2
für Feldlängen N=2, M=3
Ausführungszeit 0,588 msek

Druckerausgabe auf parallele Schnittstelle

Fast jeder Systembetreiber hat den Wunsch, einen Drucker an seinen Rechner anzuschließen. Wenn wir einmal von Sonderausführungen absehen, so lassen sich Drucker u.a. nach der zu verwendenden Schnittstelle klassifizieren. Bekannt sind die Centronix-Schnittstelle mit Datentransport auf 8 parallelen Leitungen und 2 Handshake-Leitungen, ferner die IEEE 488-Schnittstelle (ebenfalls mit 8 Datenleitungen, 3 Handshake-Leitungen und weiteren Steuerleitungen, benutzt u.a. am CBM) sowie die serielle TTY-Schnittstelle mit nur 2 bis 4 Drahtverbindungen.

Das nachstehende Programm ist für eine Schnittstelle der ersten Art geschrieben, für eine 8-Bit parallele Schnittstelle also. Zu druckende ASCII-Zeichen werden auf den Port B geschrieben. Durch die Initialisierung des PCR (Peripheral Control Register) mit hex A0 ist sichergestellt, daß an der Leitung CB2 nach dem Beschreiben des Portes B automatisch ein kurzer Strobe-Impuls entsteht, der dem Drucker das Bereitstehen eines neuen Zeichens signalisiert. Als langsames Peripheriegerät antwortet der Drucker nach einiger Zeit mit ACKNOWLEDGE und/oder negativ gehendem Impuls an CB1, daß er das Zeichen abgenommen hat. Das nachstehende Programm verwendet diesen Impuls von BUSY.

Man beachte, daß seitens des Druckers ein Auto-Linefeed erwartet wird, sobald der Rechner ein CR (Carriage Return) sendet. Im allgemeinen wird man diesen automatischen Zeilenvorschub am Drucker einstellen können. Wenn das nicht möglich ist, muß man an der Programmstelle WRI1 auf CR prüfen und beim Auftreten desselben ein hex 0A als Zeilentransport dazuschießen.

Das Programm ist für alle 6502-Rechner verwendbar wenn man die eine hier berücksichtigte Eigenheit des AIM 65/PC 100 ggfs. fortläßt: Bei Ausgabe über den USER-Vektor liegt das ASCII-Zeichen auf dem Stack, es wird also nicht wie üblich im Akku übergeben. Daher das PLA. Das anfängliche BCS WRITE bewirkt die Auftrennung der Routine in einen initialisierenden Teil und in einen die Zeichen transportierenden.

```
0000            ;--------------------
0000            ;DRUCKERAUSGABE
0000            ;FUER 8-BIT PARALLELE SCHNITTSTELLE
0000            ;MIT 2 HANDSHAKE-LEITUNGEN
0000            ;NAEMLICH STROBE UND BUSY
0000            ;--------------------
0000            ;SYMBOLE
0000            ;ADDRESS THE 6522 VIA CHIP
0000 VIA                =$A000          ;AIM 65 USER VIA
0000 PORTB              =VIA            ;OUTPUT PORT
0000 DDRB               =VIA+2          ;DATA DIRECTION
0000 PCR                =VIA+$C         ;CONTROL OF HANDSHAKE
0000            ;CB2 STROBE OUTPUT IN PULSE MODE
0000            ;CB1  INPUT OF BUSY FROM PRINTER GOING LOW
0000 IFR                =VIA+$D         ;INTERRUPT FLAG REGISTER
0000
0000 CR                 =$D             ;CARRIAGE RETURN
0000 LF                 =$A             ;LINEFEED
0000            ;--------------------
0000            ;MAIN PROGRAM
0000            ;USER OUTPUT ROUTINE ON AIM 65
0000            ;ADDRESS VECTOR OF PAROUT MUST RESIDE IN $10A/10B
0000
0000                    *=$200
0200 PAROUT
0200            B00E    BCS WRITE
0202            A9FF    LDA #$FF        ;INITIALIZE PORT
0204            8D02A0  STA DDRB
```

```
0207          A9A0    LDA #%A0        ;REACT ON NEG. TRANSITION OF BUSY
0209          8D0CA0  STA PCR         ;& PULSE CB2 AFTER WRITING PORTB
020C          A90D    LDA #CR         ;ADVANCE TO NEW LINE
020E          D001    BNE WRI1        ;BRANCH ALWAYS
0210
0210 WRITE    68      PLA             ;GET CHARACTER FROM STACK
0211 WRI1     8D00A0  STA PORTB       ;WRITE TO PORT
0214          A910    LDA #%10        ;CHECK FOR BUSY
0216 WRI2     2C0DA0  BIT IFR         ;HAS CB1 BEEN ACTIVE?
0219          F0FB    BEQ WRI2        ;NO YET
021B          60      RTS             ;RETURN TO CALLER
021C                  .END                                    R.L. #
```

6502 Multiplikation, Division

Die CPU 6502 kann bekanntlich mit dem Befehl SED (Set Decimal Mode) für die Befehle ADC SBC, Addition und Subraktion, in den dezimalen Rechenmodus versetzt werden. Diese Fähigkeit ist nach wie vor außergewöhnlich, gleichwohl ist sie bei Sprachimplementierungen, wie z.B. Microsoft-BASIC, nicht ausgeschöpft worden, und es wurde in den Veröffentlichungen nur selten vom Dezimalmodus Gebrauch gemacht. Eine große Ausnahme bildet das im 65xx MICRO MAG von M. Zimmermann veröffentlichte ASP (Advanced Subroutine Package, Hefte 2 bis 6 und Heft 9, bzw. 'Das Buch 1-6 des 65xx MICRO MAG' für die ersten fünf Folgen). Anregungen für die Programmierung von Addition und Subtraktion für verschieden breite Datenfelder wurden auch in Heft 5 gegeben ('Ein Leitfaden für die Programmierung, Teil 2'). Weitere Veröffentlichungen betrafen das KIMath und einen Aufsatz von M. Konz in der FUNKSCHAU 19/1979 '6502-Rechenroutinen'. Den beiden letztgenannten ist gemeinsam, daß sie, um multiplizieren und dividieren zu können, ein eigentlich nur als unglücklich und aufwendig zu bezeichnendes Rechenformat herstellen müssen.

Zahlen- und Formatwandlungen sind zeit- und speicherplatzaufwendig. Bei Multiplikation und Division hat man aus Gründen, die in der Logik des binären Rechnens liegen, weiterhin ausgiebig davon Gebrauch gemacht, die Operandenfelder bitweise zu verschieben. Nun sind Verschiebebefehle im Memory die zeitaufwendigsten (5-6 Maschinenzyklen bei Adressierung der Zeropage, 6-7 Zyklen für absolut, bzw. absolut,X). Um z.B. ein Datenfeld von 4 Byte= 32 Bit um eine Bitposition im Memory zu verschieben, benötigt man 4x7=28 Zyklen als Minimum. Wenn man um die Breite einer BCD-Ziffer (=4 Bit) verschiebt, sind es also 4x28 Zyklen = 112 Zyklen, wenn man wie üblich, die mit X indizierte Adressierungsart wählt. Bei Rechenroutinen sind also Verschiebeoperationen möglichst zu vermeiden, auch wenn sie dem evtl. trägen memory des Programmierers besser konvenieren sollten. Verschiebungen im Akku kosten nur 2 Zyklen. Bei unumgänglichen Verschiebungen sollte man das also im Akku tun.

Die Nicht-Verwendung des Dezimalmodus im landläufigen BASIC hat auch zu anderen Auswegen geführt, wie z.B. im Artikel 'Kaufmännische Rechnen auf dem CBM' In Heft17. Dort werden Stringvariable verarbeitet, die BCD-numerische ASCII-Zeichen enthalten. Das war zugegebenermaßen eine Notlösung.

BCD-Zahlen dürfen auch für eine 8-Bit–CPU als außerordentlich freundlich angesehen werden. Die übliche Eingabe und Ausgabe erfolgt ja im ASCII-Code. Der Zahlenwert 24 kommt als ASCII 32 und 34 an. Wenn man die führenden Dreien wegblended und einmal schiebt, hat man ein gepacktes Byte, das die Ziffern 24 enthält. Das Entpacken für die Ausgabe verläuft entsprechend: Jedes Halbbyte wird mit einer führenden 3 versehen und ist damit reif für die ASCII-Ausgabe. Man benötigt also bei einer BCD-mäßigen Zahlendarstellung in den Bytes keinen komplizierten Algorithmus für

die Binär-zu-BCD-Wandlung. Gleichwohl soll man nicht verkennen, daß ein binäres Rechnen für Adressnrechnen unentbehrlich ist. Weiterhin soll man bedenken, daß die Monitorprogramme im allgemeinen den binären Modus voraussetzen. BCD-Rechenroutinen sind daher grundsätzlich mit CLD (Clear Decimal Mode) zu verlassen, um den Rechner nicht in den Traumzustand zu entlassen.

Kaufmännisches Rechnen verlangt vor allem für große Beträge und für Tendenzrechnungen an großen Beträgen (prozentuale Veränderungen) ein Rechnen in BCD-Zahlen. Der Programmierer muß in der Lage sein, die Breite der Datenfelder für die Operanden ausreichend für die Rechengenauigkeit bemessen zu können. Die logisch schwierigen Routinen sind dabei die der Multiplikation und die der Division für unterschiedliche Feldbreiten. Die vorliegenden Lehrbücher handeln 'zum Beispiel für den Programmierer' immer nur eine Arbeitsbreite bis zu 16 Bit = 2 Byte ab. Der allgemeine Algorithmus hat gefehlt. Addition und Subtraktion dürfen mit den zuerst zitierten Quellen als ausreichend abgehandelt betrachtet werden.

Die nachfolgenden Programme für Multiplikation und Division waren mit dem Ziel auf eine beliebige Breite der Operanden und auf zeitliche Optimierung nicht ganz schmerzlos zu entwickeln. Es sollten Kurzwegtechniken benutzt werden, wann immer die Operanden das zulassen. Verschiebebefehle werden möglichst nur im Akku oder in einem Hilfsbyte durchgeführt. Die Routinen setzen voraus, daß der Programmierer die Operanden und ihr Längenattribut bereits in die Zeropage befördert hat, wo sie dann verarbeitet werden. Natürlich kann man die Adressierung auch auf den absoluten Adreßraum umstellen, das würde jedoch zusätzliche Ausführungszeiten bedingen, könnte jedoch Datentransporte ersparen.

Multiplikation und Division werden hier nicht in der Einbindung in ein 'package' dargestellt, sondern mehr als ein Workshop-Abzug, der dem Leser ohne Erschwernisgrade weitere Gestaltungen erlaubt. Die 'allocations' in der Zeropage, die Speicherplatzzuweisungen also, gehen davon aus, daß der Benutzer nicht mit mehr als 14 Digits arbeiten möchte (zusammen mit dem Längendescriptor =8 Bytes). Das dürfte im allgemeinen ausreichen. Wenn größere Genauigkeit gewünscht wird, sollte man die Speicherbereiche der Operanden weiter auseinanderziehen. Eine Rundung der Ergebnisse sollte nach den Bedürfnissen des Programmes vorgenommen werden. Bei kaufmännischen Rechnungen geht man im allgemeinen von 2 Nachkommastellen aus. Die Verwaltung des Dezimalkommas liegt in der Verantwortung des Programmierers.

Ein wichtiger Hinweis ist für das Längenattribut der Operanden zu geben von LDEND, LSOR, LFAC1 und LFAC2: Es enthält die wirkliche Länge (Menge der Datenbytes) minus 1. 00 entspricht also der Länge 1, 03 der Länge 4 usw.. Das Längenattribut wird gleichsinnig auch für die Ergebnisse bei LP, LREST und LQ verwendet. Beim Quotienten und beim Divisionsrest wird ab NORM1 eine Normalisierung durchgeführt, führende Bytes, die 00 enthalten, werden abgeschnitten. Im vordersten Byte kann dann gleichwohl noch die führende Ziffer 0 enthalten sein.

Es ist zu erwähnen, daß die Routinen sowohl im binären als auch im dezimalen Modus einwandfreie Ergebnisse abliefern. Bei der Multiplikation bestimmt der Einsprungspunkt M0 oder M01 die Rechenweise. Bei der Division ruft man entweder auf JSR 0300 und erhält binäre Ergebnisse oder man ruft auf SED JSR 0300 und erhält dezimale. Man sollte allerdings auch bei der Multiplikation den beim Aufruf gesetzten Modus so wie bei der Division anfangs retten und später wieder aufgreifen, denn die Längenberechnung für das Produkt (LP) sollte auch für lange Operanden binär richtig erfolgen.

```
0000            ;---------------------
0000            ;BCD-MULTIPLIKATION
0000            ;WAHLWEISE BINAERE MULTIPLIKATION
0000            ;BEI JSR M01
0000            ;ANWENDBAR AUF BELIEBIG LANGE FELDER
0000            ;V09.11.81
0000            ;COPYRIGHT BY ROLAND LOEHR
0000            ;---------------------
0000            ;SYMBOLE UND DEFINITIONEN
0000            ;IT IS SUPPOSED THAT THE LENGTH OF FAC1 AND FAC2 ARE
0000            ;REAL LENGTH -1, BINARY
```

Inhaltsverzeichnis

Jahrgang 1981 - Hefte Nr. 17-22

	Heft-Seite
Allgemeine Themen	
Datenaustausch zwischen zwei Mikroprozessorsystemen	17-27
Rechnerkopplung mit Interrupt	17-31
Die (Un-) Zuverlässigkeit von Kassettenspeichern	17-32
Hannover-Messe 1981	18-47
Einkommensteuerberechnung 1980	19-11
PRINT-Formatierung	19-35
Prozeßtechnik mit Mikro-Computern (1)	20- 3
Einkommensteuerberechnung 1981	21-46
Feedback	21-47
Prozeßtechnik mit Mikro-Computern (2)	22-22
BASIC-Formatierungen	22-57
6502	
Pseudo 16-Bit CPU	18-18
APPLE II emuliert AIM 65	20-31
UNITEX - Universale Textausgabe	21-30
Binär-BCD-Wandlung	22-27
Druckerausgabe auf parallele Schnittstelle	22-30
6502 Multiplikation, Division	22-31
6809	
MC 6809: Register, Signale, Befehle	18- 3
MC 6809: Befehle (1)	19-20
MC 6809: Befehle (2)	20-50
Adressierungsarten des 6809	22-42
CBM und PET	
PASCAL für CBM	17- 3
Kaufmännisches Rechnen auf dem CBM	17- 9
Grafik-Zugriff beim CBM-Busy	17-14
1/4-Grafik für CBM 3001	17-15
CBM 3032: Benutzung arithmetischer Interpreterroutinen	17-44
ON ERROR GO TO	18-16
Garbage Collection Routinen im CBM-BASIC	18-27
Assoziative Tabellen (CBM)	18-29
Direktzugriff mit CBM	18-45

Nützliche Dokumentationen für PET und CBM	18-50
SUPER LIST CBM/Centronix	18-51
Schnelle Sortierroutine für CBM 3001	19- 3
REPEAT für den CBM 3001	19-32
Zeitanzeige PET/CBM	19-36
1/4-Grafik (CBM)	19-47
SCREENROLL (CBM)	19-52
Cross Reference List für BASIC-Variable (CBM)	20-10
Zweidimensionale Felder sortieren (CBM)	20-12
Der Datenverkehr Rechner und CBM-Floppy 4040	21-10
PRINTUSING für CBM	21-19
ROM-Test für CBM 3001	21-33
Einfache Sprachausgabe mit Kleinrechner	21-43
ERASE rechts vom Cursor	21-61
Mischbilder vom PET und einer Video-Kamera	22-38
Brechnung von Pi mit großer Genauigkeit	22-41
CBM-Math	22-50
Breite Monitorausgabe für CBM 8032	22-56
Disk-APPEND	22-57

