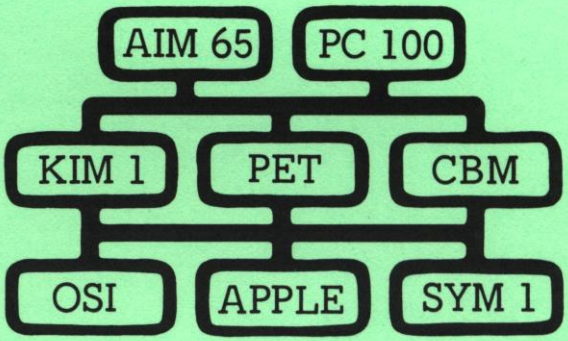
# 65xx MICRO MAG

COMPUTING · SOFTWARE · HOBBY

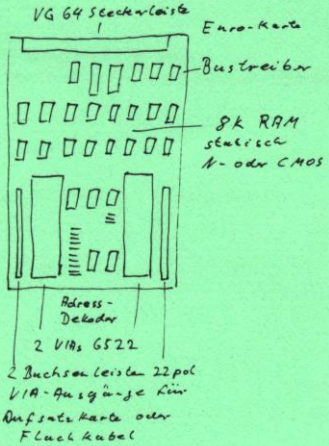
DM 8,50 Nr. 20 August 1981




## Inhaltsverzeichnis

Prozeßtechnik mit Mikro-Computern (1) ..... 3
Cross Reference List für BASIC-Variable (CBM) ..... 10
Zweidimensionale Felder sortieren (CBM) ..... 12
EPROM-Programmiereinheit für AIM 65 ..... 18
Softwareentwicklung für AIM 65 auf PDP 11 ..... 26
APPLE II emuliert AIM 65 ..... 31
Object-Code Editor (AIM) ..... 41
MC 6809: Befehle (2) ..... 50
Softwarebesprechung, Buchbesprechung ..... 59
Produkte ..... 60
AIM Spezial (9) ..... 61
Editorial ..... 61

# Combo-Karte




# Preissenkung !!!

2 VIAs 6522 = 40 Pin I/O und
8 kByte NMOS RAM
Busse gepuffert, VG 64 Steckverbinder

DM 388,--

dito mit CMOS RAMS

DM 442,--

dito nur mit NMOS RAM

DM 258,--

dito nur mit CMOS RAM

DM 312,--

dito nur mit VIAs

DM 160,--

zzgl. MwSt.

ELEKTRONISCHE BAUGRUPPEN
MICROPROZESSOR-TECHNIK
Dohmann

Im Wiehagen 15 - Tel. (05241) 67480
4830 Gütersloh 12

Bernhard Kokula, 6700 Ludwigshafen

# Prozeßtechnik mit Micro-Computern (1)

Am Beispiel eines mit Sonnenkollektoren beheizten privaten Schwimmbeckens werden Hard- und Software-Bausteine beschrieben. Besonders bei der Hardware wird darauf geachtet, nur preiswerte Lösungen mit gängigen Bauelementen auszuwählen. Eine typische Bauanleitung wird allerdings nicht gegeben. Die Bausteine sollen unabhängig voneinander und zur Verwendung in anderen Applikationen kombinierbar bleiben. Für die Verwirklichung der Vorschläge bieten sich derzeit mehrere 'single board computer' an, z.B. a) Der Junior-Computer der Zeitschrift ELEKTOR, b) der ENUF der Zeitschrift MC [2] oder der MINI 2000 von H. Regge, Bremen [3], der während der Entwicklungsphase an den AIM 65 paßt.

Für die Darstellungen zur Prozeßtechnik sind folgende Abschnitte vorgesehen.

1.) 8-fach Analogeingabe
2.) Meßpunkt-Aufbereitung und Filterung
3.) 2-Punkt-Regler-Ausgang für Pumpensteuerung
4.) 3-Punkt-Regler-Ausgang für Mischventil
5.) Multiplex-Ausgabe von internen (Meß-)Werten
   a) auf 7-Segmentanzeigen
   b) auf Analoganzeiger
6.) Störmelde-Ausgaben.

Die Software wird in Assembler dargestellt, wobei weitgehend die Rechenfähigkeit der Assembler ausgenutzt wird, um durch wenige Parameter während des Assemblierens einen optimalen, der jeweiligen Aufgabenstellung angepaßten Objektcode und zugleich Listings mit guter Lesbarkeit zu erhalten.

**Teil 1: 8-fach Analogeingabe**

Die industriell verwendeten Analog-Digital-Wandler sind weder preiswert, noch überall erhältlich und schon gar nicht einfach anzuwenden.

Die meisten Meßwerte lassen sich als Widerstandswert darstellen, für die die folgenden Fühler billig sind:

| | |
|---|---|
| Wärme | : Kaltleiter, Heißleiter |
| Licht | ; Fotowiderstand |
| Stellung, mech. | : Potentiometer. |

Mit dem preiswerten Treiberbaustein (TDB) 555 lassen sich diese Widerstände mit wenig Hardwareaufwand nach einem Vorschlag von Siemens [1] in Digitalwerte umwandeln. Dies geschieht in 2 Schritten:

Zuerst wird der Widerstandswert in ein zeit-proportionales '1'-Signal mit Hilfe des 555 umgewandelt, danach wandelt der Microcomputer die Dauer des Logiksignales über einen internen Timer in den Digitalwert. Zur Synchronisation wird der 5555 in der monostabilen Betriebsart benutzt und vom Microcomputer über seinen Triggereingang zugleich mit dem rechnerinternen Timer gestartet. Nun wartet der Microcomputer bis das Ausgangssignal des 555, das beim Start '1' wurde, wieder zu '0' wird. Dann liest er seinen Timer aus und hat damit den widerstandsproportionalen Digitalwert.

Für jeden Analog-Eingang benötigt man einenEingangsport-Pin. Triggern kann man aber alle (8) 555 zugleich und dann nacheinander immer nur einen auslesen. Man benötigt dann für beliebig viele 555 nur einen Ausgang.

Es geht natürlich auch umgekehrt: Jeder 555 hat seinen Triggerausgang vom Microcomputer, alle Ausgänge der 555 sind über eine Diode 'geodert' und werden über einen gemeinsamen Eingang des Microcomputers abgefragt.. - Die Triggerung der 555 läßt sich auch als Kaskade aufbauen: Es wird immer der folgende beim'1/0'-Übergang des Ausganges vom Vordermann angestoßen, reihum.

*Besonders vorteilhaft beim 555 ist seine Anschlußfähigkeit an so leistungsarme Ports wie die des 6522 im AIM 65, die nur eine TTL-Last bieten, und natürlich, daß er direkt mit den 5 Volt des Microcomputers arbeitet.*

*Bild 1 zeigt die verwendete Grundschaltung des 555. Mit dem 10k-Trimmpot am Pin 5 läßt sich bei Bedarf ein Feinabgleich des Zeitbereiches bei gleichem R-C-Glied erreichen. Der 1k-Widerstand dient als Pull-up. Der 100-Ohm-Widerstand am Pin 3 dient dem Schutz des verwendeten I/O-Ports bei möglicher Falschinitialisierung als Ausgang, der 555 verträgt 200 mA 'sink und source'!*

*Der Triggerimpuls darf von 1µ Sek bis zur Meßzeit 't' '0' sein. Der 1 kOhm vor 'R' dient der Strombegrenzung bei '$R_{min}$' oder Kurzschluß.*

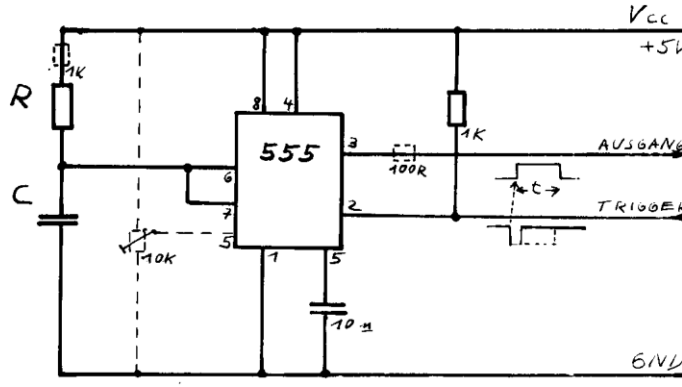



**Bild 1: Monostabile Grundschaltung**

**Tabelle 1: Anschlußbelegung des ...**

| 555 | 556/1 | 556/2 | |
|---|---|---|---|
| 1 | - 7 | - | Ground (-5V) |
| 2 | 6 | 8 | Trigger |
| 3 | 5 | 9 | Ausgang |
| 4 | 4 | 10 | Reset |
| 5 | 3 | 11 | Kontrollspannung |
| 6 | 2 | 12 | Schaltschwelle |
| 7 | 1 | 13 | Entladung |
| 8 | - 14 | - | $V_{cc}$ (+4,5 - +16) |

$$R_{(k)} \times C_{(\mu F)} = t_{(ms)} \times \frac{1}{1,1}$$

$R_{MAX}$ = 1 MOhm (Folien-ond. 20 MOhm)

$R_{MIN}$ = 1 KOhm

$t_{MAX}$ (Timer 1 = 65 msek)

= R (100 kOhm) x C (0,47µF)

*Die Grenzen des R/C-Gliedes zur Zeitmessung sind:*

*R von 1 k bis 1 MOhm (Folienkondensatoren bis max. 20 MOhm),*
*C von 100 p-1000 Farad (Begrenzung nur durch Elko-Reststrom).*

*Der Zeitfaktor bei unbeschaltetem Pin 5 beträgt*

$$\frac{t}{(sek)} = 1{,}1 \times \frac{R}{(Ohm)} \times \frac{C}{(Farad)}$$

*oder in etwas gängigeren Dimensionen*

$$\frac{t}{(msek)} = 1{,}1 \times \frac{R}{(kOhm)} \times \frac{C}{(\mu Farad)}$$

*Es ist eine Besonderheit, daß Temperatur- und Spannungsschwankungen beim 555 Zeitfehler nur im %o (Promille)-Bereich verursachen, die damit vernachlässigbar sind. $t_{MAX}$ muß kleiner/gleich 65 msek sein, entsprechend hex FFFF für Timer T1 (16-Bit-Zähler). Bei R= 100 kOhm ergibt dies ein C=0,47 µF.*

*Bild 2 zeigt den Anschluß von 8 'A/D-Wandlern 555' an einem VIA 6522 (z.B. AIM 65). Tabelle 1 erklärt die Anschlüsse des 555 und seines 'Dual-Bruders' 556.*

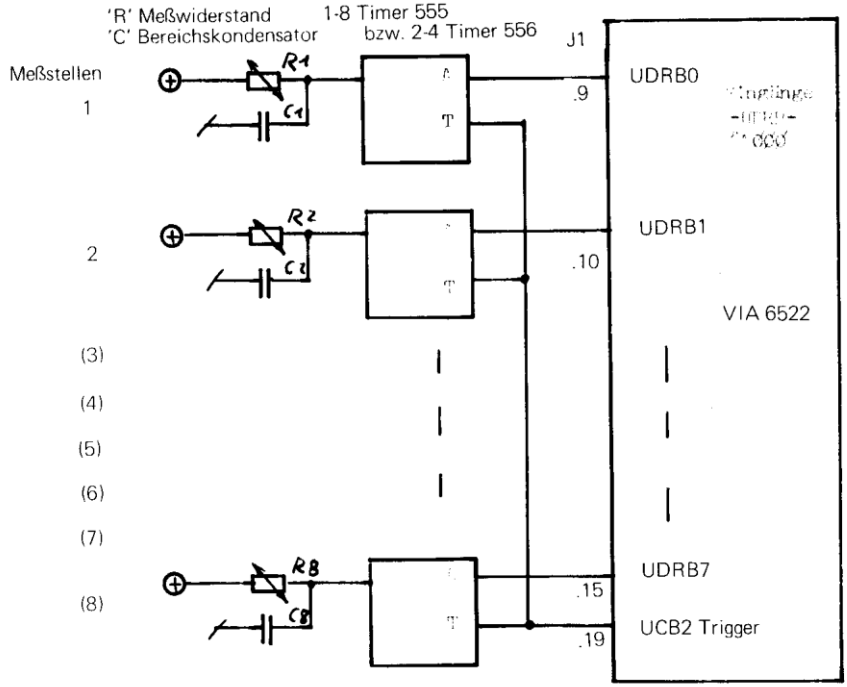



**Bild 2: Anschluß am AIM 65, User-Port B**

*Beschreibung der Software*

*INIT dient der Initialisierung der VIA 6522, Ports und Timer 1,*
*MUX555 ist der erste 'Prozeßbaustein' und bedient bis zu 8 A/D-Wandler,*
*ANTEST gibt 2 der Meßwerte analog auf den Bildschirm aus.*
*DITEST ist eine Prüfroutine für die 8 Digitalwerte,*

*Die Bausteine sind als lauffähiges Programm geschrieben mit Start auf der Taste USER-F1. ZYKLUS kann mit Werten von 1-8 assembliert werden. Zeropage und Programm werden dann ent-*

*sprechend der Anzahl der Eingänge beim Assemblieren optimiert. Wenn im Programm 'unterwegs' die Betriebsart von Timer 1 verändert wird, so muß INIT TIMER 1 mit in den MUX555 hineingenommen werden. MUX555 holt alle Meßwerte von draußen herein und übergibt sie an die 'HEX-REGISTER', andere Bausteine werden sie von dort übernehmen.*

*ANALOG TEST gibt 2 der Meßwerte als Analogwert auf dem Bildschirm aus, dazu wird durch 4 geteilt, damit der Wert kleiner 64 bleibt. Wenn man durch 16 teilt (4x LSR A), paßt die Darstellung auch in das Display des AIM 65. Mit ERST bestimmt man den ersten auszugebenden Meßwert. ERST muß immer kleiner sein als ZYKLUS.*

*TIMEFAKTOR bestimmt, wie oft Timer 1 65 ms Wartezeit vor Beginn der nächsten Runde einlegt, um nicht zu schnelle Abläufe zu haben. Will man ANTEST und DITEST abschalten, so 'JUMPT' man einfach darüber hinweg. Zum Testen genügen 2 St. 555-A/D-Wandler, man kann dann die anderen Eingänge durch Parallelschalten mit den gleichen Werten bedienen.*