AIM 65/PC 100

AIM 65 mit 'fremder' Systemsoftware	17-17
FASTLINK	17-24
Komplette BASIC-Statements durch CTRL und Tastendruck	17-35
LMR, LOAD, MOVE, RELOCATE	17-38
Laufzeitmessung für Programme	18- 9
SEARCH	18-34
Assembler Object Deplacer	18-36
CHANGE To End	18-38
Tape-Dupe	18-40
AIM FORTH V1.3	19-18
FORTH Disassembler	19-24
New Single Step	19-30
Plotten mit dem AIM 65	19-33
AIM 65 am IEEE488-Bus	19-38
EPROM-Programmiereinheit für AIM 65	20-18
Softwareentwicklung für AIM 65 auf PDP 11	20-26
Object Code Editor	20-41
AIM Spezial (9)	20-61
AIM mit Floppy Disk CBM 4040	21- 3
Uhr und Kalender	21-36
Ein- und Ausgabe am AIM 65 (4)	21-51
Monitor-Erweiterung AIM 65	21-53
AIM Spezial (10)	21-58

Sortieren mit dem AIM	22- 3
LINED	22-11
AIM Spezial (11)	22-54
Produktbesprechung	
Ein 6809-System	18-43
DAIM Floppy Disk-System für den AIM 65	18-46
Matrixdrucker EPSON MX80FT	19-49
12K BASIC für AIM 65/PC 100	19-50
Buchbesprechung	17-47 20-59 22-55
Software-Besprechung	17-46 20-59 21-41
Editorial	17-49 18-57 19-60 20-61 21-59 22-56
Aus der Branche	19-59 21-60
Produkte	20-60

Im Januar 1982 erscheint:

Das Buch 7-13 des

65xx MICRO MAG

Zusammenfassung der Hefte in der ursprünglichen Gliederung
als ansprechendes fesgebundenes Buch
mit Gesamtinhaltsverzeichnis und jeweiligen Heftinhaltsverzeichnissen
ohne Werbeseiten insges. ca. 340 Seiten, DM 42,-- Endpreis

Die Inhaltsübersicht ist umseitig abgedruckt

Inhaltsverzeichnis

Das Buch 7-13 des 65xx MICRO MAG

Allgemeine Themen

Datenaustausch zwischen KIM-1 und TRS 80	7-28
Erzeugung 'quasistatistischen Rauschens' durch Pseudo-Zufallszahlenfolgen	7-43
Zufallszahlengenerator	8-35
Magnetbandbetrieb	9-32
Umbau eines Cassettenrecorders	9-35
Anschluß von numerischer Anzeige und Tastatur (1)	11-31
Interruptgetriebene Cassetten-ein- und -ausgabe (1)	12-23
Anschluß von numerischer Anzeige und Tastatur (2)	12-32
Wie liest man ein Programm-Listing?	13- 8
Siemens PC 1000	13-17
Anschluß von numerischer Anzeige und Tastatur (3)	13-20
BASIC 'DATA'-Generator	13-33
Interruptgetriebene Cassettenein- und -ausgabe (2)	13-43

6502

Das VIA 6502	7- 3
ASP - Advanced Subroutine Package (6)	9-22
Align für ALPHA-SORT	9-28
Ein Leitfaden für die Programmierung (4)	9-41
Interrupt-Demonstrationsprogramme für das VIA 6522	10-24
Ein Leitfaden für die Programmierung (5)	10-30
Geschachtelte Interrupts	12-28

CBM und PET

Der PET-Assembler	7-37
PET Video-Driver	7-38
Primfaktoren-Zerlegung (PET)	8- 3
Video-Edit (PET)	8-11
PET-Petits	8-14
TRACE (PET)	9- 3
VIEW (PET)	9 -5
Text-Editor (PET)	9- 6
PETROL - Bilschirm rollen	9- 9
RESTORE Line Number (PET)	9-13
Datenverbund zwischen AIM 65 und PET 2001	9-14
CBM-VIEW - Neue Befehle	10-20
Berechnetes GOTO für den CBM	10-21
Tape Catalog (CBM)	10-41
Disk Utility Program (CBM)	11- 6
6502 Direct Assembler (PET)	11-12
Schaufel (RELOCATE, PET)	11-16
VARLIST - Variablenausdruck (PET)	11-17
CBM schießt sich eigene EPROMs	11-24
ROM-Vergleichsliste CBM-PET	11-28
BASIC Keywords to Shifted Keys	12- 3
CBM UNIPLOTT	12-48
Der CBM-Assembler	13-18
Binäres Speichern von Zahlen	13-27
BLANK-DELETER	13-29

AIM 65/PC 100

Der AIM-Assembler	7-14
AIM-BASIC	7-20
MOVE and RELOCATE	7-24
AIM 65 als Terminal	7-33
AIM Spezial (2)	7-34
Gedanken zum Video-AIM	8-16
Oszillograph als Bildschirm für den AIM 65	8-36
Auskunftssystem mit dem AIM 65	8-36
AIM-Tastatur mit Kleinschreibung	8-39
AIM Spezial (3)	8-41
AIM Spezial (4)	9- 8
Datenverbund zwischen AIM 65 und PET 2001	9-14
Assembler-Listing für den AIM 65	9-17
KOORD, Plotten mit dem AIM-Printer	9-29
Auskunftssystem mit dem AIM 65 (2)	9-37
AIM Spezial (5)	10- 3
MTEST, Speicherprüfung mit Zufallszahlenmustern	10-11
Binärdisplay-Programm	10-15
Listing der Assembler-Symboltafel	10-16
Vorstellung des Siemens PC 100	10-18
PRINT in 60 Spalten	10-22
TVINT - Systemintitialisierung	10-29
Erzeugung eines 'kalten' RESETs beim AIM	10-38
BASIC-Erweiterung für AIM 65/PC 100	10-40
MLIST	11- 3
AIM Spezial (6)	11-22
Ein Printer für den AIM	11-23
USCOM - User Defined BASIC-Commands	11-37
Erweiterung des AIM 65 auf den S-100-Bus	11-39
Cross Reference Table für den AIM 65-Assembler	12- 9
AIM Spezial (7)	12-43
NUMBR	12-45
Datenein- und -ausgabe für das AIM-BASIC	13- 3
User Defined BASIC-Commands V2.1	13- 8
AIM Spezial (8)	13-52

Produkte

Video-News	8-32
Das VIDEO+	11-41
Superboard CHALLENGER II	11-44
Die Challengers von OSI	11-16
NEWTIMs für CBM	13-26
Die neue CBM-Serie 8000	13-41

Literaturhinweise 7-48

Weitere Buchangebote

```
0000           ;MSD= MOST SIGNIFICANT DIGIT
0000           ;LSD= LEAST SIGNIFICANT DIGIT
0000           ---------------------
0000           ZEROPAGE USED FOR ALL TRANSACTIONS
0000           ;OPERANDS ARE SUPPOSED TO BE HERE
0000           ;RESULTS ARE LEFT HERE IN LP AND PROD
0000                  *=$12
0012 CARRY1           *=*+1 COUNT CARRIES OF CURRENT DIGIT
0013 CARRY2           *=*+1          ;HOLD CARRIES FROM LSD-MULT
0014 FLAG             *=*+1          ;00 INDICATES LSD, FF=MSD
0015 TEMP             *=*+1          ;TEMPORARY FOR FAC2-BYTE
0016 LPR              *=*+1          ;RESIDUAL LENGTH OF PRODUCT
0017 LFAC1R           *=*+1          ;RESID. LENGTH OF FAC1
0018           ---------------------
0018           LOCATIONS OF FAC1
0018 LFAC1            *=*+1          ;LENGTH OF FAC1, BYTE COUNT-1
0019 FAC1             *=*+7          ;THE DIGITS OF FAC1
0020           ---------------------
0020           LOCATIONS OF FAC2
0020 LFAC2            *=*+1          ;LENGTH OF FAC2, DECREM. DURING
0021 FAC2             *=*+7          ;THE DIGITS OF FAC 2      MULT
0028           ---------------------
0028           LOCATIONS OF PRODUCT
0028 LP               *=*+1          ;LENGTH OF PRODUCT -1
0029 PROD             *=*+10         ;ACCEPTS THE RESULT
0033           ---------------------
0033           MAIN PROGRAM
0033                  *=$200
0200 M0       F8      SED            ;DECIMAL MODE
0201 M01
0201          A518    LDA LFAC1      ;LENGTH
0203          8517    STA LFAC1R     ;STORE TO RESIDUAL LENGTH
0205          38      SEC            ;LP ALWAYS +1
0206          6520    ADC LFAC2
0208          8528    STA LP         ;LENGTH OF PRODUCT
020A          8516    STA LPR        ;RESIDUAL FOR ADDRESSING
020C          AA      TAX
020D          ---------------------
020D          CLEAR PRODUCT
020D          A900    LDA #0
020F M1       9529    STA PROD,X
0211          CA      DEX
0212          10FB    BPL M1
0214
0214
0214 M1B      A900    LDA #0
0216          8514    STA FLAG       ;INDICATE LSD
0218          8513    STA CARRY2     ;NO CARRIES
021A
021A          A620    LDX LFAC2
021C          B521    LDA FAC2,X     ;GET BOTH DIGITS OF MULTIPLIER
021E          F064    BEQ NXTDI1     ;BYTE IS 00
0220          8515    STA TEMP
0222 M2       290F    AND #$0F       ;MASK OFF MSD
0224          F026    BEQ SHORT2
0226 M3       A8      TAY            ;Y CONTAINS MULTIPLIER DIGIT
0227          A617    LDX LFAC1R     ;ADDRESS BYTE OF FAC1
0229          A900    LDA #0         ;START EMPTY
```

```
022B          8512    STA CARRY1      ;NO CURRENT CARRY
022D          D519    CMP FAC1,X      ;SEE IF ZERO
022F          F063    BEQ SHORT1
0231          18      CLC             ;MULTIPLICATION MAIN LOOP
0232 M4       7519    ADC FAC1,X      ;X=ADDRESSER, Y=COUNTER
0234          9003    BCC *+5
0236          E612    INC CARRY1      ;COLLECT THE CURRENT CARRIES
0238          18      CLC
0239          88      DEY
023A          D0F6    BNE M4
023C          A616    LDX LPR         ;ADDRESS THE PRODUCT RANGE
023E          A414    LDY FLAG        ;SEE WHICH DIGIT WAS PROCESSED
0240          D018    BNE DIGIT2
0242          7529    ADC PROD,X      ;GET THE ALREADY DEPOSITED
0244          9529    STA PROD,X                      RESULTS
0246          A512    LDA CARRY1
0248          6900    ADC #0
024A          8513    STA CARRY2      ;SAVE CARRIES FROM LSD
024C SHORT2
024C          C614    DEC FLAG        ;FLAG=FF TO INDICATE MSD
024E          A515    LDA TEMP        ;ISOLATE MSD
0250          4A      LSR A
0251          4A      LSR A
0252          4A      LSR A
0253          4A      LSR A
0254          D0D0    BNE M3
0256          8512    STA CARRY1
0258          F00C    BEQ DIGAD
025A
025A                  --------------------
025A          MULTIPLY RESULT OF MSD BY 10
025A          ;TO ADD IT INTO THE RIGHT DIGIT OF PRODUCT
025A          ;MSD OF RESULT ROLLED INTO CARRY1
025A DIGIT2   0A      ASL A
025B          2612    ROL CARRY1
025D          0A      ASL A
025E          2612    ROL CARRY1
0260          0A      ASL A
0261          2612    ROL CARRY1
0263          0A      ASL A
0264          2612    ROL CARRY1
0266
0266 DIGAD
0266          C616    DEC LPR
0268          7529    ADC PROD,X      ;RESULT OF LSD
026A          9529    STA PROD,X
026C
026C          A513    LDA CARRY2      ;CARRIES OF LSD
026E          6512    ADC CARRY1      ;CURRENT CARRIES LEFTMOST AND MSD
0270          CA      DEX
0271          7529    ADC PROD,X
0273          9529    STA PROD,X      ;DEPOSIT FORMER RESULT
0275          9009    BCC NEXTDI      ;TO WORKING LOCATION
0277 DIGITL   CA      DEX
0278          A900    LDA #0
027A          7529    ADC PROD,X
027C          9529    STA PROD,X
027E          B0F7    BCS DIGITL
```

```
                        ;-------------------------
0280                    DECIDE FOR NEXT MAIN LOOP
0280 NEXTDI
0280            C617    DEC LFAC1R      ;MORE DIGITS OF FAC1?
0282            1090    BPL M1B         ;YES
0284 NXTDI1
0284            C620    DEC LFAC2       ;MORE FROM FAC2?
0286            3011    BMI OUT         ;ALL DONE
0288            A518    LDA LFAC1       ;RESTORE FULL LENGTH OF FAC1
028A            8517    STA LFAC1R
028C            38      SEC
028D            6520    ADC LFAC2
028F            8516    STA LPR
0291            4C1402  JMP M1B
0294
0294 SHORT1
0294            C616    DEC LPR         ;THIS BYTE OF FAC1 IS DONE
0296            4C8002  JMP NEXTDI
0299 OUT        D8      CLD
029A            60      RTS
029B                    .END

                        ;-------------------------
0000                    BCD-DIVISION
0000
0000                    ;FUER BELIEBIG LANGE ZAHLENFELDER
0000                    ;V13.11.81
0000                    ;REC. DIV14
0000                    ;COPYRIGHT BY ROLAND LOEHR
0000
0000
                        ;-------------------------
0000                    ZEROPAGE USED FOR ALL OPERANDS
0000                    ;LENGTH ATTRIBUTES: REAL LENGTH -1
0000                            *=$10
0010 CARRY                      *=*+1
0011 MODE                       *=*+1           ;ACCEPTS BINARY/DECIMAL MODE
0012
0012
                        ;-------------------------
0012                    LOCATIONS OF DIVIDEND
0012                            *=$18
0018 LDEND                                      ;LENTH OF DIVIDEND
0018 LREST                      *=*+1           ;EQUATES LDEND
0019
0019 DEND
0019 REST                       *=*+7           ;MANTISSA OF DIVIDEND AND REST
0020
0020
                        ;-------------------------
0020                    LOCATIONS OF DIVISOR
0020 LSOR                       *=*+1           ;LENGTH OF DIVISOR
0021 SOR                        *=*+7           ;MANTISSA OF DIVISOR
0028
0028
                        ;-------------------------
0028                    LOCATIONS OF QUOTIENT
0028 LQ                         *=*+1           ;LENGTH-1
0029 Q                          *=*+7           ;MANTISSA OF QUOTIENT
0030
0030
                        ;-------------------------
0030
0030                            *=$300
0300            08      PHP                     ;WHAT MODE TO BE USED LATER?
```

```
0301          68      PLA             ;GET IST TO SAVE IT
0302          8511    STA MODE
0304          D8      CLD
0305
0305 D1       A521    LDA SOR         ;GET FIRST BYTE
0307          D015    BNE DLEN        ;IT IS A VALID NUMBER
0309          AA      TAX             ;X=0
030A          A520    LDA LSOR        ;ONLY 1 BYTE LEFT IN SOR?
030C          D003    BNE D1A         ;NO, MORE
030E          A9FF    LDA #$FF        ;INDICATE DIVISION BY ZERO
0310          60      RTS             ;RETURN
0311
0311 D1A      B522    LDA SOR+1,X     ;X WAS ZERO
0313          9521    STA SOR,X       ;ALIGN TO LEFT
0315          E8      INX
0316          E420    CPX LSOR        ;ARE WE AT END?
0318          D0F7    BNE D1A         ;NO
031A          C620    DEC LSOR        ;WAS > ZERO
031C          10E7    BPL D1          ;BRANCH ALWAYS
031E
031E DLEN
031E          38      SEC             ;CALCULATE LEN OF QUOT.
031F          A900    LDA #0
0321          8529    STA Q
0323          6518    ADC LDEND
0325          E520    SBC LSOR
0327          8528    STA LQ          ;ULTIMATE LENGTH-1 OF QUOTIENT
0329          AA      TAX             ;BE ADDRESSER
032A          1003    BPL DO          ;LEN OKAY
032C          A9FE    LDA #$FE        ;INDICATE UNDERFLOW OF RESULT
032E          60      RTS             ;RETURN TO CALLER FOR ADJUSTMENT
032F
032F DO       A900    LDA #0          ;CLEAR QUOTIENT
0331          9529    STA Q,X
0333          CA      DEX
0334          10F9    BPL DO
0336          A8      TAY             ;Y=0 TO BE ADDRESSER TO QUOTIENT
0337          8510    STA CARRY       ;INDICATE NO CARRY OVER
0339
0339          A511    LDA MODE        ;GET MODE AT SUBROUTINE CALL
033B          48      PHA             ;TO STACK
033C          28      PLP             ;RESTORED TO STATUS
033D
033D D08
033D          A620    LDX LSOR        ;SUBTRACT THE DIVISOR
033F          38      SEC
0340 D10
0340 D10      B519    LDA DEND,X      ;WORKING ON THE
0342          F521    SBC SOR,X       ;PART OF DIVIDEND
0344          9519    STA DEND,X      ;STORE REST OF THIS BYTE
0346          CA      DEX             ;WE ARE MOVING TO THE FRONT
0347          10F7    BPL D10         ;WORK ON MORE BYTES
0349          900B    BCC BORROW      ;DIVIDEND < DIVISOR
034B ADD1                             ;ADD 1 TO RESULT
034B          B92900  LDA Q,Y         ;C=1
034E          6900    ADC #0
0350          992900  STA Q,Y         ;NEW RESULT
0353          4C3D03  JMP D08         ;NEXT TURN
```

```
0356 BORROW
0356          A510    LDA CARRY       ;BORROW FROM CARRY POSSIBLE?
0358          F007    BEQ NOBORR      ;NO
035A          E900    SBC #0          ;C=0
035C          8510    STA CARRY       ;1 OFF CARRY NOW
035E          4C4B03  JMP ADD1
0361
0361 NOBORR
0361          A620    LDX LSOR        ;CAN'T BORROW FROM CARRY
0363 NOBOR1
0363          B519    LDA DEND,X      ;C WAS 0
0365          7521    ADC SOR,X       ;RESTORE FORMER DIVIDEND
0367          9519    STA DEND,X
0369          CA      DEX
036A          10F7    BPL NOBOR1      ;ALLE BYTES DONE?
036C SHIFT
036C          A518    LDA LREST       ;IS A SHIFT ALLOWED?
036E          C520    CMP LSOR        ;COMPARE THE LENGTH
0370          F018    BEQ OUTT        ;ALL DONE
0372          A200    LDX #0
0374          A519    LDA DEND        ;MOVE FIRST BYTE TO CARRY
0376          8510    STA CARRY
0378          C618    DEC LREST       ;1 MOVEMENT LESS
037A SHI1
037A          B51A    LDA DEND+1,X    ;SHIFT TO LEFT
037C          9519    STA DEND,X
037E          E418    CPX LREST       ;DON'T SHIFT TOO MUCH
0380          F004    BEQ SHI2        ;DONE!
0382          E8      INX
0383          4C7A03  JMP SHI1        ;1 MORE
0386 SHI2
0386          C8      INY             ;ADDRESS RESULT +1
0387          4C3D03  JMP D08         ;TRY A NEW START
038A
038A OUTT                             ;CALCULATION DONE
038A NORM1                            ;REST TO BE NORMALIZED?
038A          A519    LDA REST        ;SEE IF LEADING ZERO
038C          D014    BNE NORM2       ;NOT ZERO
038E          A518    LDA LREST       ;> 0 ?
0390          F010    BEQ NORM2       ;NO SHIFT POSSIBLE
0392          C618    DEC LREST       ;1 LESS TO ADDRESS
0394          A200    LDX #0
0396 NORM1A   B51A    LDA REST+1,X    ;SHIFT 1 POS.
0398          9519    STA REST,X      ;TO LEFT
039A          E418    CPX LREST       ;DONE?
039C          F0EC    BEQ NORM1       ;MORE ZEROES?
039E          E8      INX
039F          4C9603  JMP NORM1A      ;MORE SHIFTS
03A2
03A2 NORM2                            ;QUOTIENT TO BE NORMALIZED?
03A2          A529    LDA Q           ;LEADING ZERO IN QUOTIENT?
03A4          D014    BNE OUTT1       ;NO
03A6          A528    LDA LQ          ;SHIFT ALLOWED?
03A8          F010    BEQ OUTT1       ;NO
03AA          C628    DEC LQ          ;1 LESS TO ADDRESS
03AC          A200    LDX #0
03AE NORM2A   B52A    LDA Q+1,X       ;SHIFT 1 PLACE
03B0          9529    STA Q,X         ;TO LEFT
```

```
03B2            E428    CPX LQ          ;ALL DONE
03B4            F0EC    BEQ NORM2       ;MORE ZEROES?
03B6            E8      INX             ;MORE SHIFTS
03B7            4CAE03  JMP NORM2A
03BA OUTT1
03BA            A900    LDA #0          ;SIGNAL SUCCESFUL DIVISION
03BC            D8      CLD             ;CLEAR DECIMAL MODE
03BD            60      RTS
03BE                    .END                                    R.L. #
```

Wolfgang Radeloff, Pinneberg, und Wolfgang Meyer, Hamburg

Mischbilder vom PET und einer Video - Kamera

Es wird eine Geräteanordnung nebst Betriebsprogramm beschrieben, bei der ein PET 2001 Bannerschrift (auffällige Großschrift) erzeugt, die mit dem Bild einer Video-Kamera auf einen Bildschirm-Monitor gemischt wird (siehe Bild 1). Diese Anordnung wurde für Lehrzwecke speziell in einer Schreibmaschinen-Schule entwickelt, sie läßt sich aber auch für andere Lehrzwecke und im Werbebereich sinnvoll einsetzen und auch auf andere CBM-Rechner übertragen, wenn einige Einsprünge in ROM-Routinen verändert werden.

Nach dem Grundprinzip wird der Bildschirm des Monitors in zwei Zonen aufgeteilt. In der oberen Zone arbeitet eine Video-Kamera, die das Bild der Tastatur auf den Bildschirm bringt. Die Fingerhaltung kann so gezeigt werden. Jede Tastenbetätigung erzeugt im unteren Bildschirmbereich, den der Rechner verwaltet, das entsprechende Zeichen in einer Matrix von 3x4 Characteren. Durch Verwendung der 1/4 Grafik-Zeichen des PET/CBM wird im Endeffekt eine 6x8-Auflösung erreicht.

Bild 1 zeigt im Blockschaltbild die Zusammenschaltung der Geräte, Bild 2 zeigt die Schaltung des Interfaces in demonstrativ einfacher Schaltungsausführung, die die notwendige Synchronisation bewirkt.. Die Buchstaben der Bannerschrift sind in der sog. MAIN TABLE ab Adresse 1D00 in codierter Form in jeweils 6 Bytes abgelegt (Bild 5). Diese Zeichentabelle, die hier aus Platzgründen nicht abgedruckt werden kann, wurde für Sonderzeichen, Ziffern, Klein- und Großbuchstaben angelegt. Sie führt im Zusammenhang mit Maschinenprogrammen, die im 2. Cassettenpuffer abgelegt sind, mit Hilfe der 1/4-Grafik-Symbole (Bild 3) zu einer Buchstabenausgabe z.B. gemäß Bild 4.

Es ist sichergestellt, daß die Zeichenausgabe im direkten Modus unmittelbar von der Tastatur her erfolgen kann aber auch aus einem Textspeicher, der von einem BASIC-Editor-Programm angelegt und verwaltet wird. Dieser Editor folgt in der Bedienung dem Editor des AIM 65. Er hat folgende Befehle:

R READ	Texteingabe von Tastatur oder von der Cassette
T TOP	Ansteuern der obersten Zeile
B BOTTOM	Ansteuern der untersten Zeile
U UP	Aufsetzen auf Vorzeile
D DOWN	Aufsetzen auf Folgezeile
K KILL	Zeile löschen
I INSERT	1 Zeile einfügen
C CHANGE	in der Zeile ändern
F FIND	Textkette im Speicher suchen
L LIST	Textausgabe
N NEW	Löschen des gesamten Textes
*	Laufschrift von Tastatur ausgeben
X EXIT	Programmende
	Daneben werden Clear Sreen und Cursor Down ausgeführt.

Bei Interesse können die Programmteile (Zeichencodierungen, Maschinenprogramm, Editor) beim Autor bezogen werden (W.Radeloff, Elmshorner Str. 13, 2080 Pinneberg). Ein kompletter Abdruck würde den Umfang dieses Hefte sprengen. Gleichwohl mag das Prinzip der Bildmischung einen größeren Leserkreis anregen.

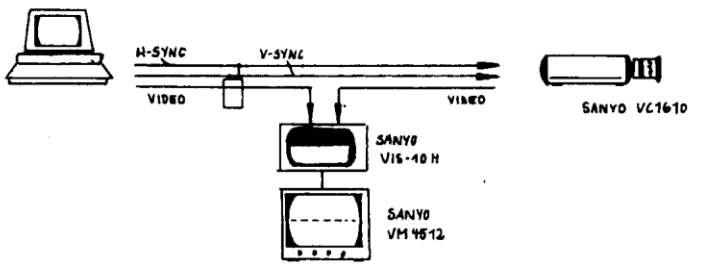


Bild 1: Blockschaltbild des Geräteverbundes

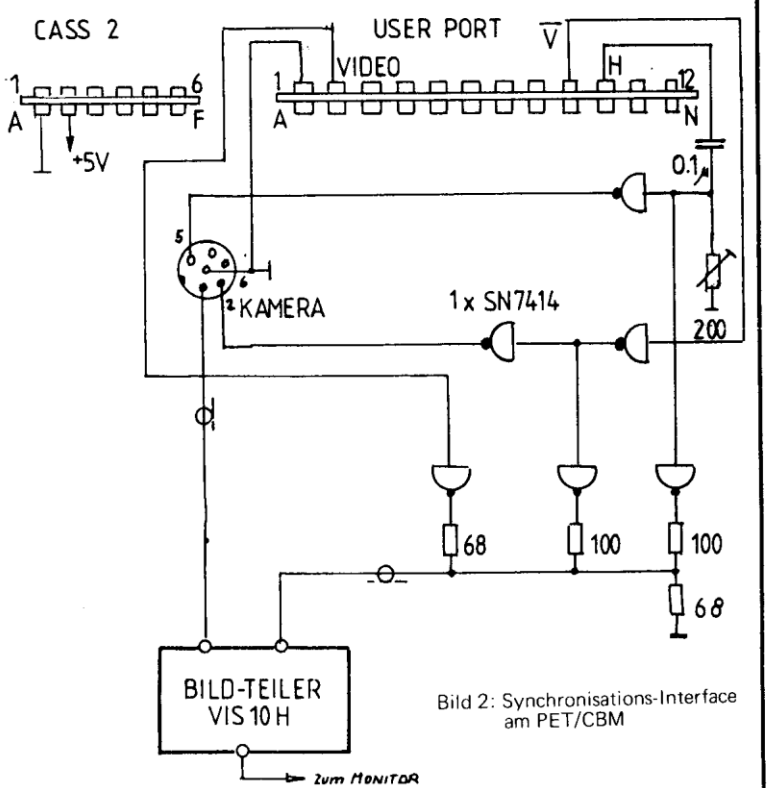


Bild 2: Synchronisations-Interface am PET/CBM

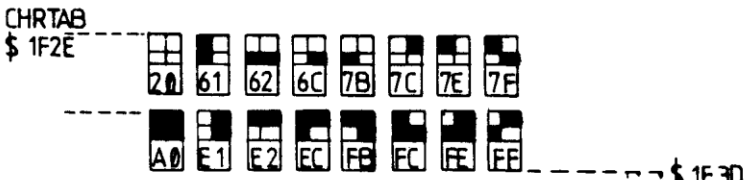


Bild 3: Zeichen der 1/4-Grafik

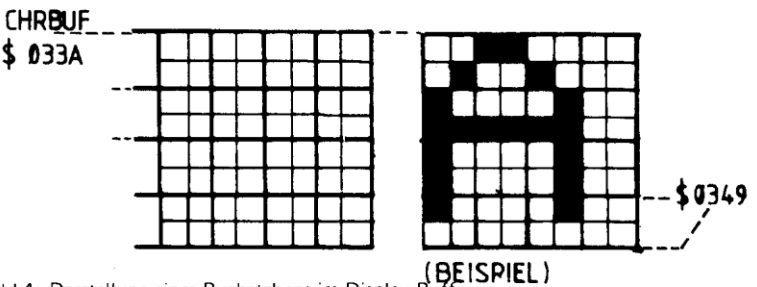


Bild 4: Darstellung eines Buchstabens im Display-Buffer

Ausschnitt aus der MAINTAB

```
1D00   00 00 00 00 00 00 ; SPACE
1D06   00 00 99 50 00 00 ;!
1D0C   00 00 61 00 61 00 ;"
1D12   50 05 B1 6B B1 6B ;#
1D18   50 03 EE 5E 42 0F ;$
```

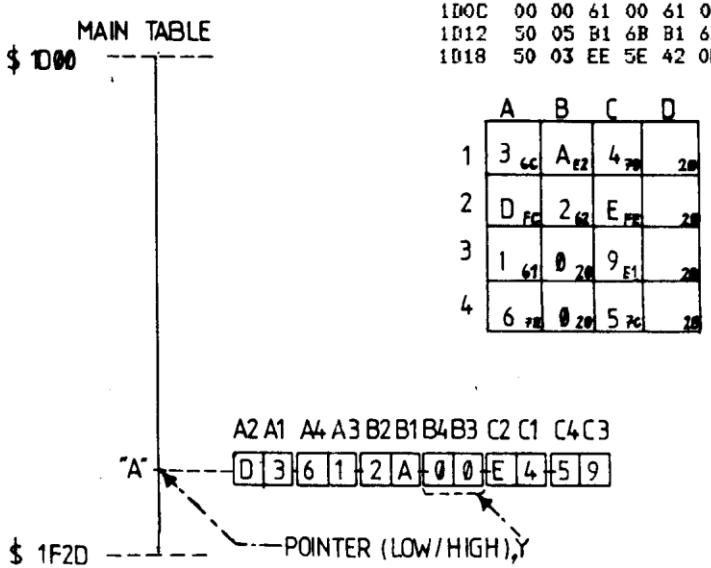


Bild 5: Codierung des Zeichenmusters in der MAINTAB

#

Horst Brettin, 1000 Berlin 44

Berechnung von Pi mit großer Genauigkeit

Anhand eines Programmes, das bei genügend großem Speicher die Berechnung der Zahl Pi auf 10 000 (zehntausend) Stellen ermöglicht, soll gezeigt werden, wie derartige Probleme gelöst werden können (ein CBM 3032 würde dafür etwa 844 std. benötigen = Hochrechnung).

Das Programm dürfte mit geringen Änderungen auf allen BASIC-Rechnern laufen, die die Dimensionierung großer Felder gestatten (beim PET 2001 werden Felder mit mehr als 255 Elementen nicht mehr richtig verarbeitet). - Für die Berechnung wurde eine Formel von John Machin (1706) zugrunde gelegt:

$$\pi = \arctan(1/5) - \arctan(1/239)$$

Die arctan-Funktion wird durch eine Reihenentwicklung berechnet:

$$\arctan(x) = x - x^3/3 + x^5/5 - x^7/7 + \ldots$$

Die Rechnung selbst erfolgt in einem 2-dimensionalen Feld mit N=3 Dezimalstellen pro Element. Dafür langt eine Integer-Variable. n muß folgender Ungleichung genügen:

$$Z_m > 10^n \cdot DI_{max}$$

Darin ist:

Z_m die größte Zahl, für die der Rechner noch Z_m einwandfrei von z Z_m+1 unterscheiden kann (beim CBM $4.29.10^9$)

n die Zahl der Dezimalstellen pro Feldelement (hier 3)

DI_{max} der größte vorkommende Divisor

Wird diese Ungleichung verletzt, so können bei den einzelnen Divisionsschritten Fehler durch Rundung entstehen! Addition und Subtraktion sind in dieser Beziehung unproblematisch. Alles weitere erklärt sich aus dem Programm selbst.