*Zum Schutz vor Leitungsbruch der Meßwiderstände wird der Timer 1 auch noch auf '00' abgefragt. Unter der Voraussetzung, daß im Normalfall die Meßzeit die maximalen 65 ms nicht erreicht, wird auch '00' nie erreicht (der Timer läuft vom Sollwert rückwärts nach '00'). Beim Erreichen dieser '00' erfolgt hier ein Aussprung und Programmabbruch. Also Vorsicht bei der Inbetriebnahme, das Programm hat keinen Fehler, ggfs. ist die Zeit länger als 65 ms. Diese Fehlermeldung zusammen mit der Meßstellenmaske oder dem Pointer (X) kann leicht zu einer individuellen Fehlerbehandlug oder Meldung verarbeitet werden. Es ist auch möglich, ein Flag zu setzen und die Messung in Abhängigkeit davon dann auszulassen.*

*Im MUX555 werden die Möglichkeiten des BIT-Befehles mehrfach ausgenutzt:*

| | | |
|---|---|---|
| *bei MU2* | *(BIT UIFR)* | *fragen wir das Interrupt-Bit von Timer 1 ab, ob ein Nulldurchgang* |
| | *(BVC MU3)* | *vorliegt. Es ist Bit 6 im Interrupt Flag Register. BIT transportiert seinen Zustand in das V-Flag.* |
| *bei MU3* | *(BIT UDRB)* | *wird die AND-Eigenschaft von BIT ausgenutzt. Im Akku liegt von* |
| | *(BNE MU2)* | *LDA SHIFT (weiter oben) her die Meßstellenmaske.* |

*Der 0-Impuls an UCB2 zum Triggern nutzt die Programmierbarkeit von UCB für das zu erzeugende Ausgangssignal aus. Dafür wird das Kontrollregister UPCR benutzt:*

| | |
|---|---|
| *für Bit 7,6,5 = '1'* | *ist UCB2='1'. Durch ASL UPCR entsteht: Bit 7,6='1', Bit 5='0'. Das bedeutet: UCB2='0'. ROR UPCR stellt den alten Zustand her.* |

*Diese Art der Impulserzeugung geht allerdings nur dann, wenn Bit4 im UPCR '0' ist (UCB1 ist nicht gesteuert) und wenn das Carry zwischenzeitlich nicht verändert wird. EOR =$FF ist nötig, wenn der Hex-Meßwert widerstandsproportional sein soll, da der Timer 1 rückwärts zählt.*

Literaturhinweise:
1) Siemens Bauteile-Report, Heft 6/77, S. 230.
2) mc-Microcomputer Zeitschrift, Heft 2/81, S. 20.
3) Hans-Joachim Regge, Bremen (T. 0421-71114) Europaboard 100x160 mm, KIM/AIM-Stecker. Bestückung 6502, 2x6522, 2x2716, 2x2114, 7805. Ohne 6502 = KIM/AIM/SYM-buskompatible Erweiterung. Mit 6502= single board computer.

```
                  ---------------------
0000              PROZESSTECHNIK TEIL 1
0000
0000
0000              ;                          27.6.81/KOKULA
0000              ;
0000              ; 1 BIS 8 '555'A/D-WANDLER, 1 BYTE
0000              ;
0000              ;      +++++ 'ZEROPAGE & KONSTANTEN'
0000              ;
0000 ZYKLUS              =8                ;FUER 1-8 MESSWERTE
0000              ;ASSEMBLER OPTIMIERT PROGRAMM U. ZEROPAGE
```

```
0000                    *=0
0000 HEXREG             *=*+ZYKLUS ,EINGABE REGISTER
0008 SHIFT              *=*+1               ;MESST.ABFRAGE-MASKE
0009            ;
0009                    *=$10C              ;USER-F1
010C            4C0002 JMP START
010F            ;
010F            ;
010F            ;       +++++ 'PROGRAMM-TEIL INIT'
010F            ;
010F                    *=$0200             ;***
0200
0200            ;INITIALISIERUNG TIMER 1 UND PORT B
0200            ;(USER-VIA 6522 AIM.65)
0200 START
0200            A97F    LDA #%01111111      ;DISABLE
0202            8D0EA0 STA UIER             ;INTERUPT
0205            A900    LDA #0              ;ALLE BIT =0; IN
0207            8D02A0 STA UDDRB            ;DAT.DIR.REG.B
020A            ;
020A            ;INIT TRIGGER LO-IMPULS AN UCB2
020A            ;ZUM STARTEN DER A/D-WANDLER
020A            AD0CA0 LDA UPCR             ;HOLE OLD
020D            09E0    ORA #%11100000      ;BIT 5,6,7=1; UCB2=1
020F            8D0CA0 STA UPCR
0212
0212            ;
0212            ;INIT TIMER 1
0212            AD0BA0 LDA UACR             ;HOLE OLD
0215            293F    AND #%00111111      ;ONE SHUT MOD, BIT 6+7=0
0217            8D0BA0 STA UACR
021A            A9FF    LDA #$FF            ;VOLLE ZEIT LADEN
021C            8D05A0 STA UT1CH            ;T1 1 X START
021F            ;
021F            ;
021F            ;       +++++ 'PROGRAMM-TEIL MUX555'
021F            ;
021F MUX555
021F            A201    LDX #%00000001
0221            8608    STX SHIFT           ;MASKE FUER PORTABFRAGE
0223            CA      DEX                 ;X=0
0224 MU1
0224            2C0DA0 BIT UIFR             ;TEST TIMER 1 ABGELAUFEN
0227            50FB    BVC MU1             ;NOCH NICHT, BIT 6=0
0229            A508    LDA SHIFT           ;MESSTELL.MASKE
022B            A0FF    LDY #$FF
022D            8C04A0 STY UT1L             ;LADE T1-LO
0230            0E0CA0 ASL UPCR             ;UCB2=TRIGGER-0-PULS
0233            8C05A0 STY UT1CH            ;LADE T1 UND STARTE
0236            EA      NOP                 ;PULS LAENGE EINSTELLEN
0237            6E0CA0 ROR UPCR             ;TRIGGER-PULS END
023A MU2        2C0DA0 BIT UIFR             ;ERROR FALLS T1 '00'
023D            5003    BVC MU3             ;NEIN
023F            4C72FF JMP ERROR            ;PROGRAMM ABBRUCH
0242 MU3        2C00A0 BIT UDRB             ; FRAGE PORT B
0245            D0F3    BNE MU2             ;ZEIT LAEUFT NOCH
0247            ;
```

```
0247            ;AUSGEWAEHLTES PORT-BIT IST '0' GEWORDEN
0247            ;ERGEBNIS LIEGT NUN IM TIMER T1
0247            AD05A0 LDA UT1CH        ;NUR HI
024A            49FF   EOR #$FF         ;INVERTIERE DATE
024C            9500   STA HEXREG,X    ;AUFFANG-REG
024E            0608   ASL SHIFT       ;NAECHSTER MESSWERT
0250            E8     INX              ;NAECHSTE RUNDE
0251            E008   CPX #ZYKLUS      ;LETZTE RUNDE ?
0253            D0CF   BNE MU1          ;NEIN
0255            ;
0255            ;
0255            ;      +++++ 'PROGRAMM-TEIL ANTEST'
0255            ;
0255            ;ZUR ANALOGAUSGABE VON 2 MESSWERTEN
0255 ERST           = 2                 ;(0-6) ERSTER MESSWERT
0255 LEZT           =ERST+1             ;2.MESSWERT
0255            ;
0255 ANTEST
0255            A002   LDY #ERST
0257 AN0        20F0E9 JSR CRLF
025A            B90000 LDA HEXREG,Y     ;HOLE DEN WERT
025D            2046EA JSR NUMA         ;GIB IHN AUS
0260            A93A   LDA #':'         ;KENNZEICHEN 1.MESSW.
0262            C002   CPY #ERST
0264            F002   BEQ AN1
0266            A92E   LDA #'.'         ;KENNZEICHEN 2.MESSW.
0268 AN1        20BCE9 JSR OUTALL
026B            B90000 LDA HEXREG,Y     ;NOCHMAL
026E            4A     LSR A
026F            4A     LSR A            ;GETEILT DURCH 4
0270            AA     TAX
0271 AN2        A920   LDA #$20         ;ALS 'BLANK'ZAHL AUSGEB.
0273            20BCE9 JSR OUTALL
0276            CA     DEX
0277            10F8   BPL AN2
0279            A92A   LDA #'*'         ;DAHINTER DEN '*'
027B            C002   CPY #ERST
027D            F002   BEQ AN3
027F            A92B   LDA #'+'         ;2.MESSW. DEN
0281 AN3        20BCE9 JSR OUTALL
0284            C8     INY
0285            C004   CPY #LEZT+1
0287            D0CE   BNE AN0
0289            ;
0289            ;
0289            ;      +++++ 'PROGRAMM-TEIL DITEST'
0289            ;
0289            ;DIGITAL AUSGABE DER 1-8 HEXWERTE IN 1 ZEILE
0289            ;
0289 DITEST
0289            20F0E9 JSR CRLF
028C            A005   LDY #5           ;5 STELLEN EINRUECKEN
028E DI1        A920   LDA #$20         ;'SPACE'
0290            20BCE9 JSR OUTALL
0293            88     DEY
0294            D0F8   BNE DI1
0296            A000   LDY #0           ;RUNDEN ZAEHLER
```

```
0298 DI2
0298            203BE8 JSR BLANK2
029B DI3        B90000 LDA HEXREG,Y    ;HOLE DATE
029E            2046EA JSR NUMA        ;GIB AUS
02A1            C8     INY
02A2            C008   CPY #ZYKLUS     ;FERTIG ?
02A4            D0F2   BNE DI2         ;NEIN
02A6            ;
02A6            ;WARTEZEIT VOR DER NAECHSTEN RUNDE
02A6            ;ZUR VERLANGSAMUNG BEIM TEST
02A6 TIMFAK             =10            ;(01-FF) WARTEZEITFAK.
02A6            ;
02A6            A20A   LDX #TIMFAK
02A8            20B102 JSR WAIT        ;WARTEZEIT
02AB            2007E9 JSR RCHEK       ;STOP WANTED ?
02AE            4C1F02 JMP MUX555      ;EIN NEUES SPIEL
02B1            ;
02B1 WAIT
02B1            ;WARTE-ZEIT MIT T1
02B1            ;'N' X 65 MS, 'N' IN <X> MITBRINGEN
02B1            A9FF   LDA #$FF
02B3            8D05A0 STA UT1CH       ;LADE UND START T1
02B6 WA1        2C0DA0 BIT UIFR        ;TEST INT.FLAG
02B9            50FB   BVC WA1         ;ZEIT LAEUFT NOCH
02BB            CA     DEX
02BC            10F3   BPL WAIT        ;NOCH 'NE RUNDE
02BE            60     RTS
02BF            ;
02BF            ;      +++++ 'EQUATES'
02BF            ;
02BF BLANK2              =$E83B
02BF CRLF                =$E9F0
02BF ERROR               =$FF72        ;PRINT AIM
02BF NUMA                =$EA46
02BF OUTALL              =$E9BC
02BF RCHEK               =$E907        ;PROGRAMM-STOP MOEGLICH
02BF            ;
02BF            ;      +++++ 'VIA 6522-USER'
02BF            ;
02BF UDRB                =$A000        ;DATA REG B
02BF UDDRB               =$A002        ;DIR  REG B
02BF UT1L                =$A004        ;TIMER 1,LOW
02BF UT1CH               =$A005        ;TIMER 1,HIGH
02BF UACR                =$A00B        ;AUX CONTR REG
02BF UPCR                =$A00C        ;PER CONTR REG
02BF UIFR                =$A00D        ;INT FLAG  REG
02BF UIER                =$A00E        ;INT ENABL REG
02BF                     .END
02BF          ERRORS= 000
```

Ω

Dipl.-Ing. B. Dohmann, 4600 Dortmund 70

# Cross Reference List für BASIC-Variable (CBM)

BASIC ist eine Programmiersprache, die besonders gut für interaktives Programmieren geeignet ist. Im Gegensatz zu anderen Sprachen wie COBOL oder FORTRAN lassen sich viele Programme direkt am Rechner erstellen, ohne daß vorher ein Konzept geschrieben werden müßte. Dabei kommt aber die Dokumentation oft zu kurz. Wenn Wochen oder Monate nach Programmerstellung eine Änderung oder Erweiterung notwendig wird und wenn nur das Programmlisting als Dokumentation vorhanden ist, muß sich der Programmierer meist erst mühsam wieder einarbeiten.

Eine kleine Erleichterung ist dabei das vorliegende Programm. Es erstellt eine alphabetisch sortierte Liste aller in einem Programm benutzten Variablennamen. Zu jeder Variablen wird angegeben, in welchen Programmzeilen sie benutzt und in welchen ihr ein Wert zugewiesen wird. Zeilennummern mit eckigen Klammern bedeuten dabei Wertzuweisungen. Sie geschehen außer durch das Gleichheitszeichen ja auch durch BASIC-Anweisungen: FOR X = ..., INPUT X, INPUT#Y,X und GET X bzw GET#Y,X.

Bedienung.

Das zu untersuchende Programm muß sich auf einer initialisierten Diskette in der Floppy-Disk befinden. Nach dem Starten des Programmes CROSSREFERENCE muß der Name des zu untersuchenden Programmes eingegeben werden. Nach der Analyse wird die Variablenliste auf den Bildschirm und (wenn angeschlossen) einen Drucker mit der Gerätenummer 4 ausgegeben.

Wirkungsweise.

Wie bekannt ist ein BASIC-Programm in folgender Form in einem File abgespeichert: Die ersten zwei Byte des Files geben die Startadresse des Programms an. Sie werden nicht benötigt und deshalb einfach überlesen (Zeile 110). Dann folgen die einzelnen BASIC-Zeilen. Die ersten zwei Byte (Link Bytes) enthalten die Adresse der nächsten Zeile. Enthalten beide eine Null, so ist damit das Ende angezeigt. Das Programm verzweigt dann zur Ausgaberoutine (Zeile 210). Die nächsten beiden Byte enthalten die Zeilennummer in binärer Form (Zeilen 200 - 260).