Die Rechenzeit steigt etwa mit dem Quadrat der Stellenzahl. Für 1000 Stellen beträgt sie etwa 8 std 30 min. Auf besondere Maßnahmen zur Verkürzung der Rechenzeit wurde hier der Klarheit wegen verzichtet. Es wäre z.B. möglich, die Berechnung erst beim ersten von Null verschiedenen Element zu beginnen. Die Formel von C. F. Gauß wäre auch günstiger gewesen:

$$\pi = 48\arctan(1/18) + 32\arctan(1/57) - 20\arctan(1/239)$$

Literatur: Kleine Enzyklopädie Mathematik, Meyers Großer Rechenduden I.

Zum Schluß darf der Autor auf eine 'kleine Sünde' aufmerksam zu machen, die häufig in Maschinenprogrammen auftritt: Die Befehlsfolge JSR XYZ und dann RTS ist unnötig kompliziert. Wenn man schon in einem Unterprogramm ist, so sollte man codieren JMP XYZ. Man erspart der CPU die zweimalige Umschaufelei über den Stack im Zusammenhang mit dem unnötigen Unterprogrammaufruf.

```
π  10 000

100 REM (C) BY HORST BRETTIN 1981
110 PRINT"■■■VIEVIEL STELLEN VON π SOLLEN BERECHNET  WERDEN";
120 INPUT SI
130 T=1000:SI=(SI+2)/3:DIM A%(SI,2),C(2)
140 C(1)=25:C(2)=23912:PRINT"■":Z=TI
150 FOR PA=1TO2:GOSUB600
160 GOSUB 700:A=1:DI=E:GOSUB 360
170 IF SN<0THENGOSUB430:GOTO 190
```

```
180 GOSUB510
190 E=E+2:SN=-SN
200 A=0:DI=C(PA):GOSUB360
220 IF ZE THEN 160
230 NEXT:Z=TI-Z
240 PRINT"XXXXDER WERT VON π AUF"3*SI-1"STELLEN":PRINT"π=";
260 PRINT STR$(A%(0,2))".";
270 FOR P=1 TO SI
280 X$=STR$(A%(P,2)):X$=RIGHT$(X$,LEN(X$)-1):PRINT RIGHT$("00"+X$,3);
300 NEXT:PRINT:PRINT"RECHENZEIT"INT(Z/60+.5)"SEC":END
310 REM                                                    " ";
320 REM ==============
330 REM   AC1 / DI
340 REM ==============
350 REM
360 RS=0:ZE=0:FORP=0 TO SI:D=RS*T+A%(P,A):QU=D/DI
370 RS=D-DI*INT(QU):ZE=ZE OR QU:A%(P,A)=QU:NEXT:A%(SI,A)=QU+.5:RETURN
380 REM
390 REM ==============
400 REM   AC2 + AC1
410 REM ==============
420 REM
430 CA=0:FOR P=SI TO 0 STEP-1:SU=A%(P,2)+A%(P,1)+CA
440 CA=0:IF SU>=T THEN SU=SU-T:CA=1
450 A%(P,2)=SU:NEXT:RETURN
460 REM
470 REM ==============
480 REM   AC2 - AC1
490 REM ==============
500 REM
510 LO=0:FOR P=SI TO 0 STEP-1:DC=A%(P,2)-A%(P,1)-LO
520 LO=0:IF DC<0 THEN DC=DC+T:LO=1
530 A%(P,2)=DC:NEXT:RETURN
540 REM
550 REM ============================
560 REM  AC0 U. AC2 (NUR BEI PA=1)
570 REM    LOESCHEN
580 REM ============================
590 REM
600 FOR P=0 TO SI:A%(P,0)=0:IF PA=1 THEN A%(P,2)=0
610 NEXT:A%(0,0)=16/PA^2:DI=234*PA-229:A=0:GOSUB 360:E=1:SN=3-2*PA
640 RETURN
650 REM
660 REM ================
670 REM   AC0 NACH AC1
680 REM ================
690 REM
700 FOR P=0 TO SI:A%(P,1)=A%(P,0):NEXT:RETURN
READY.                                                     #
```

Adressierungsarten des 6809

In den vorausgegangenen Artikeln haben wir den 6809 häufig aus der Sicht des 6502-Programmierers betrachtet. Das hatte für Leser speziell dieser Zeitschrift den Vorteil, daß Vergleiche zu bekanntem gezogen werden konnten. Wir setzen das hier fort, weisen aber schon darauf hin, daß Motorola für die Adressierungsarten z.T. andere Bezeichnungen verwendet, auch wenn sie das Gleiche wie beim 6502 bewirken.

Die Frage nach den Adressierungsarten eines Mikroprozessors ist in Verbindung mit seiner Architektur und seinem Befehlssatz zu sehen. Die Architektur bestimmt a) die Zahl und Größe seiner Register, b) die möglichen internen Transportpfade auf der CPU selbst sowie c) die Fähigkeit, an den Adreßpins berechnete Adressen auszusenden, um Operanden anzusprechen. Wir haben dabei die Frage zu stellen, wie und aus welchen Bestandteilen der Prozessor die effektiv ausgesandten Adressen zu bilden vermag.

Zum besseren Verständnis wollen wir uns noch einmal vor Augen halten, daß der Speicher eines Computers in einen Programm- und einen Datenspeicher zu gliedern ist. DerProgrammzähler rastert Byte für Byte das Programm ab. Er ist dabei als ein Adressensender für den Programmspeicher zu sehen. Die CPU holt sich von dort über den Datenbus Opcodes (hexadezimale Befehlsbytes), ggfs. auch Direktoperanden oder, wenn es zum Befehl gehört, auch Adressen von Operanden.

Das zweite Speichersegment ist der Datenspeicher, der Information enthält, die nicht mit dem eigentlichen Programmablauf zu tun hat, sondern meistensteils die Operanden für die Befehle enthält, auch die Interfacebausteine, die wie normale Speicherzellen adressiert werden. Der Datenspeicher mag aber auch 'Briefkästen' enthalten, in denen nicht der Operand selbst hinterlegt ist, sondern die Speicheradresse, unter der man ihn finden kann. Man nennt das indirekte Adressierung. Sie wird uns noch weiter beschäftigen.

Man kann also die Bildung der Effektivadressen (die Adressierungsarten) wie folgt gliedern:

a) Der Befehl verlangt keine Adreßaussendung an den Programm- oder Datenspeicher, weil der oder die Operanden durch den Befehl ausreichend gekennzeichnet sind. Beim 6502 heißt diese Adressierung 'implied', wie z.B. bei INX, PHA oder 'accumulator' bei ASL A.

Beim 6809 heißt die Adressierung 'inherent' bzw. auch 'accumulator'. 'Inherent' sind:

RTS, RTI	Return from Subroutine/Interrupt
MUL	Multiplikation von Akku A mit Akku B
DAA	Decimal Adjust Akku A
SEX	Vorzeichenexpansion von Akku B nach Akku A
ABX	Addiere Akku B zu Indexregister X
SYNC	Synchronisation auf Interrupt.

Die Adressierungsart 'accumulator' (A oder B) haben die Verschiebebefehle sowie die prüfenden oder verändernden NEG, COM, INC, DEC, TST und CLR.

b) Der Befehl adressiert automatisch einen Direktoperanden im Programmspeicher.

Lade Register (A,B,D,X,Y,S oder U) 'immediate' mit Operand von 1 bzw. 2 Byte (für 2-Byte-breite Register).

Memory-Immediate-Befehle.

PSHS, PSHU, PULS, PULU, bringe/hole einen Registersatz auf den/vom Hardware-/Userstack. Der Registersatz ist im Folgebyte des Befehles als binäre Strichliste enthalten.

Memory-Immediate sind auch ORCC und ANDCC. Bekanntlich hat der 6809 keine Befehle wie 'Set or Clear Status Flag'. Stattdessen bestimmt der Direktoperand des Befehles, welches Flag im Status hinzugeODERt oder hinwegmaskiert wird. - Der Befehl CWAI gehört auch zu dieser Gruppe. Er stackt alle Register auf den Hardwarestack und ANDed den Direktoperanden mit den Prozessorstatus CC.

Inherent-Befehle mit Direktoperand im zweiten Byte des Befehles sind EXG und TFR (Exchange, Transfer Registers). Der Direktoperand ist wieder als binäre Strichliste zu sehen.

c) Die effektiv zu verwendende Adresse ist im Befehl enthalten.

c1) Es handelt sich um eine Direct Page-Adresse. Der Befehl enthält im 2. Byte nur den niederwertigen Teil der Adresse. Während der Befehlsausführung wird automatisch der Inhalt des DP (Direct Page Register) auf den höheren Adreßbus ausgesandt, um den Operanden zu erreichen. Die Adreßbildung geschieht also wie beim 6502 mit den verkürzten Zero Page-Befehlen, allerdings kann beim 6809 jede Speicherseite die Direct Page sein.

c2) Die effektive Adresse ist in den beiden Folgebytes des Befehles enthalten. Motorola nennt diese Adressierung 'extended'. Sie ist wirkungsgleich mit der 'absoluten' Adressierung des6502. Man beachte nur die unterschiedliche Abfolge des hohen (ADH) und niedrigen (ADL) Adreßteiles im Befehl:

6809	Opcode	ADH	ADL
6502	Opcode	ADL	ADH

d) Extended Indirect: Die beiden auf den Opcode folgenden Adreßbytes stellen nicht die effektive Adresse dar, sondern die des 'Briefkastens', in dem die effektive Adresse abgelegt ist.

Diese 'absolut-indirekte' Adressierung gibt es beim 6502 nur für den indirekten Sprung JMP (XXYY). Beim 6809 ist diese Adressierung für die Menge der weiter unten genannten Befehle möglich. Aus Gründen der Prozessorarchitektur wurde sie opcode-mäßig als Teilmenge bei den indizierten Adressierungen untergebracht, obwohl sie nicht indiziert ist.

e) Die effektive Adresse ergibt sich aus dem Inhalt eines der Register X, Y, S oder U mit einem Offset von ±4 Bit, ±7 Bit oder ±15 Bit gegen den Registerinhalt.

Das auf den Befehl folgende Byte, das 'Postbyte' bestimmt, woher dieser Offset zu holen ist, ob auch dem Postbyte selber, aus Akku A, B oder D oder aus den 1-2 Bytes, die auf adas Postbyte im Befehl folgen. Diese Adressierungsart heißt 'indexed', weil Registerinhalte im Spiel sind. Um das Maß voll zu machen gibt es direkte und indirekte Indizierung.

e1) Direkte Indizierung: Die aus Registerinhalt und Offset gebildete Adresse ist die effektive Adresse des Operanden.

e2) Indirekte Adressierung: Die im ersten Durchgang wie vor errechnete Adresse ist die des 'Briefkastens', in dem erst die effektive Adresse hinterlegt ist.

Es besteht damit eine gewisse Ähnlichkeit zum 'pre-indexing' des 6502, das aber nur als (Zero-page,X) dort hinterlegte effektive Adressen zu erreichen gestattet.

Da die verwendbaren Register X, Y, S,U und PC des 6809 16 Bit breit sind, kann mit der indirekten Adressierung jeder 'Briefkasten' im memory erreicht werden.

Der 6809 kennt kein 'post-indexing', wie es in 6502 Befehlen wie LDA (POINTER),Y implementiert ist, wobei Y als ein 'Läufer' über einen Bereich von 256 Byte benutzt werden kann. Aber: Es ist dem Programmierer mit den Befehlen LEAX, LEAY, LEAS und LEAU (Lade eine effektive Adresse in das Indexregister) unbenommen, den Inhalt eines Briefkastens als Adreßbasis in ein Indexregister zu laden und den Index von dort an laufen zu lassen, um ein Datenfeld zu überstreichen. Hierfür gibt es dann noch so schöne Befehlskombinationen mit Auto-Inkrement oder Auto-Dekrement um 1 oder 2 des Indexregisters im Zuge der Befehlsausführung. Das erspart Befehlsfolgen z.B. mit INX, DEX, INY, DEY, Befehle, die es beim 6809 nicht gibt.

Aus vorstehendem wird klar, daß die indizierten Adressierungsarten der besonderen Darlegung bedürfen. Wir kommen alsbald darauf zurück. - Der 6809 hat eine sehr gleichmäßige Architektur die Adressierungsarten

Direct Page
Extended (absolut) und
Indiziert

gelten gleichmäßig für alle folgenden Befehle, ohne daß einer der Befehle nicht mit dem einem oder dem anderen Register ausführbar wäre (beim 6502 unterscheiden sich die mit X und Y indizierbaren Befehlssätze).

Befehle die das Memory verändern (direct in memory):
NEG, COM, die Verschiebebefehle, INC, DEC, TST und CLR.

Befehle, die 8- oder 16-Bit-breite Register und einen Direktoperanden (immediate) oder einen Operanden im memory betreffen:

LOAD, STORE, COMPARE Register		(A, B, D, X, Y, U, S)
Arithmetik:	ADD, SUB	(A, B, D)
Logik:	OR, AND, EOR	(A, B, D)
	BIT	(A, B).

f) Aus Gründen der Systematik ist auch auf die Verzweigungs- und Sprungbefehle sowie auf die Unterprogrammaufrufe hinzuweisen.

Vom 6502 her kennen wir schon die einfachen Branches mit einem ∓7-Bit-Offset für den gegenwärtigen Stand des Programmzählers. Dieser im Befehl stehende Offset ist ein Direktoperand, wenn die Verzweigungsbedingung zutrifft (Adressierungsart 'relative'). Das Gleiche gilt für die Long Branches des 6809 mit einem Offset in ±15 Bit (Adressierungsart 'long relative'). Neu ist, daß es auch Branches to Subroutine gibt (BSR, LBSR), man kann also adreßraum-unabhängig programmieren und jedes Verzweigungsziel erreichen.

Auch die unbedingten Sprünge (JMP) und Unterprogrammaufrufe (JSR) stellen im Programmspeicher Operanden für den Befehlszähler bereit. Beim 6809 sind diese Befehle für folgende Adressierungsarten implementiert:

Direct Page
Extended
Indiziert.

Die Bildung der effektiven Adresse erfolgt analog zu obenstehenden Ausführungen. Insbesondere ist es also möglich, indizierte Sprungleisten aufzubauen, die der BASIC-Formulierung ON .. GOTO oder GOSUB wirkungsmäßig entsprechen.

Wenn man einmal von den indizierten Befehlen, die bis zu 5 Byte lang sein können, absieht, so liegen die Adressierungsarten des 6809 ziemlich einfach. Gleichwohl sollte man bei der Programmierung von Direktoperanden für eine Registerliste oder Verzweigungsziele nicht eines guten Assemblers entraten, denn solche Operanden sind für eine Handcodierung zu fehleranfällig. Das gilt mehr noch für die indizierten Adressierungen.

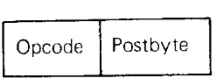
Die Struktur indizierter Befehle

Die Vielfalt der indizierten Befehle legt es nahe, hier zunächst nach einer Ordnung zu suchen. Diese sieht wie folgt aus: Auf den Opcode des Befehles (z.B. A4 für LDA, indiziert) folgt immer ein sog. Postbyte, in dem die Art der Indizierung verschlüsselt ist. Beim Entwurf des 6809 hat man sich bemüht, für die am häufigsten benutzten Befehle die Instruktion mit diesem Postbyte enden zu lassen, so daß sie dann nur insgesamt 2 Byte Speicherplatz beansprucht.

Mit Befehlscode und Postbyte kommen folgende Indizierungsarten aus:

Registeradresse mit Offset 0 (die schnellste Indizierungsart)
Registeradresse mit ±4-Bit Offset, codiert im Postbyte
Registeradresse mit ±Offset in Akku A, B oder D
Registeradresse mitAuto-Inkrement oder -Dekrement um ±1 oder ±2

Befehlsstruktur



Zu vorstehender Befehlsform sind folgende Hinweise zu geben: Sie gilt für direkte Indizierung mit einem der Register X, Y, S oder U. Sie gilt auch für indirekte Indizierung ('Briefkasten'), außer für den Offset von ±4 Bit. Wenn ein solcher Offset benutzt wird, so wird er in einem dritten Byte angelegt, so wie beim untenstehenden ±7-Bit Offset.