Alles folgende ist Programmtext und kann auf Variablennamen untersucht werden. Dabei wird der Text zwischen Anführungszeichen (Zeilen 310 - 340) und hinter einem REM (Zeilen 400 - 420) überlesen. Wird eine Null gefunden, so bedeutet dies das Ende der Zeile. Ist eine Variable gefunden, wird mit einem binären Suchverfahren in den Zeilen 605 bis 650 geprüft, ob sie schon vorhanden ist. Ist dies nicht der Fall, so wird sie in die Liste eingefügt (Zeilen 700 - 710). Ansonsten wird die entsprechende Zeilennummer angehängt.

Die Variablen A$(X) enthalten dabei die Variablennamen, B$(X,0) die Nummern der Zeilen, in denen der Variablen ein Wert zugewiesen wird und B$(X,1) die restlichen Zeilennummern. In den Zeilen 1000 - 1040 wird die Variablenliste schließlich ausgegeben.

```
0 PRINT:PRINT"CROSS-REFERENZ / 1981 B.DOHMANN
1 PRINT:PRINT"DIESES PROGRAMM ERZEUGT AUS AUS EINEM
2 PRINT"PROGRAMM, DAS SICH AUF EINER INITIALI-
3 PRINT"SIERTEN DISKETTE IN DER FLOPPY BEFINDEN
4 PRINT"MUSS, EINE ALPHABETISCH SORTIERTE LISTE
5 PRINT"ALLER BENUTZTEN VARIABLEN.
6 PRINT"DIE ZAHLEN BEZEICHNEN DIE PROGRAMMZEI-
7 PRINT"LEN, IN DENEN DIE VARIABLE BENUTZT WIRD.
8 PRINT"ZAHLEN IN ECKIGEN KLAMMERN SIND ZEILEN,
9 PRINT"IN DENEN DIESER VARIABLEN EIN WERT ZU-
10 PRINT"GEORDNET WIRD.
50 DIMA$(200),B$(200,1):A$(0)="ZZZ":AZ%=0
```

```
100 INPUT"PROGRAMM-NAME";FI$
110 OPEN3,8,3,FI$+",P,R":GET#3,A$,A$
200 REM LINK-BYTES
210 GET#3,A$,B$:IFA$+B$=""THENCLOSE3:GOTO1000
220 REM ZEILENNUMMER
230 GET#3,A$,B$:IFA$=""THENA$=CHR$(0)
240 IFB$=""THENB$=CHR$(0)
250 ZN$=A$+B$:G=0:H=0
260 PRINTCHR$(20);PRINTASC(B$)*256+ASC(A$);
270 GET#3,A$
300 IFA$=""GOTO200:REM ZEILENENDE
302 IFA$=":"ORA$=CHR$(167)THENG=0
305 IFA$="("THENH=1
306 IFA$=")"THENH=0
310 IFA$<>CHR$(34)GOTO400:REM "
320 GET#3,A$:IFA$=""GOTO200
330 IFA$=CHR$(34)GOTO270
340 GOTO320
400 IFA$<>CHR$(143)GOTO500:REM REMARK
410 GET#3,A$:IFA$=""GOTO200
420 GOTO410
500 A=ASC(A$)
502 IFA>127THENIFA<>129ANDA<>132ANDA<>133ANDA<>136ANDA<>161ANDA<>167
505 IFA<65ORA>90GOTO270                                          <THENG=1
510 V$=A$
520 GET#3,A$:IFA$=""GOTO600
530 A=ASC(A$):IF(A>47ANDA<58)OR(A>64ANDA<91)THENV$=V$+A$:GOTO520
540 IFA=36ORA=37THENV$=V$+A$:GET#3,A$
600 IFA$="("THENV$=V$+"()"
602 IFG=0ANDH=0THENPRINT"<";
605 X=AZ%/2:Y=X
610 IFA$(X)=V$GOTO800
630 Y=(.5+(A$(X)>V$))*ABS(Y):IFABS(Y)>.4THENX=X+Y:GOTO610
650 X=INT(X-(Y>0))
700 FORJ=AZ%TOXSTEP-1:A$(J+1)=A$(J):B$(J+1,0)=B$(J,0):B$(J+1,1)=B$(J,1):
710 AZ%=AZ%+1:B$(X,0)="":B$(X,1)=""                              <NEXT
800 A$(X)=V$:J=G:IFH>.5THENJ=1
810 IFRIGHT$(B$(X,J),2)<>ZN$THENB$(X,J)=B$(X,J)+ZN$
820 PRINTV$;:IFJ<.5THENPRINT">";
830 PRINT",";
840 GOTO300
1000 D$="VARIABLEN-LISTE FUER "+FI$:PRINT:PRINTD$:OPEN4,4:PRINT#4,D$:
1010 FORI=0TOAZ%-1:D$=LEFT$(A$(I)+"      ",5)+" : <"              <PRINT#4
1015 FORG=0TO1:IFB$(I,G)=""GOTO1035
1020 A$=B$(I,G):L=LEN(A$)
1025 FORJ=1TOLSTEP2:D$=D$+STR$(ASC(MID$(A$,J,1))+256*ASC(MID$(A$,J+1,1)))
1030 IFLEN(D$)>70THENPRINTD$:PRINT#4,D$:D$="        "
1032 NEXT
1035 IFG<.5THEND$=D$+">"
1040 NEXT:PRINTD$:PRINT#4,D$:NEXT:CLOSE4
READY

 VARIABLEN-LISTE FUER CROSSREFERENZ

 A     : < 500 530> 502 505 530 540
 A$    : < 110 210 230 270 320 410 520 540 1020> 210 230 250 260 300 302
         305 306 310 320 330 400 410 500 510 520 530 540 600 1020 1025
```

```
A$()   : < 50 700 800> 50 610 630 700 1010
AZ%    : < 50 710> 605 700 710 1010
B$     : < 210 230 240> 210 240 250 260
B$()   : < 700 710 810> 50 700 810 1015 1020
D$     : < 1000 1010 1025 1030 1035> 1000 1025 1030 1035 1040
FI$    : < 100> 110 1000
G      : < 250 302 502 1015> 602 800 1015 1020 1035
H      : < 250 305 306> 602 800
I      : < 1010> 1010 1015 1020
J      : < 700 800 1025> 700 810 820 1025
L      : < 1020> 1025
V$     : < 510 530 540 600> 530 540 600 610 630 800 820
X      : < 605 630 650> 605 610 630 650 700 710 800 810
Y      : < 605 630> 630 650
ZN$    : < 250> 810                                        Ω
```

Meno Sellschopp, 2407 Bad Schwartau

# Zweidimensionale Felder sortieren (CBM)

Dieses Programm sortiert Strings im Arbeitsspeicher des CBM 30... Der Autor verdankt Grundzüge des Programms der Veröffentlichung von A. Quint in 65xx MICRO MAG, Heft 16.

Die Vorzüge dieser Version sind:

1. Das Programm sortiert zweidimensionale Felder. Als Sortierschlüssel kann eine beliebige Ebene der 2. Dimension gewählt werden. Die zugehörigen Variablen der anderen Ebenen werden mitsortiert.

2. Es sucht auch unter mehreren dimensionierten Feldern das richtige mit dem Variablennamen `X$(A,B)`.

3. Es ist frei verschiebbar, dazu brauchen keine Sprung- und Verzweigungsadressen verändert zu werden.

4. Das Initialprogramm (hex 0433-046F) verschiebt das Arbeitsprogramm selbst an das Ende des RAM und begrenzt den freien Platz im RAM neu.

5. Ebenfalls vom Initialprogramm wird die aktuelle Anfangsadresse in $01/02 geladen. Der Einsprung in die Sortierroutine erfolgt immer über SYS(0)S. Dabei ist S die Nummer der Ebene (2. Dimension), nach der sortiert werden soll.

Die freie Verschiebbarkeit des Programms wurde mit einem einfachen Trick erreicht, dem ich den Namen 'Relais-Springen' gebe und der hier besonders mitgeteilt werden soll:

Der BIT-Befehl, z.B. 2C 90 A6, hat keine Wirkung, sofern nicht sofort ein BRANCH Befehl folgt. Springe ich jedoch auf sein zweites Byte, dann wird daraus BCC $A6(relativ). Mit einem BCC angesprungen, leitet es den Sprung weiter. Von mehreren Relais können so relative Sprünge über große Distanzen geführt werden. Als Speicher werden die Plätze hex 83E9-83FF hinter dem Bildschirm genutzt, auch sie können also stehenbleiben.

```
0433  A9 70        LDA ##$70      Initialprogramm
0435  85 5C        STA $5C        verschiebt das Sortierprogramm
0437  A9 04        LDA ##$04      an das Ende des RAM
0439  85 5D        STA $5D
043B  A9 91        LDA ##$91
043D  85 57        STA $57
043F  A9 06        LDA ##$06
0441  85 58        STA $58
0443  A5 34        LDA $34
0445  85 55        STA $55
0447  A5 35        LDA $35
0449  85 56        STA $56
044B  20 DF C2     JSR $C2DF
044E  38           SEC
044F  A5 34        LDA $34        setzt Einsprungadresse in $01/02
0451  E9 21        SBC ##$21
0453  85 01        STA $01
0455  A5 35        LDA $35
0457  E9 02        SBC ##$02
0459  85 02        STA $02
045B  A9 4C        LDA ##$4C
045D  85 00        STA $00
045F  38           SEC
0460  A5 01        LDA $01        setzt Obergrenze des RAM neu
0462  E9 03        SBC ##$03
0464  85 34        STA $34
0466  A5 02        LDA $02
0468  E9 00        SBC ##$00
046A  85 35        STA $35
046C  EA           NOP
046D  4C 5D C5     JMP $C55D      NEW
0470  EA           NOP
0471  20 8B CC     JSR $CC8B      Sortierprogramm
0474  20 D2 D6     JSR $D6D2      holt Nr. der Sortierebene
0477  20 29 E2     JSR $E229      löscht Bildschirm
047A  18           CLC
047B  A5 11        LDA $11        speichert Nr. der Sortierebene
047D  8D E9 83     STA $83E9
0480  A5 12        LDA $12
0482  8D EA 83     STA $83EA
0485  EA           NOP
0486  A5 2C        LDA $2C        sucht richtigen Variablennamen
0488  85 54        STA $54
048A  A5 2D        LDA $2D
048C  85 55        STA $55
048E  A0 00        LDY ##$00
0490  B1 54        LDA ($54),Y
0492  C9 58        CMP ##$58
0494  D0 0F        BNE $04A5
0496  C8           INY            prüft auf richtige Dimension
0497  B1 54        LDA ($54),Y
0499  C9 80        CMP ##$80
049B  D0 08        BNE $04A5
049D  A0 04        LDY ##$04
049F  B1 54        LDA ($54),Y
04A1  C9 02        CMP ##$02
04A3  F0 20        BEQ $04C5
```

```
04A5  18        CLC
04A6  A0 02     LDY #$02
04A8  B1 54     LDA ($54),Y  sucht nächstes Feld, wenn
04AA  65 54     ADC $54      X$(A,B) nicht gefunden
04AC  AA        TAX
04AD  C8        INY
04AE  B1 54     LDA ($54),Y
04B0  65 55     ADC $55
04B2  85 55     STA $55

04B4  86 54     STX $54
04B6  A5 55     LDA $55
04B8  C5 2F     CMP $2F
04BA  90 D2     BCC $048E
04BC  A5 54     LDA $54
04BE  C5 2E     CMP $2E      prüft auf Ende der Array-Tabelle
04C0  90 CC     BCC $048E
04C2  4C 20 D1  JMP $D120    BAD SUBSRIPT, wenn Feld nicht
04C5  EA        NOP          vorhanden
04C6  A0 05     LDY #$05
04C8  A2 03     LDX #$03
04CA  B1 54     LDA ($54),Y

04CC  9D EB 83  STA $83EB,X
04CF  C8        INY
04D0  CA        DEX
04D1  10 F7     BPL $04CA
04D3  EA        NOP
04D4  AD EA 83  LDA $83EA
04D7  CD EE 83  CMP $83EE
04DA  90 0D     BCC $04E9
04DC  F0 03     BEQ $04E1    prüft, ob gewählte Sortierebene
04DE  4C 23 D1  JMP $D123    vorhanden
04E1  AD E9 83  LDA $83E9    ILLEGAL QUANTITY, wenn nicht vorhanden
04E4  CD ED 83  CMP $83ED
04E7  B0 F5     BCS $04DE
04E9  EA        NOP
04EA  18        CLC
04EB  A5 54     LDA $54      Zeiger auf Länge und Adresse des
04ED  69 09     ADC #$09     0-_lements der 0-Ebene
04EF  8D FA 83  STA $83FA
04F2  8D FC 83  STA $83FC
04F5  A5 55     LDA $55
04F7  69 00     ADC #$00
04F9  8D FB 83  STA $83FB
04FC  8D FD 83  STA $83FD
04FF  EA        NOP          Einsprung nach Erhöhung von A
0500  A2 01     LDX #$01
0502  BD EB 83  LDA $83EB,X  Zähler auf Zahl der Elemente und
0505  9D EF 83  STA $83EF,X  Ebenen bringen
0508  9D F3 83  STA $83F3,X
050B  BD ED 83  LDA $83ED,X
050E  9D F5 83  STA $83F5,X
0511  CA        DEX
0512  10 EE     BPL $0502
0514  EA        NOP
0515  A2 01     LDX #$01     X=A
0517  BD EB 83  LDA $83EB,X
051A  9D F7 83  STA $83F7,X
```

```
051D  CA          DEX
051E  10 F7       BPL $0517
0520  A2 01       LDX #$01
0522  BD FA 83    LDA $83FA,X Zeiger A/X auf 0-Ebene bringen
0525  9D FC 83    STA $83FC,X
0528  95 54       STA $54,X
052A  CA          DEX
052B  10 F5       BPL $0522
052D  A2 03       LDX #$03
052F  18          CLC
0530  A5 54       LDA $54     Zeiger A/X auf Sortierebene bringen
0532  6D E9 83    ADC $83E9
0535  85 54       STA $54
0537  A5 55       LDA $55
0539  6D EA 83    ADC $83EA
053C  85 55       STA $55
053E  CA          DEX
053F  D0 EE       BNE $052F
0541  CE F7 83    DEC $83F7
0544  D0 E7       BNE $052D
0546  CE F8 83    DEC $83F8
0549  10 E2       BPL $052D
054B  18          CLC
054C  AD FA 83    LDA $83FA   M = A+1 in der 0-Ebene
054F  69 03       ADC #$03
0551  8D FE 83    STA $83FE
0554  AD FB 83    LDA $83FB
0557  69 00       ADC #$00
0559  8D FF 83    STA $83FF
055C  18          CLC
055D  A5 54       LDA $54     M = A+1 in der Sortierebene
055F  69 03       ADC #$03
0561  85 56       STA $56
0563  A5 55       LDA $55
0565  69 00       ADC #$00
0567  85 57       STA $57
0569  38          SEC
056A  AD EF 83    LDA $83EF   Zähler M = (Zähler A)-1
056D  E9 01       SBC #$01
056F  8D F1 83    STA $83F1
0572  AD F0 83    LDA $83F0
0575  E9 00       SBC #$00
0577  8D F2 83    STA $83F2
057A  2C 90 97    BIT $9790   RELAIS
057D  EA          NOP
057E  A0 00       LDY #$00
0580  B1 54       LDA ($54),Y Vergleich der String-Längen
0582  F0 2F       BEQ $05B3
0584  8D F9 83    STA $83F9
0587  B1 56       LDA ($56),Y
0589  F0 28       BEQ $05B3
058B  CD F9 83    CMP $83F9
058E  B0 03       BCS $0593
0590  8D F9 83    STA $83F9
0593  A0 02       LDY #$02
0595  A2 01       LDX #$01
0597  B1 54       LDA ($54),Y
0599  95 58       STA $58,X   Zeiger auf die Strings stellen
```

```
059B  B1 56      LDA (*56),Y
059D  95 5A      STA *5A,X
059F  88         DEY
05A0  CA         DEX
05A1  10 F4      BPL *0597
05A3  A0 00      LDY #*00
05A5  B1 5A      LDA (*5A),Y Strings byteweise vergleichen
05A7  D1 58      CMP (*58),Y
05A9  90 10      BCC *05BB
05AB  D0 29      BNE *05D6
05AD  C8         INY
05AE  CE F9 83   DEC *83F9
05B1  D0 F2      BNE *05A5
05B3  A0 00      LDY #*00    wenn bis dahin gleich, dann kürzeres
05B5  B1 56      LDA (*56),Y Element nach vorn
05B7  D1 54      CMP (*54),Y
05B9  B0 1B      BCS *05D6
05BB  A2 01      LDX #*01
05BD  B5 56      LDA *56,X   X=M
05BF  95 54      STA *54,X
05C1  BD FE 83   LDA *83FE,X
05C4  9D FC 83   STA *83FC,X
05C7  CA         DEX
05C8  10 F3      BPL *05BD
05CA  2C D0 B0   BIT *B0D0   RELAIS
05CD  2C F0 55   BIT *55F0
05D0  2C 90 A8   BIT *A890
05D3  2C 10 A7   BIT *A710
05D6  EA         NOP
05D7  18         CLC
05D8  A5 56      LDA *56     M erhöhen
05DA  69 03      ADC #*03
05DC  85 56      STA *56
05DE  A5 57      LDA *57
05E0  69 00      ADC #*00
05E2  85 57      STA *57
05E4  18         CLC
05E5  AD FE 83   LDA *83FE
05E8  69 03      ADC #*03
05EA  8D FE 83   STA *83FE
05ED  AD FF 83   LDA *83FF
05F0  69 00      ADC #*00
05F2  8D FF 83   STA *83FF
05F5  CE F1 83   DEC *83F1   Zähler für M stellen
05F8  D0 D1      BNE *05CB
05FA  CE F2 83   DEC *83F2
05FD  10 D5      BPL *05D4
05FF  EA         NOP
0600  A2 01      LDX #*01
0602  BD ED 83   LDA *83ED,X Zähler zum Tauschen vorbereiten
0605  9D F5 83   STA *83F5,X
0608  BD FA 83   LDA *83FA,X
060B  95 54      STA *54,X   Zähler auf Längen und Adressen der
060D  BD FC 83   LDA *83FC,X Tauschelemente in der 0-Ebene bringen
0610  95 56      STA *56,X
0612  CA         DEX
0613  10 ED      BPL *0602
0615  A0 02      LDY #*02
```

```
0617  B1 54        LDA (*54),Y Längen und Adressen tauschen
0619  AA           TAX
061A  B1 56        LDA (*56),Y
061C  91 54        STA (*54),Y
061E  8A           TXA
061F  91 56        STA (*56),Y
0621  88           DEY
0622  10 F3        BPL *0617
0624  EA           NOP
0625  2C F0 30     BIT *30F0   RELAIS
0628  2C 90 A6     BIT *A690
062B  A2 03        LDX #*03
062D  18           CLC
062E  A5 54        LDA *54
0630  6D EB 83     ADC *83EB   Zeiger auf Längen und Adressen durch
0633  85 54        STA *54     die Ebenen hindurch erhöhen
0635  A5 55        LDA *55
0637  6D EC 83     ADC *83EC
063A  85 55        STA *55
063C  18           CLC
063D  A5 56        LDA *56
063F  6D EB 83     ADC *83EB
0642  85 56        STA *56
0644  A5 57        LDA *57
0646  6D EC 83     ADC *83EC
0649  85 57        STA *57
064B  CA           DEX
064C  D0 DF        BNE *062D
064E  CE F5 83     DEC *83F5
0651  D0 C2        BNE *0615
0653  CE F6 83     DEC *83F6
0656  10 BD        BPL *0615
0658  EA           NOP
0659  18           CLC
065A  AD FA 83     LDA *83FA   wenn Vertauschung beendet,
065D  69 03        ADC #*03    dann A erhöhen
065F  8D FA 83     STA *83FA
0662  AD FB 83     LDA *83FB
0665  69 00        ADC #*00
0667  8D FB 83     STA *83FB
066A  CE EF 83     DEC *83EF   Zähler für A stellen
066D  AD EF 83     LDA *83EF
0670  C9 01        CMP #*01    prüfen, ob Sortiergang beendet
0672  D0 06        BNE *067A
0674  CE F0 83     DEC *83F0
0677  10 01        BPL *067A
0679  60           RTS
067A  EA           NOP
067B  20 57 E2     JSR *E257   Cursor home
067E  AE EF 83     LDX *83EF
0681  AD F0 83     LDA *83F0
0684  20 D9 DC     JSR *DCD9   Sortierzähler A auf Bildschirm
0687  A9 20        LDA #*20
0689  20 EA E6     JSR *E6EA   Space ausgeben
068C  18           CLC
068D  90 9A        BCC *0629   weiter im Programm
068F  00           BRK
0690  00           BRK
```

```
10 PRINT"  T  E  S  T  P  R  O  G  R  A  M  M   "
20 PRINT"  ERZEUGT UND SORTIERT ZIFFERNFOLGEN":GOSUB200:PRINT
25 A=INT(RND(TI)*99)+5:B=INT(RND(TI)*5)+2
30 PRINT"1.DIMENSION (ZAHL DER ELEMENTE)  A = ";A
40 PRINT"2.DIMENSION (ZAHL DER EBENEN)    B = ";B
50 GOSUB200:PRINT:INPUT"#  SORTIERSCHLUESSEL  S    IN   B =";S
110 DIMA$(A):DIMX$(A,B)
120 PRINT"":FORI=0TOA:FORJ=1TOB:X$(I,J)=STR$(INT(RND(TI)*10))
130 X$(I,0)=X$(I,0)+RIGHT$(X$(I,J),1):NEXT:A$(I)=X$(I,0)
140 FORJ=0TOB:PRINTX$(I,J);:NEXT:PRINT:NEXT
150 T=TI:SYS(0)S:T=INT((TI-T)/6)/10
155 PRINT"A$(A)","X$(A,B)"
160 FORI=0TOA:PRINTA$(I),:FORJ=0TOB:PRINTX$(I,J)," ";:NEXT:PRINT:NEXT
166 GOSUB200:PRINT," 0";SPC(B-1);:FORJ=1TOB:PRINTJ;:NEXT:PRINT
170 PRINT,"X$(A,B) SORTIERT NACH:";S:PRINT"A$(A) UNSORTIERT":GOSUB200
180 PRINT"JE";A+1;"ELEMENTE IN";B+1;"EBENEN":PRINT"SORTIERT IN";T;
190 GOSUB200:END                                          "SEKUNDEN"
200 FORI=0TO38:PRINT".";:NEXT:PRINT:RETURN
```

Ω

St.Dir. Peter Rix, 2350 Neumünster

# EPROM-Progrogrammiereinheit für AIM 65

Die zu den ROM-Sockeln der AIM-Platine steckerkompatiblen 5 V-EPROMs 2516/2716 (2k-Byte) und 2532 (4k-Byte) sind heute preiswert erhältlich, so daß der Gedanke, eigene Software im EPROM zu speichern oder Änderungen am Betriebssystem selber vorzunehmen, realistisch geworden ist. Kommerzielle Programmiergeräte sind allerdings noch sehr teuer, und veröffentlichte Selbstbaulösungen unterliegen gewöhnlich erheblichen Betriebseinschränkungen.

Nachfolgend werden Hard- und Software für eine Programmiereinheit beschrieben, die sich im praktischen Einsatz als problemlos und komfortabel erwiesen hat, mit der einwandfrei die Typen 2516/2716/2532 programmiert werden können und deren Materialkosten unter DM 100,- liegen.

Die Hardware (siehe Bild) ist auf einer Europaplatine aufgebaut und wird direkt am Anwenderstekker der AIM-Platine betrieben. Teuerstes Bauelement ist der der 24-polige Nullkraftsockel für das EPROM. Die Spannungsversorgung erfolgt von der Rechnerplatine, als Brennspannung dient die Druckerspannung. Bei einigermaßen konstanter Druckerspannung (24-26 V) kann auf die angegebene Spannungsstabilisierung noch verzichtet werden.

Adreß- und Datentransfer erfolgen im Zeitmultiplexbetrieb über Port A des Benutzer-VIA in der Reihenfolge Adresse-Low, Adresse-High, Daten. Zwei 6-Bit-Speicher (74LS174 oder 40174) nehmen die Adresse auf. Alle Steuersignale (siehe Graphik) werden über Port B des Benutzer-VIA übertragen. Ein Oder-Gatter ist die ganze Hardware für die Anpassung an unterschiedliche Sockelbelegungen bei 2k- und 4k-EPROMs (z.B. 74LS00, 3xNAND).

Die Software ist dialogorientiert, sie gestattet sowohl das Einbrennen und Nachbrennen aus beliebigen Memory-Adressen in beliebige EPROM-Adressen, als auch das Einlesen aus beliebigen EPROM-Adressen in beliebige Memory-Adressen. Freizügige Wahl der angesprochenen Adressen erlaubt Brennprozesse von einzelnen Bytes bis zu vollständigen EPROMs.

Nach dem Programmstart müssen EPROM-Typ (2 oder 4k), Memory-Adreßbereich und EPROM-Anfangsadresse vorgegeben werden. Unzulässige Werte werden zurückgewiesen und führen zu einer Fehlermeldung. Anschließend steht die Menue-Auswahl -A-L-P-T-V-? zur Verfügung:

-A- Vorgabe neuer Adressen
Bis zu einer Änderung bleiben die vorhergehenden Adressen gespeichert und können z.B. beim Brennen mehrerer gleicher EPROMs weiter benutzt werden.

-B- Lesen EPROM in Memory
Editierung von EPROM-Programmen wird erleichtert. EPROM kann als externer Speicher für Programme benutzt werden, die im RAM ablaufen sollen.

-P- Programmieren EPROM
Vor dem Kommando Brennspannung einschalten!
Am Ende des Brennvorganges Brennspannung ausschalten!
EPROM bei eingeschalteter Brennspannung nicht einsetzen oder entnehmen! 4k-EPROM bei eingeschalteter Brennspannung nicht mit Kommando -L-T-V betreiben!

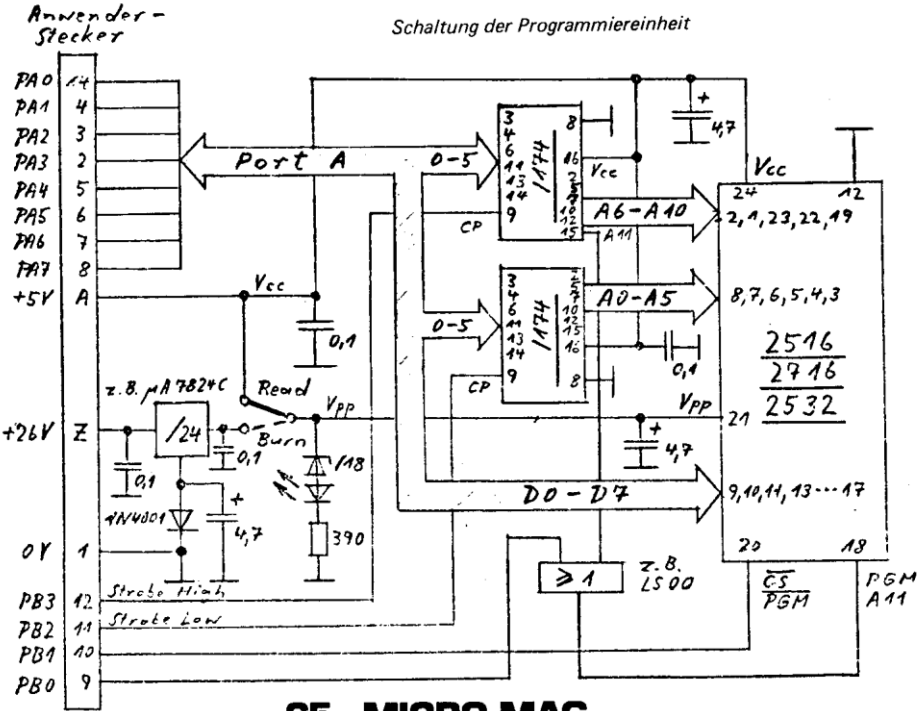
Wird beim Brennen eine Page-Grenze erreicht, so wird dies durch Ausgabe eines P Angezeigt. Zu diesem Zeitpunkt kann der Brennvorgang mit der ESC-Taste abgebrochen werden (ca. alle 13 Sekunden).