Die genannten Indizierungsarten sind bereits bequem und leistungskräftig. Insbesondere ist auf das Auto-Inkrement um +1 oder +2 des indizierenden Registers nach der Befehlsausführung sowie auf das Auto-Dekrement um -1 oder -2 vor der Befehlsausführung hinzuweisen. Das Register läuft dabei ohne weiteres Zutun über ein Datenfeld und erspart Befehle wie INX, DEX usw. des 6502. Diebenutzung dieser Automatik schließt die Verwendung eines Offsets gegen die im Register enthaltene Adresse aus. Da man das indizierende Register sowieso laden muß, ist es nur logisch, wenn man es ggfs. mit einem Offset gegen einen Tabellenanfang oder den Beginn einer anderen Struktur so lädt, daß es unmittelbar auf das zu bearbeitende Feld zeigt. Weiterhin ist es auch logisch, daß man bei indirekter Indizierung, bei der man ja Adreßworte in 'Briefkästen' benutzt, nur Inkremente/Dekremente von 2 zur Verfügung hat und nicht solche von 1 (s. untenstehende Gesamtübersichten).

Wir wiesen bereits darauf hin, daß einige weniger häufig benutzte Befehle vor dem eigentlichen Befehlscode noch ein Prebyte haben, das eine Umschaltung der Befehlsinterpretation bewirkt. Solche Befehle sind z.B. LDY, STY, CMPY, LDS, LDU usw.. Für sie haben die Befehle mit der obigen Indizierung folgendes Aussehen:

Prebyte 10 oder 11	Opcode	Postbyte

Bei der Indizierung sind weitere Formen des Offsets möglich:

±7-Bit Offset gegen das indizierende Register
±15-Bit Offset gegen das indizierende Register.

Im ersten Fall wird ein weiteres Byte an den Befehl angehängt, im zweiten werden 2 Bytes angehängt. In diesen Bytes ist (vorzeichenbehaftet) der Wert des Offset enthalten. Für die weniger häufigen Befehle wird wiederum mit dem Prebyte gearbeitet. Wir haben damit folgende Fomen der Instruktion:

Offset ±7 Bit	Opcode	Postbyte	Offset	
Offset ±15 Bit	Opcode	Postbyte	Offset High	Offset Low

Beim 6809 kann auch der Programmzähler indizierendes Register sein, was für die Erzeugung von vollverschieblichem Maschinencode notwendig ist. Wenn man ein Programm in Segmenten codiert, so wird man auch benötigte Tabellen, Texte usw. direkt im Segment anlegen. Wenn man diese nun innerhalb des Segmentes anspricht, so haben sie eine relative Distanz zum augenblicklichen Stand des Programmzählers. Der Wert der Distanz, ob 8 oder ob 16 Bit ist gleichbleibend, wo immer auch das Segment abgespeichert ist. Es ist Aufgabe des Assemblers, die Distanz zum Label zu berechnen und den Offset in 1-2 Bytes nach dem Postbyte abzulegen. Will man ein Datenfeld adressieren, so wird man also z.B. codieren: LEAX LABEL,PC. Dadurch nimmt X zur runtime die effektive Adresse von LABEL auf.

Die nachfolgenden Übersichten, die den Motorola-Unterlagen entnommen wurden, zeigen die Bedeutung der Bitpositionen im Postbyte, die Palette der Indizierungsmöglichkeiten sowie diejenigen Kombinationen, für die die Verwendung eines Offsets möglich ist.

INDEXED ADDRESSING POSTBYTE REGISTER BIT ASSIGNMENTS

Post-Byte Register Bit								Indexed Addressing Mode
7	6	5	4	3	2	1	0	
0	R	R	X	X	X	X	X	EA = ,R ± 4 Bit Offset
1	R	R	0	0	0	0	0	,R+
1	R	R	1	0	0	0	1	,R++
1	R	R	0	0	0	1	0	,-R
1	R	R	1	0	0	1	1	,--R
1	R	R	1	0	1	0	0	EA = ,R + 0 Offset
1	R	R	1	0	1	0	1	EA = ,R + ACCB Offset
1	R	R	1	0	1	1	0	EA = ,R + ACCA Offset
1	R	R	1	1	0	0	0	EA = ,R + 7 Bit Offset
1	R	R	1	1	0	0	1	EA = ,R + 15 Bit Offset
1	R	R	1	1	0	1	1	EA = ,R + D Offset
1	X	X	1	1	1	0	0	EA = ,PC + 7 Bit Offset
1	X	X	1	1	1	0	1	EA = ,PC + 15 Bit Offset
1	R	R	1	1	1	1	1	EA = ,Address

Addressing Mode Field
Indirect Field
Sign bit when 87 = 0
Register Field
00 R = X
01 R = Y
10 R = U
11 R = S
X = Don't Care

INDEXED ADDRESSING MODES

Type	Forms	Non Indirect		+ ~	+ #	Indirect		+ ~	+ #
		Assembler Form	Postbyte OP Code			Assembler Form	Postbyte OP Code		
Constant Offset From R (Signed Offsets)	No Offset	,R	1RR00100	0	0	[,R]	1RR10100	3	0
	5 Bit Offset	n, R	0RRnnnnn	1	0	defaults to 8-bit			
	8 Bit Offset	n, R	1RR01000	1	1	[n, R]	1RR11000	4	1
	16 Bit Offset	n, R	1RR01001	4	2	[n, R]	1RR11001	7	2
Accumulator Offset From R (Signed Offsets)	A — Register Offset	A, R	1RR00110	1	0	[A, R]	1RR10110	4	0
	B — Register Offset	B, R	1RR00101	1	0	[B, R]	1RR10101	4	0
	D — Register Offset	D, R	1RR01011	4	0	[D, R]	1RR11011	7	0
Auto Increment/Decrement R	Increment By 1	,R+	1RR00000	2	0	not allowed			
	Increment By 2	,R++	1RR00001	3	0	[,R++]	1RR10001	6	0
	Decrement By 1	,-R	1RR00010	2	0	not allowed			
	Decrement By 2	,--R	1RR00011	3	0	[,--R]	1RR10011	6	0
Constant Offset From PC	8 Bit Offset	n, PCR	1XX01100	1	1	[n, PCR]	1XX11100	4	1
	16 Bit Offset	n, PCR	1XX01101	5	2	[n, PCR]	1XX11101	8	2
Extended Indirect	16 Bit Address	—	—	—	—	[n]	10011111	5	2

R = X, Y, U or S X = 00 Y = 01
X = Don't Care U = 10 S = 11

+ and ~ Indicate the number of additional cycles and bytes for the particular variation
#

VALID INDEX MODE COMBINATIONS

OPERAND REGISTER / OFFSET	X	Y	S	U	PCR	--X -X X+ X++	--Y -Y Y+ Y++	--S -S S+ S++	--U -U U+ U++
0									
CONSTANT + OR - UP TO 16 BITS					SEE NOTE 3				
A									
B									
D									

1. OPERATION REGISTER MAY BE A, B, D, X, Y, S, OR U.
2. SHADED AREAS ARE INVALID COMBINATIONS.
3. WHEN PCR IS THE OPERAND REGISTER, THE NUMBER IN THE OFFSET POSITION OF THE ASSEMBLY LANGUAGE STATEMENT IS THE MEMORY LOCATION ADDRESS OR LABEL. THE ASSEMBLER CALCULATES THE OFFSET.

6809 Assembler-Anweisungen

Vor den nachstehenden Programmierbeispielen noch schnell ein Blick auf die wichtigsten Assembleranweisungen, soweit sie sich von denen des 6502 unterscheiden:

* Ein Stern am Zeilenanfang kennzeichnet eine reine Kommentarzeile
ORG Setzen des Zuordnungzählers (entsprechend der *= Anweisung des 6502)
EQU Equate-Anweisung, Wertzuweisung (wie =)
RMB Reserve Memory Byte(s), Speicherplatzreservierung ohne Wertzuweisung
FCB Form Constant Bytes (entsprechend der Anweisung .BYT)
FDB Form Double Byte, Ablage von 2 Bytes, Adreßwörtern
FCC Form Constant Characters, Ablage von ASCII-Zeichen/Strings
Label müssen in der vordersten Schreibstelle beginnen
Befehle und Anweisungen werden mindestens um 1 Stelle eingerückt.

Als anschauliche und einfache Beispiele für die 6809-Programmierung folgen Programme für das Füllen eines Speicherbereiches FILL und für den Transport von Daten von einem Bereich zu einem anderen MOVE. Für beide Aufgaben gibt es weitere Lösungsmöglichkeiten.

```
0001                    ;FUELL-PROGRAMM
0002                    ;SYMBOLERKLAERUNG
0003 4000               START EQU $4000
0004 5000               BIS EQU $5000 ;ENDE+1
0005 4800               VON EQU $4800
0006 ABCD               FILL EQU $ABCD
0007
0008 4000                ORG START      ;PROGRRAMMSTART
0009 4000 8E 4800        LDX #VON       ;LADE INDEXREGISTER
0010 4003 CC ABCD        LDD #FILL      ;FUELLMUSTER
```

```
0011 4006 ED 81          LOOP STD ,X++ ;SPEICHERE MIT AUTO-INKREMENT
0012 4008 8C 5000         CMPX #BIS     ;ENDE ;ERREICHT?
0013 400B 26 F9           BNE LOOP      ;NEIN
0014 400D 3F              SWI
```

```
0001                     ;FUELL-PROGRAMM
0002                     ;SYMBOLERKLAERUNG
0003 4000                START EQU $4000
0004 5000                BIS EQU $5000
0005 4800                VON EQU $4800
0006 0055                FILL EQU $55
0007
0008 4000                 ORG START     ;PROGRAMMSTART
0009 4000 CE 5000         LDU #BIS      ;LADE STACKPOINTER U
0010 4003 86 55           LDA #FILL     ;FUELLMUSTER
0011 4005                LOOP
0012 4005 36 02           PSHU A        ;DEPOSIT ON USER STACK
0013 4007 11 83 4800      CMPU #VON     ;ENDE ERREICHT?
0014 400B 26 F8           BNE LOOP      ;NEIN
0015 400D 3F              SWI           ;RUECKGABE AN MONITOR
```

```
0001                     * MOVE-PROGRAMM
0002                     * ES WERDEN 2 INDEXREGISTER BENUTZT
0003                     ; SYMBOLERKLAERUNG
0004 4000                VON EQU $4000
0005 4200                BIS EQU $4200       ;ENDADRESSE+1
0006 5000                NACH EQU $5000      ;ZIEL DES MOVE
0007
0008 3C00                 ORG $3C00
0009 3C00 34 52           PSHS A,X,U         ;REGISTER RETTEN
0010 3C02 8E 4000         LDX #VON           ;HERKUNFTSADRESSE
0011 3C05 CE 5000         LDU #NACH          ;ZIELADRESSE LADEN
0012 3C08 A6 80          LOOP LDA ,X+        ;HOLE BYTE, INDIZIERT, X=X+1
0013 3C0A A7 C0           STA ,U+            ;INDIZIERT SPEICHERN, U=U+1
0014 3C0C 8C 4200         CMPX #BIS          ;ENDE+1 ERREICHT?
0015 3C0F 26 F7           BNE LOOP           ;NEIN
0016 3C11 35 52           PULS A,X,U         ;REGISTER WIEDERHERSTELLEN
0017 3C13 3F              SWI
```

```
0001                     * MOVE-PROGRAMM
0002                     * ES WIRD NUR 1 INDEXREGISTER BENUTZT
0003                     ; SYMBOLERKLAERUNG
0004 4000                VON EQU $4000
0005 4200                BIS EQU $4200       ;ENDADRESSE+1
0006 5000                NACH EQU $5000      ;ZIEL DES MOVE
0007
0008 3C00                 ORG $3C00
0009 3C00 8E 4000         LDX #VON           ;HERKUNFTSADRESSE
0010 3C03 A6 80          LOOP LDA ,X+        ;HOLE BYTE, INDIZIERT, X=X+1
0011 3C05 A7 89 0FFF      STA NACH-VON-1,X   ;INDIZIERUNG MIT OFFSET
0012 3C09 8C 4200         CMPX #BIS          ;ENDE+1 ERREICHT?
0013 3C0C 26 F5           BNE LOOP           ;NEIN
0014 3C0E 3F              SWI
```

R.L. #

Peter Engels, 5308 Rheinbach-Niederdrees

CBM - Math

Für CBM–Rechner mit BASIC 3 oder mit BASIC 4 werden nachfolgend 4 Mathematik-Funktionen als Maschinenprogramm abgedruckt, die im normalen BASIC nicht enthalten sind. Es handelt sich dabei um:

1. Zehnerlogarithmus	Aufruf: SYS 826(X)
2. X-te Wurzel von Y	SYS 902(X,Y)
3. Umrechnung Altgrad in Bogenmaß	SYS 941(X)
4. Umrechnung Bogenmaß in Altgrad	SYS 957(X)

X oder Y können beliebige Ausdrücke oder Variablen sein. Die Syntax wurde so gewählt, daß die Argumente in Klammern stehen müssen. Werden die Funktionen wie im Beispiel aufgerufen, dann wird das Ergebnis direkt ausgedruckt. Wird allerdings dem Argument eine Klammer mit einer Variablen nachgestellt, so wird das Ergebnis an die Variable übergeben, z.B. SYS826(100,X).

Das Ergebnis (2) wird zur späteren Benutzung an die Variable X übergeben. Die Programme machen häufigen Gebrauch von Interpreterroutinen. Alle vier Routinen finden Platz im zweiten Cassetten-Buffer.

Eine Besonderheit bieten die Routinen DEG und RAD: Um zu hoher Geschwindigkeit zu gelangen, wurde die Konstante π /180 als 5-Byte-Zahl im Speicherformat ab hex 03DC abgelegt. Sie wird bei Gebrauch durch entsprechende Routinen in die Floating Point Accus geladen und dort verarbeitet. Auf diese Weise muß die Zahl nicht jedesmal neu berechnet werden.