-T- Testen auf leeres EPROM
Der Test sollte jedem Programmiervorgang vorausgehen. Der vorgegebene Memory-Adreßbebereich dient hier nur der Festlegung des zu testendem EPROM-Bereiches, sein Inhalt kann dabei beliebig sein. Abweichungen vom Bitmuster 'FF' werden mit Adresse ausgegeben, danach kann mit beliebigem Menue-Kommando fortgesetzt werden (z.B. -T- für weiteres Testen oder -X- für Menue-Neustart).

-V- Vergleichen Memory- mit EPROM-Inhalt
Der Vergleich sollte nach jedem Programmiervorgang erfolgen. Abweichungen werden mit Adressen und Bitmustern ausgegeben, danach kann mit beliebigem Menue-Kommando fortgesetzt werden (z.B. -V- für weiteres Vergleichen oder -X- für Menue-Neustart).

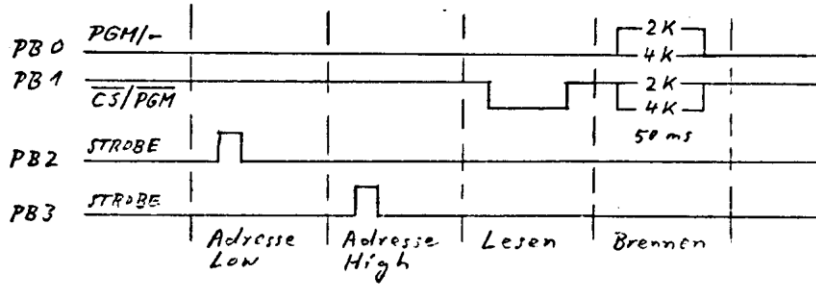
Die ESC-Taste führt zum Monitor zurück, jede sonstige Taste setzt bei Anzeige "?" die Adressen auf die vorgegebenen Anfangswerte zurück und startet das Menue neu.

*Schaltung der Programmiereinheit*



Folgende Hinweise aus dem Begleitschreiben des Autors: Beim AIM 65 muß evtl. noch die Druckerspannung an den Pin Z (Applikation) angeschlossen werden, beim PC 100 liegt sie dort bereits an.

Zum Betrieb der 2k-EPROMs sollte man wissen, daß Pin 18 der Chip-Select-Anschluß und Pin 20 der Output-Enable-Anschluß ist. Legt man Pin 18, wie häufig beobachtbar, fest auf Low-Potential, so verzichtet man ohne Not auf die Low-Power-Betriebsart. Zweitbeste Lösung: $\overline{CS}$ parallel an Pin 18 und an Pin 20 sichert Low-Power-Betrieb. Beste Lösung: $\overline{CS}$ an Pin 18, 02 (Phi2) an Pin 20 liefert saubere Busimpulse, macht aber den Sockel für 4k-ROMs unbrauchbar.




*Von Port B ausgehende Steuersignale per Programm*

```
0000            ;PROM-BRENNPROGRAMM
0000            ;FUER 2516/2716/2532
0000            ;15/05/81
0000            ;P.RIX/2350 NMS          ;------------------
0000
0000 START              =$D00            Startadresse nach Bedarf einrichten
0000                                     ;PROGRAMMVARIABLE
0000                    *=$40
0040 MRLO               *=*+2            Memory Low
0042 PRLO               *=*+2            PROM Low
0044 MRHI               *=*+2            Memory High
0046 MRADR              *=*+2            Adreßvariable Memory
0048 PRADR              *=*+2            Adreßvariable PROM
004A VAR1               *=*+1            Arbeitsvariable 1
004B VAR2               *=*+1            Arbeitsvariable 2
004C TYP                                 Typ-Variable (2k/4k)
004C
004C UDRB               =$A000
004C UDRA               =$A001
004C UDDRB              =$A002
004C UDDRA              =$A003
004C CNTH30             =$A417
004C ADDR               =$A41C
004C                                     ;MONITORPROGRAMME
004C
004C BLANK              =$E83E
004C CRLF               =$E9F0
004C DELAY              =$EC0F
004C FROM               =$E7A3
004C TO                 =$E7A7
004C NUMA               =$EA46
004C WRAX               =$EA42
```

```
004C OUTALL          =$E9BC
004C RCHEK           =$E907
004C PSL1            =$E837
004C QM              =$E7D4
004C EQUAL           =$E7D8          ;------------------
004C
004C                 *=START
0D00 START   20F0E9  JSR CRLF
0D03         A007    LDY #TXT2-TXT1
0D05         20320F  JSR TEXT        Abfrage nach ROM-Typ
0D08         20F0E9  JSR CRLF        2k oder 4k
0D0B         A00E    LDY #TXT3-TXT1
0D0D         20320F  JSR TEXT
0D10 ROMTYP  2007E9  JSR RCHEK
0D13         C9FF    CMP #$FF
0D15         F0F9    BEQ ROMTYP
0D17         48      PHA
0D18         20BCE9  JSR OUTALL
0D1B         68      PLA
0D1C         C932    CMP #'2         Typvariable setzen
0D1E         D004    BNE K4          8 bei 2k-Typ
0D20         A908    LDA #8          16 bei 4k-Typ
0D22         D006    BNE TVAR
0D24 K4      C934    CMP #'4
0D26         D0D8    BNE START       zurück zum Start
0D28         A910    LDA #16         falls nicht 2k oder 4k
0D2A TVAR    854C    STA TYP         ;------------------
0D2C
0D2C MEMAD   20F0E9  JSR CRLF
0D2F         A000    LDY #0
0D31         20320F  JSR TEXT
0D34         20A3E7  JSR FROM        Memory-Adreßbereich
0D37         B0F6    BCS MEMAD+3     mit FROM= ... TO= ...
0D39         AD1CA4  LDA ADDR        übernehmen
0D3C         8540    STA MRLO
0D3E         AD1DA4  LDA ADDR+1
0D41         8541    STA MRLO+1
0D43         203EE8  JSR BLANK
0D46         20A7E7  JSR TO
0D49         B0E4    BCS MEMAD+3
0D4B         AD1DA4  LDA ADDR+1      Gültigkeitsprüfung für Adreßbereich
0D4E         8545    STA MRHI+1      ≤ 8 Pages für 2k
0D50         AD1CA4  LDA ADDR        ≤16 Pages für 4k
0D53         8544    STA MRHI
0D55         38      SEC
0D56         E540    SBC MRLO
0D58         A545    LDA MRHI+1
0D5A         E541    SBC MRLO+1
0D5C         C54C    CMP TYP
0D5E         900A    BCC START1      zur Eingabe
0D60         20F0E9  JSR CRLF        PROM Anfangsadresse
0D63         A01F    LDY #TXT4-TXT1
0D65         20320F  JSR TEXT        Wiederholung bei Adreßfehler
0D68         30C2    BMI MEMAD
0D6A
0D6A START1  20F0E9  JSR CRLF        PROM-Anfangsadresse mit
0D6D         A007    LDY #TXT2-TXT1  FROM= ... übernehmen
0D6F         20320F  JSR TEXT
0D72         20A3E7  JSR FROM
```

```
0D75           B0F6    BCS START1+3
0D77           AD1CA4  LDA ADDR
0D7A           8542    STA PRLO
0D7C           AD1DA4  LDA ADDR+1
0D7F           8543    STA PRLO+1
0D81           C54C    CMP TYP          Gültigkeitsprüfung
0D83           900A    BCC INIT
0D85           20F0E9  JSR CRLF
0D88           A01F    LDY #TXT4-TXT1   Wiederholung bei Adreßfehler
0D8A           20320F  JSR TEXT
0D8D           30DB    BMI START1       ;------------------
0D8F
0D8F INIT      A9FF    LDA #$FF         VIA 6522
0D91           8D02A0  STA UDDRB        Port A und Port B initialisieren
0D94           20900E  JSR DISABL
0D97           A9C3    LDA #$C3         Monitor-DELAY-Routine
0D99           8D17A4  STA CNTH30       auf T=50 ms einstellen
0D9C           A950    LDA #$50
0D9E           8D18A4  STA CNTH30+1
0DA1           A203    LDX #3
0DA3 INIT1     B540    LDA MRLO,X       Adreßvariable MRADR/PRADR
0DA5           9546    STA MRADR,X      mit Anfangsadressen laden
0DA7           CA      DEX
0DA8           10F9    BPL INIT1        ;------------------
0DAA
0DAA MENUE     20F0E9  JSR CRLF
0DAD           A02C    LDY #TXT5-TXT1
0DAF           20320F  JSR TEXT         Betriebsart aus Menue
0DB2 MEN1      2007E9  JSR RCHEK        -A-L-P-T-V-?  wählen
0DB5           C9FF    CMP #$FF
0DB7           F0F9    BEQ MEN1
0DB9           48      PHA
0DBA           20BCE9  JSR OUTALL
0DBD           20F0E9  JSR CRLF
0DC0           68      PLA
0DC1           C941    CMP #'A          neue Adressen?
0DC3           D003    BNE MEN2
0DC5           4C2C0D  JMP MEMAD
0DC8 MEN2      C94C    CMP #'L          Lesen PROM in Memory
0DCA           F058    BEQ LESEN
0DCC           C954    CMP #'T          Testen auf leeres PROM?
0DCE           F033    BEQ TEST
0DD0           C956    CMP #'V          Vergleichen Memory/PROM
0DD2           F009    BEQ VERGL
0DD4           C950    CMP #'P          Programmieren PROM?
0DD6           D0B7    BNE INIT
0DD8           F078    BEQ PROGR        ;------------------
0DDA
0DDA VGLNXT    20D90E  JSR NXTADR
0DDD VERGL     A000    LDY #0           Vergleich Memory (VAR1)
0DDF           B146    LDA (MRADR)Y     mit PROM (VAR2)
0DE1           854A    STA VAR1
0DE3           20960E  JSR ADRESS       zur Adreßprüfung
0DE6           3005    BMI VGLEND
0DE8           20E60E  JSR GLEICH       zum Lesen PROM und Vergleich
0DEB           F0ED    BEQ VGLNXT
0DED VGLEND    20F0E9  JSR CRLF         erfolgreicher Vergleich
0DF0           A000    LDY #0           Schlußtext
```

```
0DF2            20320F JSR TEXT         MEMORY=PROM
0DF5            20D8E7 JSR EQUAL
0DF8            A007   LDY #TXT2-TXT1
0DFA            20320F JSR TEXT         zum Menue-Start
0DFD            4C8F0D JMP INIT         |------------------
0E00
0E00 TSTNXT     20D90E JSR NXTADR
0E03 TEST       A9FF   LDA #$FF          Testen PROM (VAR2)
0E05            854A   STA VAR1         auf Inhalt FF (VAR1)
0E07            20960E JSR ADRESS
0E0A            3005   BMI TSTEND
0E0C            20E60E JSR GLEICH
0E0F            F0EF   BEQ TSTNXT
0E11 TSTEND     20F0E9 JSR CRLF          erfolgreicher Test
0E14            A007   LDY #TXT2-TXT1   Schlußtext
0E16            20320F JSR TEXT         PROM LEER!
0E19            A039   LDY #TXT6-TXT1
0E1B            20320F JSR TEXT
0E1E            4C8F0D JMP INIT         zum Menue-Start
0E21                                    |------------------
0E21
0E21 LSNXT      20D90E JSR NXTADR
0E24 LESEN      20960E JSR ADRESS        Lesen PROM in Memory
0E27            300A   BMI LSEND
0E29            20850E JSR READ
0E2C            A000   LDY #0
0E2E            9146   STA (MRADR)Y
0E30            4C210E JMP LSNXT
0E33 LSEND      20F0E9 JSR CRLF          Lesen beendet
0E36            A000   LDY #0           Schlußtext:
0E38            20320F JSR TEXT         MEMORY GELADEN!
0E3B            A03E   LDY #TXT7-TXT1
0E3D            20320F JSR TEXT         zum Menue-Start
0E40            4C8F0D JMP INIT
0E43                                    |------------------
0E43
0E43 PRGNXT     20D90E JSR NXTADR
0E46            A548   LDA PRADR
0E48            D008   BNE PROGR
0E4A            A950   LDA #'P         Ausgabe P und
0E4C            20BCE9 JSR OUTALL       Prüfen auf Abbruch (ESC)
0E4F            2007E9 JSR RCHEK        am Ende jeder Page
0E52 PROGR      20960E JSR ADRESS        Programmieren PROM
0E55            301E   BMI PRGEND
0E57            A000   LDY #0
0E59            B146   LDA (MRADR)Y     Bitmuster aus Memory
0E5B            8D01A0 STA UDRA         über Port A ausgeben
0E5E            A903   LDA #3           PGM-Impuls
0E60            A64C   LDX TYP          PB0 High für 2k
0E62            E008   CPX #8           PB1 Low für 4k
0E64            F002   BEQ *+4
0E66            A900   LDA #0
0E68            8D00A0 STA UDRB         PGM-Start
0E6B            200FEC JSR DELAY        50 ms Verzögerung
0E6E            A902   LDA #2
0E70            8D00A0 STA UDRB         PGM-Stopp
0E73            D0CE   BNE PRGNXT
0E75 PRGEND     20F0E9 JSR CRLF          Programmieren beendet
```

```
0E78            A007    LDY #TXT2-TXT1  Schlußtext:
0E7A            20320F  JSR TEXT        PROM GELADEN!
0E7D            A03E    LDY #TXT7-TXT1
0E7F            20320F  JSR TEXT
0E82            4C8F0D  JMP INIT        zum Menue-Start
0E85                                    !------------------
0E85                                    !UNTERPROGRAMME
0E85
0E85 READ       A200    LDX #0          Port A-Eingänge
0E87            8E03A0  STX UDDRA       CS aktiv Low
0E8A            8E00A0  STX UDRB        PROM-Daten lesen
0E8D            AD01A0  LDA UDRA        CS High
0E90 DISABL     A202    LDX #2          PROM inaktiv
0E92            8E00A0  STX UDRB
0E95            60      RTS
0E96
0E96 ADRESS     A547    LDA MRADR+1     Prüfen Memory-Adresse
0E98            C545    CMP MRHI+1
0E9A            900B    BCC ADRES1
0E9C            D006    BNE ENDADR      auf überschrittene
0E9E            A544    LDA MRHI        Endadresse
0EA0            C546    CMP MRADR
0EA2            B003    BCS ADRES1
0EA4 ENDADR     A9FF    LDA #$FF        Endadresse
0EA6            60      RTS             zurück mit Statusbit N=1
0EA7 ADRES1     A549    LDA PRADR+1     Prüfen PROM-Adresse auf
0EA9            C54C    CMP TYP         überschrittene Endadresse
0EAB            B0F7    BCS ENDADR
0EAD            A9FF    LDA #$FF
0EAF            8D03A0  STA UDDRA
0EB2            A548    LDA PRADR       Ausgabe PROM-Adresse Low
0EB4            854B    STA VAR2        Bit 0...5 über Port A
0EB6            8D01A0  STA UDRA
0EB9            A202    LDX #2
0EBB            A906    LDA #6
0EBD            8D00A0  STA UDRB        Strobe-Impuls
0EC0            8E00A0  STX UDRB        für Adreßlatch Low
0EC3            A549    LDA PRADR+1
0EC5            264B    ROL VAR2        Zusammenfassen PROM-Adresse Hogh
0EC7            2A      ROL A           Bit 6...11
0EC8            264B    ROL VAR2        und
0ECA            2A      ROL A           Ausgabe über Port A
0ECB            8D01A0  STA UDRA
0ECE            A90A    LDA #$A
0ED0            8D00A0  STA UDRB        Strobe-Impuls
0ED3            8E00A0  STX UDRB        für Adreßlatch High
0ED6            A900    LDA #0
0ED8            60      RTS
0ED9
0ED9 NXTADR     E648    INC PRADR       PROM-Adresse erhöhen
0EDB            D002    BNE NXTMA
0EDD            E649    INC PRADR+1
0EDF NXTMA      E646    INC MRADR       Memory-Adresse erhöhen
0EE1            D002    BNE NXTEND
0EE3            E647    INC MRADR+1
0EE5 NXTEND     60      RTS
0EE6
```

```
0EE6 GLEICH   20850E JSR READ          zum Lesen PROM
0EE9          854B   STA VAR2          Vergleich PROM/Memory
0EEB          C54A   CMP VAR1
0EED          D001   BNE UNGL
0EEF          60     RTS
0EF0 UNGL     68     PLA               Memory ungleich PROM
0EF1          68     PLA
0EF2          20F0E9 JSR CRLF
0EF5          A000   LDY #0
0EF7          20320F JSR TEXT          Druck der unterschiedlichen Inhalte
0EFA          2037E8 JSR PSL1          von Memory und PROM
0EFD          A007   LDY #TXT2-TXT1
0EFF          20320F JSR TEXT
0F02          20F0E9 JSR CRLF
0F05          A547   LDA MRADR+1
0F07          A646   LDX MRADR
0F09          2042EA JSR WRAX
0F0C          203EE8 JSR BLANK
0F0F          A54A   LDA VAR1
0F11          2046EA JSR NUMA
0F14          2037E8 JSR PSL1
0F17          A54B   LDA VAR2
0F19          2046EA JSR NUMA
0F1C          203EE8 JSR BLANK
0F1F          A549   LDA PRADR+1
0F21          A648   LDX PRADR
0F23          2042EA JSR WRAX
0F26          203EE8 JSR BLANK
0F29          20D90E JSR NXTADR
0F2C          20D4E7 JSR QM            Fortsetzung Menue
0F2F          4CB20D JMP MEN1          mit Folgeadresse
0F32
0F32 TEXT     B9400F LDA TXT1,Y        Text-Druckroutine
0F35          48     PHA
0F36          297F   AND #$7F
0F38          20BCE9 JSR OUTALL
0F3B          C8     INY
0F3C          68     PLA
0F3D          10F3   BPL TEXT
0F3F          60     RTS
0F40
0F40 TXT1     4D45   .BYT 'MEMORY',$A0
0F46          A0
0F47 TXT2     5052   .BYT 'PROM  ',$A0
0F4D          A0
0F4E TXT3     3235   .BYT '2516(2)/2532(4)?',$A0
0F5E          A0
0F5F TXT4     4144   .BYT 'ADRESSFEHLER',$A1
0F6B          A1
0F6C TXT5     2D41   .BYT '-A-L-P-T-V-?',$A0
0F78          A0
0F79 TXT6     4C45   .BYT 'LEER',$A1
0F7D          A1
0F7E TXT7     4745   .BYT 'GELADEN',$A1
0F85          A1
0F86
0F86                 .END BURN3
0F86          ERRORS= 000
```

Ω

*Andreas Spiegelberg, Dipl.-Ing., 1000 Berlin 19*
*Steve D. Nielsen, B.S., Salt Lake City, UT 84112*

# Softwareentwicklung für AIM 65 auf PDP 11

*Problem:*

*Im Bereich der medizinischen Technik sollte ein preiswertes intelligentes Meßgerät entwickelt werden. In Echtzeit sollte nach einer bekannten Gleichung aus einem Druck- und einem Volumensignal das Herzzeitvolumen beim künstlichen Herzen berechnet werden (1). Das Drucksignal war vorhanden, für das Volumensignal war ein Anemometer zu entwickeln. Da keine großen Stückzahlen zu erwarten waren und da für die Entwicklung nur eine begrenzte Zeit zur Verfügung stand, erschien eine Lösung, die auf einem Einkartenrechner aufbaut, angemessen. Wegen der guten Testmöglichkeiten, die durch seinen Monitor gegeben sind, wurde der AIM 65 ausgewählt. Er erlaubt es, Assemblerprogramme mit Verfahren, die dem Stand der Technik entsprechen, zu testen und in Betrieb zu nehmen. Für die Programmierung von EPROMs mußte ein Gerät angeschafft werden. Die Entscheidung fiel zugunsten des DRAM der Fa. Computerist. Es sollte zugleich als Speichererweiterung während der Testphase eingesetzt werden. Zur A/D- und D/A-Wandlung wurde ein ADC1 der Fa. Optimal Technology ausgewählt.*

*Eine PDP 11-34 unter RSX 11/M stand zur Verfügung. Sie ist mit vier RK 05 Plattenlaufwerken (je 2.4 Mbyte), zwei Bildschirmterminals und einem Matrixdrucker ausgerüstet. Ein Crossassembler für den 6502 war vorhanden. Dabei handelt es sich um einen makrofähigen Assembler, dessen Objektformat mit dem Objektformat für Lochstreifen des AIM kompatibel ist. Zur Kommunikation zwischen der PDP 11 und dem AIM stand eine serielle Schnittstelle mit 20 mA Stromschleife zur Verfügung. Der Crossassembler kann einzelne Quellfiles assemblieren. Es besteht keine Möglichkeit, mehrere Objektfiles zu linken.*

*Wegen des Umfanges der zu entwickelnden Programme war es unmöglich, die gesamte Entwicklung nur auf dem Zielsystem durchzuführen. Das Entwicklungssystem, das aus den vorhandenen Elementen zu konfigurieren war, sollte folgende Anforderungen erfüllen:*

- *Mehrere Programmmodule müssen zu einem Paket gelinkt werden können.*
- *Für das Editieren und Assemblieren der Programme soll der Plattenspeicher der PDP 11 genutzt werden.*
- *Die Programme, die später im EPROM laufen sollen, sollen im RAM des AIM getestet werden können.*

*Methode:*

*Es wurde eine Konfiguration entwickelt, in der die Quellprogramme auf der PDP 11 editiert und assembliert werden. Der Objektkode wird über die serielle Schnittstelle ins RAM geladen. Nach Test und Modifikation werden die Programme in EPROMs gebrannt. Bild 1 zeigt die Konfiguration.*

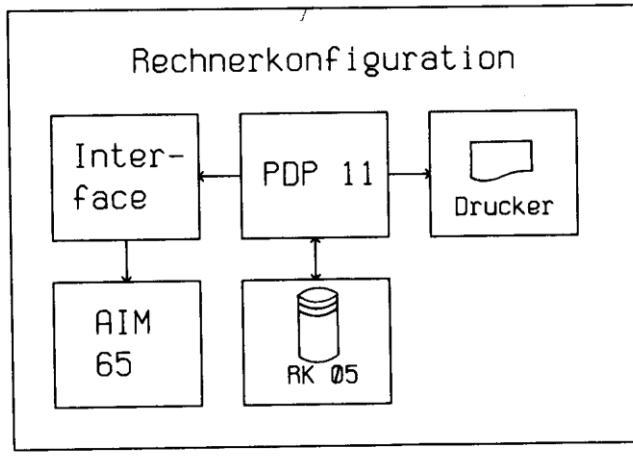
*Folgende Modifikationen und Entwicklungen mußten vorgenommen werden:*

*Für die Entwicklung eines größeren Programmpaketes ist es unbedingt erforderlich, die Programme in einzelne Module aufzuteilen, die der Reihe nach getestet und in Betrieb genommen werden. Geändert werden muß dann nur das jeweils bearbeitete Modul, die anderen bleiben, wie sie sind. Es wurde ein Crosslinker erstellt, der zugleich die Benutzerschnittstelle für den Assembler übernimmt. Er ist in der Indirect Command File Sprache des RSX 11/M Betriebssystemes formuliert. Diese Sprache enthält als Elemente alle Systemkommandos des Betriebssystemes, zusätzlich sind arithmetische und logische Funktionen möglich. Listing 1 zeigt den Crosslinker. Der Reihe nach werden folgende Funktionen ausgeführt:*

*Es werden die Namen für bis zu fünf Quellfiles, die gelinkt werden sollen, eingelesen. Der Name des Ausgangsfiles wird eingelesen. Es wird gefragt, ob ein Listing gewünscht wird. Die Inhalte der*

*Quellfiles werden zusammen in ein temporäres Quellfile mit dem Namen INTFIL.ASM geschrieben. Dieses wird später das Eingangsfile für den Crossassembler. Der Crossassembler erwartet als Eingabe vom Benutzerterminal die Namen von Eingangsfile, Listfile, Objektfile, Arbeitsfile, die Länge des Arbeitsfiles, den Namen eines Makrofiles und dessen Länge. All diese Angaben werden vom Linker mit .DATA-Anweisungen in ein File mit dem Namen FOR006.DAT geschrieben. Mit der REA-Anweisung wird dem Kanal 6 des Crossassemblers dieses File, das auf der Systemplatte SY: liegt, zugewiesen. Der Crossassembler liest nachdem er gestartet wurde, die notwendigen Daten aus diesem File und nicht vom Terminal. Der Crossassembler übersetzt das Eingangsfile und legt ein Objektfile sowie ein Listfile an. Abschließend werden die nicht mehr gebrauchten temporären Files gelöscht und, falls gewünscht, das Listing gedruckt. Dies erfolgt mit der Routine PRI, die die Ausgabe von Blanks, mit denen alle Zeilen ausgefüllt sind, unterdrückt.*

*Der AIM ist mit einer 20 mA Stromschleifen-Schnittstelle ausgerüstet. Diese ist für den Anschluß eines Teletypes ausgelegt. Das serielle Interface des AIM ist so beschaffen, daß zwischen den Kontakten R und T lediglich ein Schalter geöffnet und geschlossen werden muß. Das serielle Interface der PDP 11 ist ebenfalls für den Anschluß eines Teletypes ausgelegt, ihre beiden Sendeleitungen werden also nicht nur geschaltet, sondern sie liefern aktiv den Strom. Bild 2 zeigt die Schaltung für ein Interface, das diese Probleme beseitigt. Der Optokoppler dient der Potentialtrennung, er sorgt für die erforderliche Flankensteilheit.*



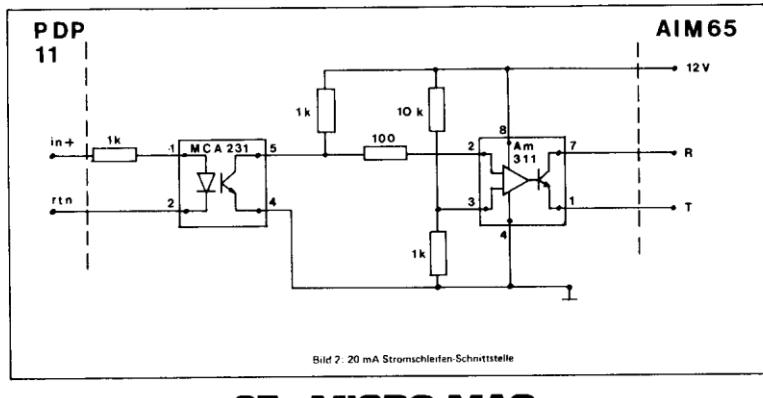




Bild 2. 20 mA Stromschleifen-Schnittstelle

*Der Lader des AIM benötigt zwischen den einzelnen empfangenen Zeichen jeweils eine kleine 'Denkpause', um sie zu verarbeiten. Das steht nicht im Handbuch. Diese Bedingung wird von einem Teletype-Lochstreifenleser sicher erfüllt. Wegen dieser erforderlichen Pause ist es unmöglich, die Objektfiles mit dem Peripheral Interchange Program (PIP) der PDP 11 von der Platte auf die serielle Leitung zu schicken. Die einzelnen Zeichen würden zu schnell aufeinander folgen. Um dieses Problem zu lösen, wurde ein Sendeprogramm erstellt, das die zZeichen eines Objektfiles zeilenweise von der Platte liest und sie mit Hilfe des Systemprogrammes WTQIO zeichenweise auf die serielle Leitung schickt. Wichtig ist der Funktionskode für die WTQIO-Routine: Er vereint die Funktionen:*