Die Routinen arbeiten nach folgenden Formeln:

DEZLOG(X)	=LOG(X)/LOG(10)
X-te Wurzel von Y	=Y↑(1/X)
DEG(X)	=X*(π/180)
RAD(X)	=X/(π/180)

```
LINE# LOC   CODE          SOURCE CODE

001                       *=$033A
002                       ;
003                       ;MATH. ROUT. FUER CBM 4000
004                       ;            UND  CBM 8000
005                       ;
006                       ;DEZ-LOG, X-TE WURZEL,
007                       ;
008                       ;DEG, RAD
009                       ;
010                       KLAUF  =$BEF2
011                       KLZU   =$BEEF
012                       KOMMA  =$BEF5
013                       GET    =$0076
014                       ERROR  =$BF00
015                       ARGIN  =$BD84
016                       STFLP1 =$CD0D
017                       FLPOUT =$CF8D
018                       FLP1-2 =$CD45
019                       LDFLP1 =$CCD8
020                       MEMDIV =$CC45
021                       MEMMUL =$CB5E
022                       POWER  =$D10F
023                       VARLET =$B94D
```

```
024                                    TEST$  =$BD87
025                                    SUCHE  =$C12B
026                                    LOG    =$CB20
027                                    ;
028   033A  20 F2 BE    DLOG      JSR KLAUF
029   033D  20 84 BD              JSR ARGIN    ;ARGUM. HOLEN
030   0340  20 20 CB              JSR LOG
031   0343  A0 01                 LDY #$01
032   0345  A2 00                 LDX #$00     ;FLP1 AB $0100
033   0347  20 0D CD              JSR STFLP1   ;SPEICHERN
034   034A  A9 2F                 LDA #$2F
035   034C  A0 CC                 LDY #$CC     ;REALZAHL 10
036   034E  20 D8 CC              JSR LDFLP1   ;IN FLP1
037   0351  20 20 CB              JSR LOG      ;LOG(10)
038   0354  A9 00                 LDA #$00
039   0356  A0 01                 LDY #$01
040   0358  20 45 CC              JSR MEMDIV   ;LOG(X)/LOG(10
041   035B  20 76 00    VAR       JSR GET      ;GET CHAR
042   035E  C9 29                 CMP #$29     ;WENN NICHT )
043   0360  D0 06                 BNE SEARCH   ;VAR SUCHEN
044   0362  20 EF BE              JSR KLZU
045   0365  4C 8D CF              JMP FLPOUT   ;FLP1 DRUCKEN
046   0368  20 F5 BE    SEARCH    JSR KOMMA
047   036B  20 2B C1              JSR SUCHE    ;VAR SUCHEN
048   036E  85 46                 STA $46
049   0370  84 47                 STY $47
050   0372  20 87 BD              JSR TEST$    ;TEST OB $-VAR
051   0375  A5 08                 LDA $08
052   0377  20 4D B9              JSR VARLET   ;WERTZUWEISUNG
053   037A  20 EF BE              JSR KLZU
054   037D  20 76 00              JSR GET      ;GET CHAR
055   0380  F0 03                 BEQ RETURN   ;RET.WENN ENDE
056   0382  4C 00 BF              JMP ERROR    ;SYNTAX-ERROR
057   0385  60          RETURN    RTS          ;RETURN BASIC
058   0386  20 F2 BE    WURZEL    JSR KLAUF
059   0389  20 84 BD              JSR ARGIN    ;EXPONENT
060   038C  A9 F2                 LDA #$F2     ;REALZAHL 1
061   038E  A0 CA                 LDY #$CA     ;IN FLP2
062   0390  20 45 CC              JSR MEMDIV   ;KEHRWERT BER.
063   0393  A2 00                 LDX #$00
064   0395  A0 01                 LDY #$01     ;FLP1 AB $0100
065   0397  20 0D CD              JSR STFLP1   ;SPEICHERN
066   039A  20 F5 BE              JSR KOMMA
067   039D  20 84 BD              JSR ARGIN    ;BASIS HOLEN
068   03A0  20 45 CD              JSR FLP1-2   ;IN FLP2
069   03A3  A9 00                 LDA #$00     ;KEHRWERT
070   03A5  A0 01                 LDY #$01     ;ZURUECKHOLEN
071   03A7  20 0F D1              JSR POWER    ;X^(1/Y)
072   03AA  4C 5B 03              JMP VAR      ;VAR O. DRUCK
073   03AD  20 F2 BE    DEG       JSR KLAUF
074   03B0  20 84 BD              JSR ARGIN    ;WINKEL HOLEN
075   03B3  A0 03                 LDY #$03     ;HI+LO BYTE
076   03B5  A9 DC                 LDA #$DC     ;KONSTANTE
077   03B7  20 5E CB              JSR MEMMUL   ;X*π/180
078   03BA  4C 5B 03              JMP VAR      ;VAR O. DRUCK
079   03BD  20 F2 BE    RAD       JSR KLAUF
080   03C0  20 84 BD              JSR ARGIN    ;WINKEL HOLEN
081   03C3  A2 00                 LDX #$00     ;
082   03C5  A0 01                 LDY #$01     ;FLP1 AB $0100
```

```
083   03C7  20 0D CD          JSR STFLP1   ;SPEICHERN
084   03CA  A0 03             LDY #$03     ;HI+LO BYTE
085   03CC  A9 DC             LDA #$DC     ;KONSTANTE
086   03CE  20 D8 CC          JSR LDFLP1   ;PI/180 IN FLP1
087   03D1  A9 00             LDA #$00
088   03D3  A0 01             LDY #$01
089   03D5  20 45 CC          JSR MEMDIV   ;X/PI/180
090   03D8  4C 5B 03          JMP VAR      ;VAR 0. DRUCK
091   03DB  00                BRK
092                           ;
093                           ;KONSTANTE 'PI/180'
094                           ;  = 0.174532925
095                           ;IN 5-BYTE SPEICHERFORMAT
096                           ;
097   03DC              KONST  .BYT $7B $0E $FA $35 $11
098                           ;
099                           .END

001                           *=$033A
002                           ;
003                           ;MATH. ROUT. FUER CBM 3000
004                           ;
005                           ;DEZ-LOG, X-TE WURZEL,
006                           ;
007                           ;DEG, RAD
008                           ;
009                           ;
010                           KLAUF   =$CDF5
011                           KLZU    =$CDF2
012                           KOMMA   =$CDF8
013                           GET     =$0076
014                           ERROR   =$CE03
015                           ARGIN   =$CC8B
016                           STFLP1  =$DAE3
017                           FLPOUT  =$DCE3
018                           FLP1-2  =$DB1B
019                           LDFLP1  =$DAAE
020                           MEMDIV  =$DA1B
021                           MEMMUL  =$D934
022                           POWER   =$DE65
023                           VARLET  =$C8CA
024                           TEST$   =$CC8E
025                           SUCHE   =$CF6D
026                           LOG     =$D8F6
027                           ;
028   033A  20 F5 CD     DLOG   JSR KLAUF
029   033D  20 8B CC            JSR ARGIN    ;ARGUM. HOLEN
030   0340  20 F6 D8            JSR LOG
031   0343  A0 01               LDY #$01
032   0345  A2 00               LDX #$00     ;FLP1 AB $0100
033   0347  20 E3 DA            JSR STFLP1   ;SPEICHERN
034   034A  A9 05               LDA #$05
035   034C  A0 DA               LDY #$DA     ;REALZAHL 10
036   034E  20 AE DA            JSR LDFLP1   ;IN FLP1
037   0351  20 F6 D8            JSR LOG      ;LOG(10)
038   0354  A9 00               LDA #$00
039   0356  A0 01               LDY #$01
040   0358  20 1B DA            JSR MEMDIV   ;LOG(X)/LOG(10
041   035B  20 76 00     VAR    JSR GET      ;GET CHAR
```

```
042   035E  C9 29          CMP #$29     ;WENN NICHT )
043   0360  D0 06          BNE SEARCH   ;VAR. SUCHEN
044   0362  20 F2 CD       JSR KLZU
045   0365  4C E3 DC       JMP FLPOUT   ;FLP1 DRUCKEN
046   0368  20 F8 CD  SEARCH JSR KOMMA
047   036B  20 6D CF       JSR SUCHE    ;VAR SUCHEN
048   036E  85 46          STA $46
049   0370  84 47          STY $47
050   0372  20 8E CC       JSR TEST$    ;TEST OB $-VAR
051   0375  A5 08          LDA $08
052   0377  20 CA C8       JSR VARLET   ;WERTZUWEISUNG
053   037A  20 F2 CD       JSR KLZU
054   037D  20 76 00       JSR GET      ;GET CHAR
055   0380  F0 03          BEQ RETURN   ;RET.WENN ENDE
056   0382  4C 03 CE       JMP ERROR    ;SYNTAX-ERROR
057   0385  60        RETURN RTS         ;RETURN BASIC
058   0386  20 F5 CD  WURZEL JSR KLAUF
059   0389  20 8B CC       JSR ARGIN    ;EXPONENT
060   038C  A9 C8          LDA #$C8     ;REALZAHL 1
061   038E  A0 D8          LDY #$D8     ;IN FLP2
062   0390  20 1B DA       JSR MEMDIV   ;KEHRWERT BER.
063   0393  A2 00          LDX #$00
064   0395  A0 01          LDY #$01     ;FLP1 AB $0100
065   0397  20 E3 DA       JSR STFLP1   ;SPEICHERN
066   039A  20 F8 CD       JSR KOMMA
067   039D  20 8B CC       JSR ARGIN    ;BASIS HOLEN
068   03A0  20 1B DB       JSR FLP1-2   ;IN FLP2
069   03A3  A9 00          LDA #$00     ;KEHRWERT
070   03A5  A0 01          LDY #$01     ;ZURUECKHOLEN
071   03A7  20 65 DE       JSR POWER    ;X^(1/Y)
072   03AA  4C 5B 03       JMP VAR      ;VAR O. DRUCK
073   03AD  20 F5 CD  DEG  JSR KLAUF
074   03B0  20 8B CC       JSR ARGIN    ;WINKEL HOLEN
075   03B3  A0 03          LDY #$03     ;HI+LO BYTE
076   03B5  A9 DC          LDA #$DC     ;KONSTANTE
077   03B7  20 34 D9       JSR MEMMUL   ;X*π/180
078   03BA  4C 5B 03       JMP VAR      ;VAR O. DRUCK
079   03BD  20 F5 CD  RAD  JSR KLAUF
080   03C0  20 8B CC       JSR ARGIN    ;WINKEL HOLEN
081   03C3  A2 00          LDX #$00     ;
082   03C5  A0 01          LDY #$01     ;FLP1 AB $0100
083   03C7  20 E3 DA       JSR STFLP1   ;SPEICHERN
084   03CA  A0 03          LDY #$03     ;HI+LO BYTE
085   03CC  A9 DC          LDA #$DC     ;KONSTANTE
086   03CE  20 AE DA       JSR LDFLP1   ;π/180 IN FLP1
087   03D1  A9 00          LDA #$00
088   03D3  A0 01          LDY #$01
089   03D5  20 1B DA       JSR MEMDIV   ;X/π/180
090   03D8  4C 5B 03       JMP VAR      ;VAR O. DRUCK
091   03DB  00             BRK
092                        ;
093                        ;KONSTANTE 'π/180'
094                        ; = 0.174532925
095                        ;IN 5-BYTE SPEICHERFORMAT
096                        ;
097   03DC           KONST .BYT $7B $0E $FA $35 $11
098                        ;
099                        .END                          #
```

AIM Spezial (11)

PASCAL für den AIM 65/PC 100/Syko-Logic 100 & 300. Etwa ab Januar 1982 wird für diese Rechner neben Assembler, BASIC, FORTH und PL 65 nun auch PASCAL lieferbar sein. Herr Hans-Joachim Regge aus Bremen sandte einen ersten Bericht: PASCAL umfaßt 20 kByte in der Form von 5 ROMs für den Speicherbereich hex 4000-7FFF und B000-BFFF. Zu den ROMs gehört ein kleines Handbuch im Umfang des Monitor-Listings. Bis auf geringfügige Abweichungen, die im Handbuch erläutert sind, ist es kompatibel mit Standard-Pascal von Nicolaus Wirth. Durch das Zusammenspiel mit dem AIM–Monitor ist die Benutzung des Texteditors (nur 'C' hat hier eine andere Funktion, nämlich check'), ggfs. eines Terminals oder einer Video-Karte und eines Cassettenrekorders zur Programmaufzeichnung möglich. Die vom Programm benutzten Zero-Page-Adressen (wichtig bei evtl. erforderlicher DOS-Anpassung) sowie alle belegten Adreßbereiche sind im Anhang des Handbuches dokumentiert. Als RAM sind nur die 4 kByte auf der AIM-Platine erforderlich, für längere Programme kann dann ggfs. der Bereich bis hex 3FFF mit RAM bestückt werden. Nach ersten Erfahrungen arbeitet das DAIM-DOS 2.2 beim Keyboard-Betrieb des AIM problemlos mit PASCAL zusammen. Wird wie beim Verfasser ein Terminal benutzt, so ist das nebenstehende Mini-Programm zur Ausblendung von LF erforderlich. Es ist voll verschieblich und leicht irgendwo unterzubringen.

```
0000          ;PROGRAMM ZUR AUSBLENDUNG VON LINEFEED
0000          ;BEIM GEBRAUCH DER DAIM-FLOPPY
0000          ;ZUSAMMEN MIT EINEM TERMINAL UND PASCAL
0000
0000                *=$C000
C000  B003    BCS $C005
C002  4CB79E  JMP $9EB7
C005  68      PLA             ;IS DATA, RECOVER CHARCTER
C006  C90A    CMP #$0A        ;LINE FEED?
C008  D001    BNE $C00B
C00A  60      RTS             ;YES, NO OUTPUT TO DISK
C00B  48      PHA             ;SAVE CHARCTER ON STACK
C00C  4CAC9E  JMP $9EAC       ;JUMP TO DISK ROUTINE
C00F          JMP $9EAC       ;JUMP TO DISK ROUTINE
```

Der AIM 65/40 wurde in Heft 16 angekündigt. Seit September 1981 wird er in Stückzahlen ausgeliefert. Rockwell stellte diesen Rechner natürlich auf der SYSTEMS 81 in München aus und berichtete über eine große Zahl vorliegender Bestellungen. Es ist geplant, den AIM 65/40 in Heft 23 ausführlich zu besprechen. Man kann den Rechner in verschiedenen Ausstattungsoptionen beziehen. Eine Preisliste liegt hier nicht vor, es wird aber berichtet, daß die Standardausführung mit 16 k RAM, Tastatur, Drucker und Fluoreszenzdisplay in Einzelstückzahlen DM 3407,- DM + 13% kosten soll. - Die Firma GWK-Elektronik in Herzogenrath arbeitet dem Vernehmen nach bereits daran, ihre diversen Ergänzungskarten auch auf den Anschluß an diesen Rechner einzurichten. Weiterhin soll bis Januar/Februar 1982 auch ein Extended BASIC für den 65/40 von dort aus lieferbar sein.

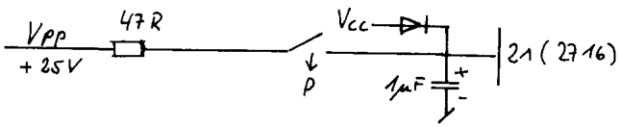
Zusätzliche Ein- und Ausgänge am AIM befinden sich an der System-PIA 6520, die das LED-Display bedient. Herr Kokula aus Ludwigshafen machte auf diesen verborgenen Schatz aufmerksam. Gemeint sind die Kontrolleitungen CA1, CA2, CB1 und CB2, die allerdings noch nicht herausgeführt aber leicht zu erreichen sind.

Bei Insert- und Killoperationen im Editor traten bei ihm, wie auch wohl bei verschiedenen anderen Lesern immer einmal Fehler ein, die zu einer Veränderung von Zeichen führten (auch hier beobachtet). Er beseitigte den Fehler, indem er das für die Hauptplatine selbst nicht verwertete Signal RAM-R/$\overline{W}$ (es führt zum Stecker J3-Z) auch für die RAMs on board benutzt. Das Besondere an diesem Signal ist, daß $\overline{WRITE}$ nicht während des ganzen Zyklus, sondern nur während der Phase $\phi 2$

aktiv ist. Bei frühen AIMs ist allerdings im board ein Schaltfehler: An Z13-4 liegt Z-16-6, das Signal SYS R/$\overline{W}$, es muß aber Z16-4, das negierte Signal anliegen. Wir machten bereits in Heft 5, Seite 17 darauf aufmerksam..

#

Hinweise für das Brennen von EPROMs: Immer wieder sterben EPROMs beim Zuschalten von V_{pp} ohne daß dafür ein erkennbarer Grund vorliegt. Bisher half ich mir durch einen Vorwiderstand von 47 Ohm in der V_{pp}-Zuleitung und einem 1 µF-Elko am Pin 21 (2716) gegen Ground:



Im neuesten INTEL-EPROM-Handbuch fand ich einen Hinweis, der meine Maßnahme rechtfertigt: Dort wird empfohlen, am V_{pp}-Pin einen Kondensator von mindestens 0,1 µF gegen Ground vorzusehen gegen 'transiente Störungen', die das EPROM zerstören können. Solche Störungen können durch das Prellen des Schalters eintreten, mit dem die Brennspannung zugeschaltet wird!

INTEL liefert seit kurzem einen EPROM 2732A. Er wird in einer neuen Technologie gefertigt (HMOS) und ist dadurch schneller und leistungsärmer. ABER: Die Programmierspannung darf maximal 21,5 V betragen, höhere Spannung zerstört ihn (wer achtet schon auf das 'A'). Im READ-Betrieb ist er gleich dem 2732.

Bernhard Kokula

Bücher

Warren, Carl D., The MC 6809 Cookbook, Tab Book Inc., Blue Ridge Summit, PA. 17214, 1981, ISBN 0-8306-1209 -2, 176 S., ca. DM 29,80. Das Buch stellt vornehmlich den Befehlssatz und die Adressierungsarten des 6809 vor. Es wird eine 'Very Tiny Language' beschrieben.

Leventhal, Lance A., 6809 Assembly Language Programming, Osborne/McGraw Hill, Berkeley, CA. 1981, ISBN 0-931 988-35-7, ca. 500 S., ca. DM 54,30. - Der Autor hatbereits mehrere Lehrbücher für Microprozessoren verfaßt. Ihm ist wieder nach bewährtem Aufbauschema eine gründliche Darstellung des 6809 und seiner Programmierung gelungen. Empfehlenswert.

Schumny, H. (Herausgeber), Taschenrechner + Microcomputer Jahrbuch 1982, Vieweg Verlag, Braunschweig/Wiesbaden 1981, 274 S., ISBN 3-528-04196-X, DM 29,80. - Es ist die dritte Ausgabe des beliebten Jahrbuches. Die Gliederung umfaßt Taschenrechner/Taschencomputer, Rechner in der Schule, Programmierung, Microprozessor-Praxis, Programmsammlung, Produktübersichten, Adressen, Bücher- und Zeitschriftenverzeichnis. Der Leser findet in gewohnter Sorgfalt bearbeitet eine Fülle von Produktbesprechungen, Programmen, Lieferantenverzeichnisse für Bausteine und Systeme und wichtige Branchenadressen. Das Jahrbuch schafft einmal wieder Übersicht, zeigt Trends und hilft entscheiden.

Pauly, M. et alia, Hardware-Auswahl leicht gemacht, Verlag Markt & Technik, Haar 1981, ca. 200 S., ISBN 3-922 120-14-8, DM 29,--. - Der Untertitel lautet: Peronal Computer und ihre Peripherie. Entscheidungshilfen und kennzeichnende Daten für alle, die vor dem Kauf eines Rechnersystems stehen. - Themengliederung: Personal Computer, Eingabeeinheiten, Ausgabeeinheiten, Massenspeicher, Schnittstellen, Möbel und Zubehör, Bezugsquellenverzeichnis, Produktionformation einzelner Hersteller. Die Abschnitte gleidern sich jeweils übersichtlich in einen erklärenden Aufsatz, Auswahlkriterien, eine Marktübersicht und ein Stichwortverzeichnis. Dem Leser wird damit die Orientierung sehr erleichtert.

Kleitner, A. und Pauly, M., Software-Auswahl leicht gemacht, Verlag Markt & Technik, Haar 1981, ca. 220 S., ISBN 3-922 120-13-X, DM 29,--. Das Buch enthält mehr als 800 Programmbeschreibungen aus allen Anwendungsbereichen für Personal Computer. Die Gliederung umfaßt Rechnungswesen- und Verwaltung, branchenneutral, Branchenpakete, Technik und Wissenschaft, Systemsoftware. Die einzelnen Einträge umfassen den Namen des Programmes und seine Leistungen, die benötigte Hardware, das Betriebssystem, den Preis, Autor und Bezugsquelle. Da die Entwicklung von Software bekanntlich sehr zeit- und kostenaufwendig ist, sollte der Leser vor eigenen Anstrengungen oder einem Kauf in diesem Buch prüfen, ob er nicht im vorhandenen Angebot das Passende findet.

Johann Leitner, 6000 Frankfurt

Breite Monitorausgabe für CBM 8032

Der Bildschirm des 8032 wird bei der Darstellung der Speicherinhalte über den maschinensprachlichen Monitor nicht voll ausgenutzt, genausowenig die meisten Papierformate. Durch die Änderung zweier Bytes im Monitorprogramm kann die Anzahl der Adressen pro Zeile auf 16 (oder eine beliebige andere Zahl) vergrößert werden. Geändert wird in den Adressen hex D5EF und D626 von hex 08 auf hex 10 (siehe Listing).

Das D-ROM (Position UD8 auf der Hauptplatine) wird dann durch ein entsprechend programmiertes 4k-EPROM ausgetauscht. - Man könnte noch einen Schritt weitergehen und am Zeilenende nach den hexadezimalen Werten die zugehörigen ASCII-Zeichen ausgeben. Im D-ROM ist noch eine Menge Platz (DEC2 - DFFF) für eine entsprechende Routine.

Literaturhinweis: cbm 3016 und cbm 3032 Assembler Listing (ohne BASIC), Fa. Commodore, Neu-Isenburg, Juni 1979.

```
c*
     pc  irq  sr ac xr yr sp
.;  b780 e455 3a 9e 35 34 fa
.
.:  d5ee a9 10 20 f7 d4 f0 dd 4c ba d4 4c a4 d7 20 63 d7
.:  d5fe 20 54 d7 90 03 20 ec d4 20 15 f2 20 54 d7 90 0a
.:  d60e a5 fb 8d 08 02 a5 fc 8d 07 02 20 23 d5 d0 0a 20
.:  d61e 63 d7 20 54 d7 90 d3 a9 10 85 b5 20 98 d7 20 0b
```

#

Editorial

Dieses Heft enthält erstmals in seiner Mitte ein Jahresinhaltsverzeichnis. Es zeigt für die hauptsächlich betreuten Systeme von Commodore (CBM) und Rockwell nebst Abkömmlingen (AIM 65) die durchschnittliche Menge von jeweils 4-5 Beiträgen je Heft. Das war noch nie so klar geworden, macht aber die immer wieder ausgedrückte Wertschätzung der Leser für diese Zeitschrift verständlich, daß sie in ihr zielgerichtete verwertbare Information finden. Der Herausgeber (51, und seit den Anfängen 1957 immer wieder mit der EDV befaßt) darf weiter anfügen: Im Verbund mit den an vielen Themen beschäftigten freien Autoren ist es in den vergangenen 3 1/2 Jahren gelungen, ein breites Spektrum an sehr guten Beiträgen zu veröffentlichen, die allen Lesern weitergeholfen haben.Diese Beiträge stellten oft Pionierleistungen dar, die von anderen Zeitschriften im In- und Ausland erst viel später in ähnlicher Form aufgegriffen wurden. In diesem Sinne seien die Leser ermuntert, nach kurzer Absprache weiterhin aus ihren Betätigungsgebieten zu berichten.

Es konnten sicher nicht alle interessierenden Bereiche abgedeckt werden. Wir haben die Sprachen PL 65, PASCAL und FORTH für unsere Rechner, Macro-Assembler, Wordprocessoren und weitere utilities. Merkwürdigerweise ist überall nur sehr selten über diese Werkzeuge berichtet worden, also auch hier.Es gibt also ein breites Feld für künftige Beiträge, um den Lesern die Benutzungsmöglichkeiten und Besonderheiten aufzuzeigen.

In der Produktfamilie kommen neue Rechner hinzu, die der Erklärung bedürfen. Gleichwohl bleibt die Frage zu stellen, wohin sich das Interesse der Betreiber weiter bewegt. Mit verschiedenen Beiträgen wurde von hier aus der 6809 von Motorola als das nächste höhere Sprungbrett vorgestellt. Es mag sich wie bei den fehlenden Beiträgen zu den genannten 'tools' um einen Zeitfaktor handeln, bis sich das Neue durchbeißt, wir haben es ja auch in den Anfängen dieser Zeitschrift gesehen, als es nur ganz wenige Autoren gab. Unabhängig vom Zeitfaktor und der möglichen Gewöhnung bleibt aber die Frage gleichwohl: Auf welche Themenkreise und Prozessoren ist künftig vermehrt einzugehen, ohne daß das bisher gewohnte Informationsinteresse der Leser darstellungsmäßig zu kurz kommt Die nächsten Rechnergenerationen mit 16-Bit–CPU sind lieferbar, natürlich auch teurer und damit nur für einen kleinen Kreis der Leser einsatzmäßig zu vertreten. Die für 6502 noch darzustellenden Themen werden uns noch lange beschäftigen. Gleichwohl werden der Markt und die Interessen sich weiterentwickeln. Daher die Bitte an die Leser, in Briefen die künftigen Erwartungen vor zuformulieren. Daraus soll im Versand mit dem Februar-Heft eine Leserbefragung formuliert werden. Diese Zeitschrift soll weiterhin auf das Leserinteresse abgestimmt sein. Ihre Antworten haben dabei ein großes Gewicht.

#

Kurt Peter, 6053 Obertshausen

BASIC-Formatierungen

Das nachstehende Unterprogramm ab Zeile 200 ermöglicht den komma-bündigen Ausdruck von Zahlenkolonnen. Die auszugebende Zahl wird in einen String umgewandelt. Die Variable II nimmt die Stellung des Kommas auf und tabuliert in Zeile 40 die erste Schreibstelle. - In den Zeilen 10 bis 60 ist das Demonstrationsprogramm für das rechtsstehende Beispiel enthalten.

```
10 FOR I=1 TO 20
20 A=10000/I
30 GOSUB200
40 PRINT#4,TAB(10-II);A
50 NEXTI
60 END
70 :
200 F$=STR$(A)
220 FORII=1 TO LEN(F$)
230 XX$=MID$(F$,II,1)
240 IF ASC(XX$)=46THEN RETURN
250 NEXT
260 RETURN
```

```
10000
 5000
 3333.33333
 2500
 2000
 1666.66667
 1428.57143
 1250
 1111.11111
 1000
  909.090909
  833.333333
  769.230769
  714.285714
  666.666667
  625
  588.235294
  555.555556
  526.31579
  500
```

Auch das Listbild von BASIC-Programmen läßt sich ähnlich wie in PASCAL optisch gliedern. Normalerweise verschluckt der Interpreter führende Blanks. Hinter einem führenden Doppelpunkt kann man jedoch beliebig Blanks anfügen:

```
10 : FOR I=1 TO 100
20 :     FOR J=1 TO 10
30 :         PRINTI;J
40 :     NEXTJ
50 : NEXTI
60 END
```

#

Jürgen Rüd, 7800 Freiburg

Disk - APPEND

Die nachfolgende Routine ist für CBM 3032 mit Floppy Disk geschrieben. Sie erlaubt ein APPEND an vorhandene Dateien. Sie wird mit SYS 28672"Name" aufgerufen. Um sie gegen ein Überschreiben durch BASIC zu schützen, sollte man POKE 53,112 eingeben.

```
.; 7000  A9 54     LDA #$54
.; 7002  A0 70     LDY #$70
.; 7004  20 1C CA  JSR $CA1C
.; 7007  20 3E F4  JSR $F43E
```

```
.; 700A  A9 08      LDA #$08
.; 700C  85 D4      STA $D4
.; 700E  A9 60      LDA #$60
.; 7010  85 D3      STA $D3
.; 7012  EA         NOP
.; 7013  EA         NOP
.; 7014  EA         NOP
.; 7015  EA         NOP
.; 7016  A4 D1      LDY $D1
.; 7018  D0 03      BNE $701D
.; 701A  4C 03 CE   JMP $CE03
.; 701D  20 0A F4   JSR $F40A
.; 7020  20 66 F4   JSR $F466
.; 7023  20 B6 F0   JSR $F0B6
.; 7026  A5 D3      LDA $D3
.; 7028  20 28 F1   JSR $F128
.; 702B  20 8C F1   JSR $F18C
.; 702E  20 8C F1   JSR $F18C
.; 7031  38         SEC
.; 7032  A5 2A      LDA $2A
.; 7034  E9 02      SBC #$02
.; 7036  85 FB      STA $FB
.; 7038  A5 2B      LDA $2B
.; 703A  E9 00      SBC #$00
.; 703C  85 FC      STA $FC
.; 703E  20 52 F3   JSR $F352
.; 7041  20 42 C4   JSR $C442
.; 7044  18         CLC
.; 7045  A5 1F      LDA $1F
.; 7047  69 02      ADC #$02
.; 7049  85 2A      STA $2A
.; 704B  A5 20      LDA $20
.; 704D  69 00      ADC #$00
.; 704F  85 2B      STA $2B
.; 7051  4C 79 C5   JMP $C579

.< 7054 93 12 44 49 53 4B 2D 41    ..DISK-A
.< 705C 50 50 45 4E 44 20 20 28    PPEND  (
.< 7064 43 29 20 31 30 2F 31 39    C) 10/19
.< 706C 38 31 20 42 59 20 4A 55    81 BY JU
.< 7074 45 52 47 45 4E 20 52 55    ERGEN RU
.< 707C 45 44 20 20 20 20 45 49    ED    EI
.< 7084 53 45 4E 42 41 48 4E 53    SENBAHNS
.< 708C 54 52 2E 36 36 20 20 20    TR.66
.< 7094 44 2D 37 38 30 30 20 46    D-7800 F
.< 709C 52 45 49 42 55 52 47 20    REIBURG
.< 70A4 20 20 00 AA AA AA AA AA      .*****                    #
```

****

Die Industrie- und Handelskammer für München und Oberbayern führt in 1. Hj. 1982 mehrere Seminare aus dem Bereich Mikroelektronik/EDV in ihrem Bildungszentrum Westerham durch. Erfolgreicher Einsatz von Kleincomputern/Einführung in die DV im Betrieb/Rationelle Abwicklung von EDV-Projekten/Mikroelektronik - Chance für das Management/Sicherung von EDV-Anlagen. Info: IHK-Bildungszentrum, V.-Adrian-Str. 5, 8152 Feldkirchen-Westerham, T. 08063-1941

computer shop

MICRO SYSTEM-BERATER

Computershop bietet zum Jahresende einige Sonderposten an, solange Vorrat reicht!

Video-plus I Video-Platine mit hoher Grafikauflösung	DM 770,--
Kartenkäfig für KIM-1, SYM-1 und AIM-Platinen	DM 72,--
PROTO-PLUS II, eine Prototypplatine im KIM-1-Format	DM 170,--

Dazu passend aus dem Normallieferprogramm:

MOTHER - PLUS II, eine Mutterplatine mit 5 Steckplätzen zur Systemerweiterung DM 395,--

DRAM PLUS 32K RAM Speicher und EProm Programmer für 2716 und 2732 sowie 4 Steckplätzen für diese Eproms. DM 1218,--

VIDEO PLUS II, Video-Platine mit sehr hoher Grafikauflösung, 92000 Punkte und der Möglichkeit ein Stand-Alone-Terminal damit zu bauen.
Es wird ein 6502 und ein 6551 benötigt.
Monitor-Programm auf Eprom bereits vorhanden DM 998,--

Rockwell AIM 65 und AIM 65-40 sind meistens ab Lager lieferbar.

AIM 65 mit 1K RAM DM 1260,--
AIM 65 mit 4K RAM DM 1423,--

Versand unfrei. Durch Vorkasse 3% Skonto.

Computershop GmbH Mangoldstaße 10 7778 MARKDORF
Telefon 07544 - 5656 und 3575 Telex 734 628 msb d

Rockwell-Original ROMs für AIM 65 für AIM 65 wegen Systemveränderung sehr günstig zu verkaufen: Assembler A65-010 DM 75,--, BASIC A65-020 DM 130,--, Monitor E/F-ROM DM 130,--. B. Kokula, Wredestr. 17, 6700 Ludwigshafen, Tel.: 0621 - 51 82 29.

Zum **"Computer des Jahres"** haben **europäische Fachjournalisten** den ATARI 400 gewählt. Kein Wunder, daß er Spitze ist, konnten die Entwickler dieses Jünglings doch auf die Erfahrungen mit den älteren und heutigen Konkurrenzprodukten bauen und weiterentwickeln! Wir bieten diesen Supercomputer zusammen mit Interface-K. und Feingrafik-Drucker (Nadeldrucker GP 80; Normalpapier 210 mm; Traktor; 80 Z/Z; Einzelpunktansteuerung) für zusammen nur DM 2390,-. Also: ATARI 400 ist kein Spielzeug, sondern der **modernste Mikrocomputer** in seiner Klasse!
Den Feingrafik-Drucker GP80 empfehlen wir wegen der hervorragenden Qualität und des günstigen Preises auch für Ihren Apple, AIM, KIM, PET, ... **Unser Preis: DM 986,- DM**
Wirth - Ihr COMPUTERPARTNER Mühlstr. 25, 7064 Grunbach, Tel.: 07151 - 71 226

CBM Hardware: Programmiergerät für 2K und 4K Eproms incl. Software 295 DM. Löschlampe 98 DM. 4 KBytes Ramplatine, steckbar in Romfassungen zum Austesten von Programmen **vor** dem Brennen, 195 DM. ROMBOX, schaltet mit POKE-Befehl 12 Eproms und 4 Char. Generatoren um, 295 DM. Für CBM 8032: deutscher Zeichensatz 155 DM. Info gratis bei Martin Roßmüller, Kaiserstr. 34, 5300 Bonn, ab 18 Uhr 22 48 37.