- *write logical block*
- *pass all bits.*

*Mit 'pass all bits' wird erreicht, daß alle Zeichen so, wie sie im File stehen, ohne Interpretation durch den Terminaldriver übertragen werden. Achtung: Der Zahlenwert (oktal 410) für den Funktionskode gilt für RSX 11/M Version 3.2. Er kann bei verschiedenen Versionen des Betriebssystems verschieden sein. Listing 2 zeigt das in FORTRAN formulierte Sendeprogramm.*

*Die Adressen der RAM-Bereiche des DRAM wurden für den Testbetrieb so programmiert, daß sie identisch mit den Adressen waren, in denen die Programme später im EPROM laufen sollten. Dies waren die ROM-Bereiche, die für Assembler und BASIC des AIM vorgesehen sind. Diese Bereiche waren also im Testbetrieb doppelt zugewiesen. Das funktioniert nur, wenn die entprechenden ROM-Sockel des AIM leer sind. Der Vorteil ist, daß die im RAM lauffähigen Programme ohne jede Änderung in EPROMs gebrannt werden können. Dazu wurde der EPROM Programmer des DRAM benutzt.*

*Ergebnis:*

*Die entwickelte Konfiguration hat die Anforderungen sehr gut erfüllt. Die Anordnung lief während der mehrere Monate dauernden Entwicklung des Meßgerätes fehlerfrei.*

*Da das Ausgangssignal des entwickelten Anemometers nicht linear ist, wurde eine Linearisierungstabelle in die Programme integriert. Einschließlich der Arithmetik-Ubterprogramme, die annähernd unverändert übernommen werden konnten (3), und der Linearisierungstabelle konnten die Programme in zwei EPROMs vom Typ 2716 (je 2 kbyte) untergebracht werden. Nach Erreichen der Zahl von 2000 Programmzeilen mußte die Dimensionierung einiger Felder des Crossassemblers geändert werden. Das war unproblematisch, da der Crossassembler in FORTRAN geschrieben ist. Die Laufzeit des Crossassemblers für Programme dieser Größe lag bei etwa zehn Minuten, ein durchaus akzeptabler Wert. Länger dauerte der Ausdruck der Listings. Trick: Listings nicht drucken, sondern am Terminal ansehen.*

*Erste Tests des Gerätes zeigten, daß es die Erwartungen erfüllt (2). Es ist zu erwarten, daß beim Betrieb des Gerätes mögliche Verbesserungen der zugrundeliegenden Gleichung offenbar werden. Ein wesentlicher Vorteil der rechnergestützten Lösung ist, daß solche Änderungen vorgenommen werden können, ohne den Aufbau des Gerätes zu ändern.*

Literatur:
1. Hennig, E.; et. al.: Eine neue Methode zur nichtinvasiven Bestimmung des Fördermengenstromes pneumatisch angetriebener Blutpumpen. Jahresbericht, BMFT, KUE 401, 1974
2. Spiegelberg, A.; Nielsen, S. D.; Affeld, K.; Frank, J.; Olsen, D. B.: Microprocessor-Based Cardiac Output Measurement in Calves with Pneumatic Artificial Internal Organs 10, 1981
3. Zimmermann, M.: ASP - Advanced Subroutine Package. 65xx MICRO MAG, 1, Nr. 2, 1978, 21

**Listing 1**

```
;
;  6502 CROSS LINKER
;
;  AUTHOR:    A. SPIEGELBERG, 1980
;  DATE:      1-DEC-80
;  FACILITY:  UNIVERSITY OF UTAH, ARTIFICIAL ORGANS
;
```

```
;  YOU CAN LINK UP TO 5 SOURCE FILES
        .SETN NO 0
        .ENABLE SUBSTITUTION
.;  ASK FOR THE INPUT FILE NAMES
.10:    .INC NO
        .IF NO = 6. .GOTO 15
        .ASKS FIL'NO' WHICH SOURCE FILE
        .ASK INDEX DO YOU WISH TO INCLUDE MORE SOURCE FILES ?
        .IFT INDEX .GOTO 10
        .ASK INDPRT DO YOU WISH TO PRINT THE LISTING
        .GOTO 16
.15:    .ASK INDEX TOO MANY FILES, DO YOU WISH TO CONTINUE ?
        .IFF INDEX .GOTO 999
.16:    .ASKS OUTFIL NAME OF OUTPUT FILE ?
.;  NOW MERGE ALL THE DESIRED INPUT FILES INTO ONE FILE NAMED INTFIL.ASM
        .IFNDF FIL5 .GOTO 20
        PIP INTFIL.ASM='FIL1'.ASM,'FIL2'.ASM,'FIL3'.ASM,'FIL4'.ASM,'FIL5'.ASM
        .GOTO 100
.20:    .IFNDF FIL4 .GOTO 30
        PIP INTFIL.ASM='FIL1'.ASM,'FIL2'.ASM,'FIL3'.ASM,'FIL4'.ASM
        .GOTO 100
.30:    .IFNDF FIL3 .GOTO 40
        PIP INTFIL.ASM='FIL1'.ASM,'FIL2'.ASM,'FIL3'.ASM
        .GOTO 100
.40:    .IFNDF FIL2 .GOTO 50
        PIP INTFIL.ASM='FIL1'.ASM,'FIL2'.ASM
        .GOTO 100
.50:    .IFNDF FIL1 .GOTO 999
        PIP INTFIL.ASM='FIL1'.ASM
.;  WRITE ALL THE INFORMATION NEEDED BY THE CROSS ASSEMBLER
.;  INTO ONE FILE
.100:   .OPEN FOR006.DAT
        .DATA INTFIL.ASM
        .DATA 'OUTFIL'.LST
        .DATA 'OUTFIL'.OBJ
        .DATA 'OUTFIL'.INT
        .DATA 3000
        .DATA 'OUTFIL'.MAC
        .DATA 500
        .CLOSE
.;  TELL THE CROSS ASSEMBLER TO GET THE INFORMATION FROM THE SYSTEM DISK
        REA ...ASM 6. SY:
        REA ...ASM 8. SY:
.; WHAT'S THE TIME ?
        TIM
.;  START THE CROSS ASSEMBLER
        ASM
        TIM

.;  WE WANT TO LEAVE THE CROSS ASSEMBLER AS IT WAS
        REA ...ASM 6. TI:
        REA ...ASM 8. TI:
.;  DELETE THE FILES CONTAINING NAMING INFORMATION FOR THE CROSS ASSEMBLER
        PIP FOR006.DAT;*/DE
        PIP FOR008.DAT;*/DE
.;  EXIT IF NO LISTING WAS DESIRED
        .IFF INDPRT .GOTO 900
```

```
.;  PREPARE FILES CONTAINING NAMING INFORMATION FOR THE PRINTER PROGRAM
        .OPEN FOR006.DAT
        .DATA 'OUTFIL'.LST
        .CLOSE
.;  TELL THE PRINTER PROGRAM WHERE TO LOOK FOR THE INFORMATION
        REA ...PRI 1 SY:
        REA ...PRI 6 SY:
.;  RUN THE PRINTER PROGRAM
        PRI
.;  AND REDO WHAT WE CHANGED
        REA ...PRI 1 TI:
        REA ...PRI 6 TI:
.;  FINALLY DELETE ALL THE FILES WE DON'T NEED ANY MORE
        PIP FOR006.DAT;*/DE
        PIP FOR001.DAT;*/DE
.900:   TIM
        PIP INTFIL.ASM;*/DE,'OUTFIL'.INT;*/DE,'OUTFIL'.MAC;*/DE
;
.999:   /
```

**Listing 2**

```
      C
      C  SEND
      C  AUTHOR    : ANDREAS SPIEGELBERG, 1980
      C  FACILITY  : UNIVERSITY OF UTAH, ARTIFICIAL ORGANS
      C
      C
      C  THIS ROUTINE ASKS FOR A FILENAME, READS
      C  DATA FROM THAT FILE ON A LINE BY LINE BASIS AND
      C  SENDS ALL DATA TO A SERIAL LINE ON A BYTE BY BYTE BASIS
      C
      C
0001        BYTE DATA (80)       ! THIS IS THE DATA BUFFER
0002        BYTE NAME (20)       ! THIS IS THE BUFFER FOR THE FILE NAME
0003        DIMENSION KPRM (6)   ! PARAMETER BLOCK FOR WTQIO
0004        DIMENSION KSTAT (2)  ! STATUS BLOCK
0005        DATA IFUWLB/"410/    ! THIS IS THE FUNCTION CODE
      C
      C  WE WANT TO SEND ONE BYTE
      C
0006        KPRM(2)=1
      C
      C  READ THE FILENAME FROM THE TERMINAL
      C
0007        WRITE (5,1000)
0008  1000  FORMAT (X,'PLEASE ENTER THE FILENAME: ',$)
      C
      C  L IS THE NUMBER OF BYTES READ FROM THE TERMINAL
      C
0009        READ (5,2000)L,NAME
0010  2000  FORMAT (Q,80A1)
      C
      C  ASSIGN THAT FILE TO LUN 1
      C
0011        CALL ASSIGN (1,NAME,L)
      C
      C  NOW READ A LINE
      C
0012  100   READ (1,2000,END=900) L,DATA
```

```
      C
      C  AND SEND ALL BYTES TO LUN 2
      C
0013          DO 90 I=1,L
0014          CALL GETADR (KPRM,DATA(I))  !GET THE ADDRESS
0015          CALL WTQIO(IFUWLB,2,1,,KSTAT,KPRM,KSW)
0016  90      CONTINUE
0017          GOTO 100
      C
      C  WE END UP HERE AFTER THE FILE WAS COMPLETED
      C
0018  900     STOP
0019          END                                              Ω
```

Dipl.-Ing. Klaus-Rüdiger Hase, 4350 Recklinghausen

# APPLE II emuliert AIM 65

**Kurzfassung**

Es wird die Implementierung des AIM 65-Monitors mit Assembler und Editor auf einem APPLE II Plus mit 64K-Byte RAM und mindestens einem Diskettenlaufwerk unter DOS 3.3 vorgestellt. Ferner werden die notwendigen Software-Änderungen am Monitor selbst sowie Hilfsprogramme für die Parameterübergabe zur Systemschnittstelle und die Generierung einer geeigneten Slave-Diskette gezeigt. Für den einfacheren Umgang mit dem Editor werden zwei Programme zum Laden und Aufzeichnen auf die Diskette beschrieben. Bei der Hardware beschränkt sich die Darstellung auf den Aufbau einer USER-Parallelschnittstelle und Hinweise für die Single-Step-Option.

**Allgemeine Überlegungen zur Software-Übertragung**

Dieser Artikel wendet sich an solche Betreiber des APPLE-Mikrocomputers, die über einen Programmvorrat für den Rockwell AIM 65 in Maschinensprache verfügen oder Programme benutzen möchten, die speziell für diesen Rechner geschrieben wurden. Es soll gezeigt werden, unter welchen Bedingungen es möglich ist, Betriebssoftware von einem Rechner auf einen anderen Rechner zu übertragen, auch wenn die Hardware nach anderen Gesichtspunkten konzipiert wurde. Eine wesentliche Bedingung, die an das zu übertragende Systemprogramm (Monitor) gestellt werden muß, ist ein geschlossener hirarchischer Aufbau der Programmstruktur, und darüber hinaus sollte das 'gastgebende' System über eine flexible Hardware verfügen.

Beide Rechner, AIM 65 und APPLE II, besitzen den gleichen Prozessor, was die prinzipielle Möglichkeit der Übertragung vorhandener Software eröffnet. Allerdings zeigen sich hier schnell die Grenzen, wenn aus Gründen der Bequemlichkeit Unterprogramme aus dem Monitor mit herangezogen wurden. Die Programmübertragung wird also nur dann gelingen, wenn der wesentliche Teil des alten Monitors im neuen System zur Verfügung steht. Spätestens an dieser Stelle ist der Konflikt vorprogrammiert, denn wegen der Notwendigkeit der festgelegten RESET-, NMI- und IRQ-Vektoren wird der Monitor in der Regel an das obere Ende des Adressierungsbereiches verbannt. Da der Monitor dieser Systeme zwangsläufig eng an die vorhandene Hanrdware gebunden ist, wird der alte Monitor nicht mit der neuen Hardware arbeiten und umgekehrt.

Das Problem ist offensichtlich nur dann zu lösen, wenn beide Monitore im gleichen Adreßbereich verfügbar sind und über einen Softwareschalter wahlweise erreichbar gemacht werden. Diese Forderung läßt sich mit der Language-Karte oder einer ähnlichen RAM-Karte für den Bereich hex D000.. FFFF leicht erfüllen, da mit dem Abschalten der Karte der APPLE-Monitor erreichbar bleibt. Bei der Übertragung des AIM-Monitors auf den APPLE sollen vorerst nur solche Funktionen übernommen werden, die softwaremäßig realisierbar sind. Hierzu gehören die elementaren Monitorfunktionen, wie Anzeigen und Ändern von Speicher- und Registerinhalten, der Miniassembler, der Disas-

sembler, der Editor und der symbolische Assembler. Hingegen läßt sich die Einzelschrittfunktion, die Aufzeichnung auf Band bzw. die Druckerfunktion nur durch zusätzliche Hardware erreichen. Die fehlende USER-VIA kann auf einer entsprechenden Prototypenkarte nachgerüstet werden. Der rein technische Vorgang des Daten- oder Programmtransfers soll nicht Gegenstand der Ausführungen sein, hier sei auf entsprechende Literaturstellen verwiesen /3/, /4/.

**Erforderliche Minimalkonfiguration**

Aus den zuvor gemachten Übelegungen ergeben sich einige Randbedingungen für eine Minimalkonfiguration. Für die weitere Betrachtung muß folgende Ausstattung zur Verfügung stehen:

APPLE II Plus mit 48K-Byte RAM,
16K-Byte RAM-Erweiterungskarte (z.B. Language-Karte),
Video-Monitor,
Diskettenlaufwerk mit Controller, DOS 3.3,
VIA-Karte, z.B. Prototypenkarte mit einer VIA R6522 oder PIA R6520/MC6821 bestückt
Verbindungskabel zur AIM 65-USER-VIA.