Wer hat Software für den CBM für Anwaltspraxis? Mitteilungen an P. Mees, Nonnenwerthstr. 43, 5000 Köln 41, Tel.: 0221 - 46 47 23.

DAIM - Floppycontrollerkarte unbestückt mit DOS in Eproms und Controlerbaustein (evtl. auch zusammengebaut) für AIM 65 zu verkaufen. Tel.: 0421 - 71 114.

Problem: Datenaufzeichnung mit Hilfe des CBM, die DATEV-kompatibel ist. Lösungen: Vorschläge und Ideeen an MICRO MAG oder Peter Mees, Nonnemwerthstr. 43, Köln 41, Tel.: 0221 - 46 47 23, Vergütung möglich.

Neue CBM-Software für den Programmentwickler und Anwender. BASIC/8 die Systemerweiterung für effiziente Stringverarbeitung und Bildschirmgestaltung. EDIT+ der Universaleditor für Text, Daten und Programme mit Großrechnerkomfort, erweiterbar um Macro-Assembler, Datenbankmodul, Testverarbeitung für höchste Ansprüche und: strukturiertes BASIC, alphanumerische Labels und Unterprogramme mit Parametern, PRECOMPILER, alles von Fa. Rudolph, Lentersweg 46, 2000 Hamburg 63. (Info anforfern, DM 1,-- Rückporto).

Micro-Computer auf der Basis des KIM-1 mit verbesserter Tastatur, externer Speichererweiterung, max. 16 K (teilw. bestückt), Data-Mega-Metallpapierdrucker, mit Ansteuerung, Stromversorgung 2x. +5V, 1x 24 V, 2x Batteriepufferung, teilweise mit Akku-Pufferung, mit UAD-Karte, zum Teil bestückt, in Schroff-Gehäuse; ohne Garantie; KIM zum Teil getestet, solange Vorrat reicht, gegen Vorkasse zu verkaufen. DM 980,--. Kontakt zum Lieferanten Chiffre 22/03

DUO Plott Interface für MX80 F/T

und COMMODORE Rechner 30/40/80

Paralleles IEEE 488 - Interface:

genormter IEEE-Stecker und Kabel
kompletter Zeichensatz des CBM-Computers
zwei Geräteadressen für Groß-/Grafikmodus
und Textmodus
alle Funktionen des EPSON bleiben erhalten
Floppy-Kompatibilität
Deutsche Umlaute, ß und Paragraph
Problemloser Einbau
inclusive Deutsches EPSON-Handbuch

Komplettpreis einschl. Kabel, CBM-Grafiksatz
und Handbuch incl. MwSt. DM 398,--

Umrüstsatz MX80 F/T auf MX82F/T

Aus Ihrem EPSON MX-80 F/T wird der MX-82 F/T
Einzelnadelansteuerung
erweiterter Befehlssatz
Elite-Schrift

Preis incl. MwSt. DM 250,--

Komplettpreis Interface, Kabel, Handbuch
und Umrüstsatz incl. MwSt. DM 600,--

Komplettpreis Interface, Kabel, CBM-Grafiksatz
und Handbuch
einschließlich neuem MX-80 F/T
und incl. MwSt. nur DM 1.998,--

STELLBERG COMPUTER SYSTEME

Blindenaaf 36 * 5063 Overath * Tel.: 02206 - 6644

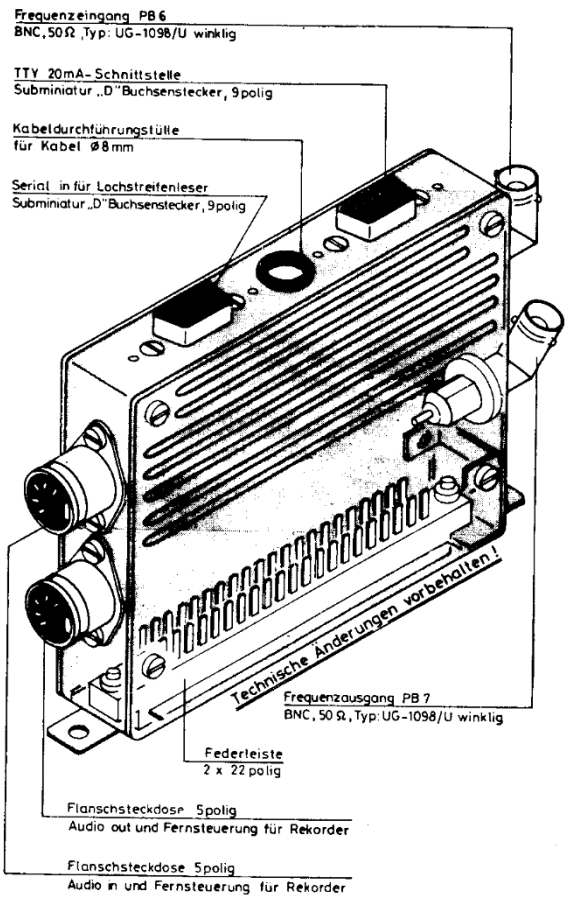
Universal-Steckverbinder

AIM 65 / Siemens PC 100

Bestückung nach Wunsch möglich

Lieferbar ab 45. KW. 81.

Preise auf Anfage



STECKER

INGENIEURBÜRO

Postfach 60 07 66 5000 Köln 60 Tel.: (0221) 74 51 12

Systemerweiterung für AIM 65 / PC 100

auf Europakarten

Einheitliches Format 100x160 mm, einheitlich nur 5 Volt. Vollständig kompatibel zum AIM-Systembus und zur AIM-Software. Alle Karten mit 44-pol. und 64-pol. a/c-DIN-Stecker lieferbar. Anschlußfertig und ausführlich getestet. 1 Jahr Garantie. - Die 64-pol. Karten sind mit unserem 6809-Europakartencomputer, der in diesem Jahre erscheinen wird, kompatibel.

32 kB RAM-Modul quasistatisch (keine CPU-Beeinflussung, DMA möglich), wie die anderen AIM-Systemerweiterungskarten auch ohne zusätzliche Hardware kompatibel zum Expansion-Connector des AIM und PC100. Adressierung in 16 - auch unzusammenhängenden - 2k-Segmenten. Mehrere Speicherkarten können durch bank select parallel betrieben werden. Stromversorgung 5V/0,8A für 32 kB. Lieferbar für AIM65, PC100, SYM, MCS-alpha.

792,- + MWSt = 894,96 DM
teilbestückt mit 16 k: 526,- + MWSt = 594,38 DM

VIDEO-Interface 615,- + MWSt = 694,95 DM

ANALOG E/A 8 Bit, max. 80 kHz Abtastrate, mit Aliasingfiltern und sample- and hold, E/A unabhängig voneinander
370,- + MWSt = 418,10 DM

EPROM-Karte 32 kB für 8x 2532 oder 2716 oder pinkompatible CMOS-RAMs. ICs einzeln beliebig adressierbar und einzeln selektierbar (für Umschaltung zwischen BASIC, PL65, FORTH, DOS etc.) ohne Eproms:
180,- + MWST = 203,40 DM

BUS-EXPANSION besteht aus: AIM Adapterkarte, Verbindungsflachkabel und Buskarte, Daten- und Adreßleitungen gepuffert, Schreibschutz für 16x4 KB, softwaregesteuerter Bankselect, 7 Steckplätze (erweiterbar), mit aktivem Busabschluß. Lieferbar für 64- (vorzugsweise) und 44-polige Karten Der 64-polige Bus entspricht unserem 6809-Bus.
240,- + MWSt = 271,20 DM

Neu: 32 KB CMOS-RAM, statisch 200 ns, Batteriepufferung und gemischte Bestückung m. EPROM 2716 mögl., für alle 8-Bit-CPUs geeignet
DM 884,- + MWSt = 998,92 DM

Die Alternative zum Nadeldrucker bietet vieles, was bisher nur die großen elektronischen Büromaschinen leisten. **Elektronische Typenrad-Schreibmaschine Praxis 35** und Computerinterface nach Wahl.
29 verschiedene Schriftarten, 3 Schriftgrößen (10, 12, 15 Zeichen/Zoll), 10 Zeichen Korrekturspeicher (list off und cover up), Carbon- und Textilfarbbandcassetten, Centronix-Schnittstelle, geschwindigkeitsoptimiert mit Nutzung des internen Schreibpuffers. Prospekt anfordern! Wir sind Olivetti-Händler und bieten vollständigen Service!

Preise: Praxis 35 ohne Interface, inkl. MwSt 1.398,--DM
Praxis 35 mit Centronix-Interface inkl MwSt 1.998,--DM

EPROM-Löschlampe mit E27-Schraubsockel DM 65,--.

DIPL.-ING. HORST NEUDECKER

INGENIEURBÜRO FÜR MESSTECHNIK

Mehringplatz 13 **Tel.: 030 - 251 20 00**
1000 Berlin 61 **030 - 251 45 18**

Video-Interface für AIM 65 / PC 100

Europakarte 100x160 mm, Stromaufnahme 5V, 0,6A, kompatibel mit den RAM-, ROM-, Analog-E/E-, Digitalinterfaces und anderen Baugruppen im Europaformat unseres Erweiterungssystems für AIM 65 und PC 100.

Die Videokarte wird mit dem Systembus (AIM-Expansion-Connector) direkt verbunden oder auf einen Steckplatz der gepufferten BUS-ERWEITERUNG gesteckt. Lieferbar mit 44-pol. Platinenrandstecker - Anschlußbelegung identisch mit AIM-Expansion- oder 64-pol. Stecker nach DIN.

Hardware: Video-RAM von $9000...$9800 unter uneingeschränktem Prozessorzugriff (1 MHz) und ständigem phasengesteuerten DMA des Videocontrollers (2 MHz). Bildstörungen durch µP-Tätigkeit ausgeschlossen. Controller: Motorola MC6845 mit allen Funktionen. Lichtgriffelanschluß. Norm-Video-Ausgangssignal. Die Videokarte ist für Erweiterung auf farbige Darstellung vorbereitet.

Bildformat: 24 Zeilen mit 80 Zeichen, 8x10 Punktmatrix. Zeichensatz im 4k-EPROM on-board: 128 Schriftzeichen, ASCII, groß und klein, deutsche Umlaute, griechische und mathematische Symbole, Inversdarstellung für Schrift möglich, 128 Grafiksymbole, davon 64 Zeichen Blockgrafik wie bei TRS 80, Bildschirmtext u. diversen Matrixdruckern; weitere 64 Grafikzeichen ähnlich cbm.

2 kB Videofirmware on-board nutzt und ergänzt Monitor, Editor, Assembler und die verschiedenen Basic-Versionen von AIM 65 und PC 100. Cursorsteuerung, erweiterter Editor, Fast-Modus für schnelles Assemblieren. Komfortabler Video-Texteditor mit schnellem Cursor-gesteuerten up/down-scroll, find, change, insert, read im gesamten Textspeicher. Dabei rollt der Text im Zusammenhang über den Bildschirm, störende Kommentare (T,B, U,D,F,C,I,R,J,N,Q,W,*) werden nicht eingefügt, 16 Zeilen vor der durch den Cursor markierten aktiven Textzeile und 8 folgende Zeilen sind sichtbar.

Option 1: zwei neue Monitor-ROMs mit auf 80 Zeichen verlängerter Editorzeile, Umschaltung auf Groß- und Kleinschreibung wie bei Schreibmaschinen (shift=groß) möglich. Anwendbarkeit aller neuen Editorbefehle auch im Schreibmaschinenmodus. M-List 16-stellig und M-change unter Cursorkontrolle 16-stellig. Automatische Video-Initialisierung beim Einschalten des Computers.

Option 2: Farbzusatz für AIM-Videokarte, RGB-Ausgang TTL-Pegel. Acht Vordergrund- und acht Hintergrundfarben, 40 Zeichen pro Zeile.

Preise:	AIM65-Videointerface	615,- +MWSt=	694,35 DM
	Option 1	86,- +MWSt=	97,18 DM
	Option 2	140,- +MWSt=	158,20 DM
	NEC-Daten-Monitor 12 Zoll, grün (P31), 20 MHz, für Lichtgriffel geeignet	582,- +MWSt=	657,66 DM
	NEC-Daten-Monitor 12 Zoll, farbig (RGB), für 25 Zeilen zu je 80 Zeichen, 640 Pkte. hor.	DM 2478,- +MWSt=	2814,40 DM

32 KB RAM-Modul zum Einstecken. Das erste Modul der neuen Serie zum direkten Einstecken in PC 100/AIM 65 ist da. Die Steckerleiste mit dem Systembus bleibt für externe Zusätze frei, die ROM-Sockel werden nicht verdeckt.

Preis für 32 KB	DM 692,- + MWSt	= DM 781,96
teilbestückt 16 KB	DM 426,- + MWSt	= DM 481,38

Neuerscheinungen der Europakartenserie: 64 KB und 128 KB RAM-Karte, CPU-Karte mit 32 KB RAM, RIOT/VIA-Karte. Bitte Unterlagen anfordern!

DIPL.-ING. HORST NEUDECKER

INGENIEURBÜRO FÜR MESSTECHNIK

65xx MICRO MAG

COMPUTING · SOFTWARE · HOBBY

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

65xx MICRO MAG erscheint zweimonatlich. - Offsetdruck: Druckartist Gerhard M. Meier, Hamburg 70.

Bezugsbedingungen: Abonnement ab laufender Ausgabe für 6 Hefte DM 49,-- (Inlandsendpreis). Ausland/Foreign via surface mail DM 54,--, USA air $ 28. Abonnements laufen bis auf Widerruf mit Kündigungsmöglichkeit bis zu zwei Wochen vor deren Ablauf.

Nachliefermöglichkeiten: Die Hefte 1-6 sind nur noch als Buch lieferbar, siehe Anzeige unten.

Die Hefte 7-21 können einzeln oder komplett nachgeliefert werden, und zwar zum Endpreis von DM 7,80/Stück. Hefte Nr. 10 und 13 sind bald vergriffen, im Januar 1982 erscheint jedoch 'Das Buch 7-13 des 65xx MICRO MAG', siehe Inhaltsverzeichnis im Heftinneren.

Private Besteller werden um Überweisung oder Scheck zusammen mit der Bestellung gebeten. Richten Sie bitte Ihre Überweisungen auf das Konto Roland Löhr, Nr. 654 70-202 Postscheckamt Hamburg, BLZ 200 100 20.

Leser-Service des Herausgebers

Thermopapier

für AIM 65/PC 100 in kontrastreicher Spitzenqualität.
Packung mit 8 Großrollen, zus. 520 m, preiswert DM 50,85

Thermokopf/Printerplatte für AIM 65

und PC 100 mit Einbauanleitung, Auffrischung des Druckes. DM 25,--

Bücher:

Das Buch 1-6 des 65xx MICRO MAG
ca. 230 Seiten, Sammelband der Hefte 1-6 dieser Zeitschrift, geb., ohne Anzeigen.Das wertvolle Arbeitsbuch mit einem Leitfaden für die Programmierung und den vielen grundlegenden Darstellungen DM 26,--

6502 Software Design
von L. J. Scanlon. Das beliebte Lehrbuch für die Programmierung, ca. 270 Seiten mit vielen verständlich aufgebauten Beispielen DM 36,--

Programming & Interfacing the 6502
'with experiments' von Marvin L. de Jong, ca. 410 Seiten, viele Schaltungsvorschläge, didaktisch gegliedert, besonders geeignet für den Anfänger DM 59,--

Das Buch 7-13 des 65xx MICRO MAG
erscheint im Januar 1982, siehe Inhaltsverzeichnis in der Heftmitte.

Weitere Bücherangebote im Inneren dieses Heftes