**Umschaltung des Monitorbereiches**

Bezeichnet man nun das zu übertragende System als 'Gastsystem' und das System des ausführenden Rechners als 'Gastgebersystem', so ist einzusehen, daß das Gastgebersystem immer dann aktiviert werden muß, wenn der Benutzer mit dem Gastsystem kommunizieren will. Mit anderen Worten: Für jedes ein- oder auszugebende ASCII-Zeichen muß der Gastmonitor ab- und der Gastgeber-Monitor zugeschaltet werden. Bei dieser Umschaltung sind gegebenenfalls Parameter zu retten. Zu beachten ist auch, daß die Umschaltung von 'neutralem Boden', d.h. von einem Adreßbereich aus erfolgen muß, der von der Umschaltung nicht berührt wird und für beide Systeme jederzeit erreichbar ist.

**Retten der 0-Page**

Eine besondere Stellung nimmt bei Systemen mit dem R6502-Prozessor die 0-Page ein. Da sowohl das BASIC und das DOS des APPLE II, als auch der ASSEMBLER des AIM 65 intensiven Gebrauch hiervon machen, empfiehlt es sich, die gesamte 0-Page vor jeder Ausgabe oder Eingabe über die Systemschnittstellen (Bildschirm und Tastatur) zu retten. Dies führt zu einer geringfügigen Verlangsamung bei solchen Operationen, die aber bei der Eingabe vom Tastenfeld aus nicht stört und bei der Ausgabe immerhin eine Übertragungsrate von ca. 1800 Baud ermöglicht, was beim Bildschirm auch nicht weiter schlimm ist.

**Möglichst wenig Änderungen am Monitor**

Ziel der Monitorübertragung soll die größtmögliche Kompatibilität bezüglich vorhandener Software sein. Dies setzt voraus, daß nur in geringem Umfange Änderungen im Monitorprogramm vorgenommen werden. Zwei Änderungen sind jedoch unabdingbar, damit der Beutzer überhaupt mit dem System in Kontakt treten kann. Gemeint sind die Eingabe über die Tastatur und die Ausgabe auf den Bildschirm.

**DILINK-Vektor zeigt auf APPLE-'COUT'**

Was die Ausgabe auf das Display anbelangt, so besitzt der AIM 65 eine elegante Lösung mit dem DILINK-Vektor, der sich vorzüglich für Video-Displays eignet. Es bietet sich also an, diesen Vektor zu benutzen. Da der Vektor erstmalig bei der RESET-Routine aus einer Tabelle heraus abgerufen wird, empfiehlt sich die Modifikation dieser Speicherstellen (hier von hex EF05 auf 0320). Das aufrufende Programm 'OUTDP1' übergibt hierbei das ASCII-Zeichen im Akku an ein entsprechendes Anzeigeprogramm im APPLE. Gemäß den Vereinbarungen im AIM-Monitor ist unter Beibehaltung der Registerinhalte der Parameter an die Routine 'OUTDIS' (EF05) wieder zu übergeben. Nach Abarbeitung der APPLE-Ausgaberoutine ist somit ein Sprung nach OUTDIS zu vollziehen. (Hierzu muß beachtet werden, daß für den APPLE das achte Bit gesetzt, für den AIM 65 aber in der Regel zurückgesetzt werden muß.) Dabei läuft das AIM-Display blind mit. Es werden Zeichen in den Adreßbereich des nicht existierenden Display-PIA (AC00) geschrieben, was sich aber nicht nachteilig auswirkt. Andererseits kann aber auf den Aufruf von OUTDIS nicht verzichtet werden,

da dieses Programm den Display-Buffer DIBUFF verwaltet, der für die meisten hexadezimaleingaben (FROM, TO, ...) und für den Editor von Bedeutung ist.

**Reservierung des Adreßbereiches von hex A0000 bis AFFF**

Die zuvor gemachten Überlegungen haben gezeigt, daß der Monitor-RAMbereich des AIM 65 bei A400 bis A47F reserviert werden muß. Ähnliches gilt auch für die anderen I/O-Schnittstellen im Bereich AXXX. Ist jedoch der APPLE mit 48K-Byte RAM bestückt (auf der Mutterplatine), so wird beim BOOT-Vorgang mit der Masterdiskette das Disk Operating System (DOS) an das obere Ende des zusammenhängenden Adreßbereiches gelegt, was zur Belegung der Speicherstellen 9600 bis BFFF führt.

**Generierung einer geeigneten Slave-Diskette**

Somit muß eine Slave-Diskette kreiert werden, die den Bereich oberhalb A000 nicht zerstört. Dies ist einfach möglich, wenn man einen der RAM-Bausteine entfernt, die den Adreßbereich von 8000 bis BFFF versorgen und dann mit 'INIT' eine solche Slave-Diskette initialisiert. Das DOS erkennt dann nur einen fehlerfreien RAM-Bereich bis 7FFF, wie dies bei einer 32K-Byte Bestückung der Fall gewesen wäre (siehe auch /5/). Diese einfache Lösung hat aber den Nachteil, daß der für den AIM 65 nutzbare Adreßbereich nur von hex 0800 bis 5600 zur Verfügung steht, während der Bereich von 8000 bis A3FF ungenutzt bliebe. Dieser Schönheitsfehler läßt sich durch eine kleine Hardware umgehen, die bei der Adressierung des Bereiches A000 bis BFFF die RAM-Prüfroutine des DOS ein Ende erkennen läßt. Auf der USER-VIA-Platine läßt sich diese Hilfsschaltung leicht provisorisch nachrüsten, zumal sie nur einmal gebraucht wird. Hat man nämlich einmal eine entsprechende 40k-Byte-Slave-Diskette erzeugt, so kann man hiervon beliebig viele Kopien erstellen und die Hilfsschaltung wieder abbauen.

**Überspielen des Monitors**

Der AIM 65-Monitor muß nun in den APPLE übertragen werden. Dies kann mit den Techniken geschehen, wie sie in /3/ und /4/ näher beschrieben sind. Hierbei arbeitet der AIM als Quelle und der APPLE als Empfänger über die beiden USER-VIAs. Eine andere Möglichkeit besteht darin, einen ROM-Sockel auf eine Prototypenplatine zu setzen und mit dem I/O-Strobe zu dekodieren. Die AIM-ROMs können dann bereichsweise gelesen werden. Der Einfachheit halber wird der gesamte Bereich von hex D000 bis FFFF (Assembler und Monitor) in den APPLE auf die Speicherplätze 1000 bis 3FFFF kopiert. Eine Übertragung in den endgültigen Bereich ist noch nicht möglich, da mit Hilfe des APPLE-Monitors noch einige Veränderungen vorgenommen werden müssen. Zur Sicherheit sollte diese Originalversion mit 'BSAVE' auf Diskette gesichert werden.

**Änderungen des Original-AIM 65-Monitors**

Nachfolgend sind die Änderungen aufgeführt, die nach Aufruf des APPLE-Monitors am Monitor des AIM 65 vorgenommen werden müssen, wobei davon ausgegangen wird, daß die Übertragung wie oben beschrieben ausgeführt wurde. Danach beginnt der Assembler bei 1000, entsprechend D000 im Original und endet der Monitor bei 3FFF entsprechend FFFF. In den Klammern sind jeweils die ursprünglichen Werte eingetragen, während davor die neue Hexadezimalzahl steht.

```
112D : EA  (20)    Vermeidung der Quellcode-Ausgabe
112E : EA  (07)    beim Assemblierungsvorgang, bewirkt
112F : EA  (E9)    eine erhebliche Beschleunigung !

211D : 4C  (2C)    Überspringen des KB/TTY - Tests
211E : 44  (00)    während der RESET-Initialisierung
211F : E1  (A8)

2215 : 70  (00)    Tastenbelegung für (1) zum
2216 : 03  (B0)    APPLE - Monitor hin
2217 : 80  (03)    Tastenbelegung für (2) zum
2218 : 03  (B0)    APPLE - BASIC hin
```

```
275A : 10  (05)    RESET-Initialisierungstabelle für
275B : 03  (EF)    DILINK-Vektor nach APPLE-Display

27EB : 00  (14)    Display-Scrolling für 'DEL'

2844 : EA  (2C)    'TTYTST' erkennt Schalter KB/TTY
2845 : EA  (00)    auf KEYBOARD und gibt damit nicht
2846 : EA  (A8)    über die serielle Schnittstelle aus.

2C44 : 20  (2A)    Sprung zur entsprechenden Routine
2C45 : 03  (ED)    'RDCHAR' des APPLE und
2C46 : 60  (A9)    dann Rücksprung mit RTS
2C47 : 60  (8F)

2CEF : 60  (AD)    'ROONEK' wird nicht bedient, RTS *)

3045 : 60  (2C)    ebenso 'IPST' für Thermoprinter
```

*) Eine Unterbrechungsmöglichkeit bei Listings ist beim APPLE II mit CNTRL-S gegeben. Weitere Einzelheiten zu den Monitorroutinen sind unter /2/ und /6/ zu finden.

**Hilfsprogramme für die Ein-/Ausgabe über Systemschnittstellen**

Auf besagtem 'neutralen Boden' sind nun noch die Programme unterzubringen, die die Übergabe der ASCII-Zeichen vom Gastsystem zum Gastgebersystem und umgekehrt ermöglichen. Der APPLE bietet hierzu den Bereich von 0300 bis 03CF an, der vom APPLE-Monitor nicht benutzt wird. Die einzelnen Einsprungspunkte wurden auf 'glatte' Adressen (300, 310, 320 usw.) gelegt. Ihre Bedeutung geht aus dem nachfolgenden Listing hervor.

Sind alle diese Veränderungen eingegeben, so kann der AIM-Monitor z.B. mit 'BSAVE AIM.MON, A $1000,L $3000' auf die Diskette übertragen werden. Die Verschiebung nach D000 geschieht dann mit dem entsprechenden DOS-Kommando 'BLOAD AIM.MON,A$D000' vom HELLO-Programm aus.

```
Einsprung zum AIM65 - Monitor

0300-   EE 82 C0    INC   $C082     Schreibschutz f. RAM
0303-   EE 80 C0    INC   $C080     Lesezugriff auf RAM
0306-   20 40 03    JSR   $0340     Austausch der 0-Page
0309-   20 90 03    JSR   $0390     Setze APPLE-Vektoren
030C-   6C FC FF    JMP   ($FFFC)   Sprung nach AIM-RESET

Ausgabe eines ASCII - Zeichens auf den Bildschirm

0310-   EE 81 C0    INC   $C081     Umschaltung APPLE-Mon.
0313-   48          PHA               rette ASCII - Zeichen
0314-   09 80       ORA   #$80       setze MSB = 1
0316-   20 5C 03    JSR   $035C     Ausgabe, tausche 0-Page
0319-   68          PLA               Wiederherstellung
031A-   EE 80 C0    INC   $C080     Umschaltung auf RAM
031D-   4C 05 EF    JMP   $EF05     weiter nach OUTDIS

Eingabe eines ASCII - Zeichens von Tastatur

0320-   EE 81 C0    INC   $C081     Umschaltung auf APPLE-Mon.
0323-   20 40 03    JSR   $0340     Tauschen der 0-Page
0326-   20 0C FD    JSR   $FD0C     'RDCHAR' des APPLE-Monitor
```

```
0329-   C9 88       CMP   #$88      Rückschritt, 'DEL'
032B-   D0 03       BNE   $0330     wenn nein, dann weiter
032D-   20 F0 FD    JSR   $FDF0     wenn ja, dann 'COUTl'
0330-   20 40 03    JSR   $0340     Rücktausche der 0-Page
0333-   29 7F       AND   #$7F      setze MSB = 0
0335-   EE 80 C0    INC   $C080     Umschaltung auf AIM-Mon.
0338-   60          RTS
```

Austausch der 0-Page mit der A5-Page

```
0340-   48          PHA             Es werden keine Register
0341-   8A          TXA             verändert.
0342-   48          PHA
0343-   98          TYA
0344-   48          PHA
0345-   A2 00       LDX   #$00      Schleifenzähler = Index
0347-   B5 00       LDA   $00,X
0349-   BC 00 A5    LDY   $A500,X   Austausch der ganzen
034C-   94 00       STY   $00,X     0-Page gegen die
034E-   9D 00 A5    STA   $A500,X   A5-Page

0351-   E8          INX             nächster Platz
0352-   D0 F3       BNE   $0347     bis fertig
0354-   68          PLA
0355-   A8          TAY             Wiederherstellung der
0356-   68          PLA             Registerinhalte
0357-   AA          TAX
0358-   68          PLA
0359-   60          RTS

035C-   20 40 03    JSR   $0340     noch zur Ausgabe gehörend
035F-   20 F0 FD    JSR   $FDF0     'COUTl' mit
0362-   20 40 03    JSR   $0340     Rücktausch der 0-Page
0365-   60          RTS
```

Einsprung in den APPLE-Monitor vom AIM65 aus

```
0370-   EE 81 C0    INC   $C081     Umschaltung auf ROM
0373-   20 40 03    JSR   $0340     Austausch der 0-Page
0376-   4C 69 FF    JMP   $FF69     APPLE-Monitoranfang
```

Einsprung in das APPLESOFT-BASIC mit Erhalt des Programms

```
0380-   EE 81 C0    INC   $C081     Umschaltung auf ROM
0383-   20 40 03    JSR   $0340     Austausch der 0-Page
0386-   4C 3C D4    JMP   $D43C     APPLESOFT Warm-Start
```

Setzen der Rücksprung-Vektoren für '&' und CNTRL-Y

```
0390-   A9 4C       LDA   #$4C      'JMP'
0392-   8D F8 03    STA   $03F8
0395-   8D F5 03    STA   $03F5     Sprung zur
0398-   A9 00       LDA   #$00
039A-   8D F9 03    STA   $03F9     Startadresse $300
039D-   8D F6 03    STA   $03F6
03A0-   A9 03       LDA   #$03      zum AIM65 - Monitor
03A2-   8D FA 03    STA   $03FA
```

```
03A5-   8D F7 03     STA   $03F7
03A8-   60           RTS
```

Die Programmeingabe kann wieder in hexadezimaler Form mit dem APPLE-Monitor erfolgen und wird dann zum Beispiel mit 'BSAVE AIM.START300,A$300,L$A9' auf eine Diskette überspielt. Es muß darauf geachtet werden, daß die DOS-Vektoren am oberen Ende der 3. Page nicht verändert werden. Nach dem erstmaligen Starten des AIM-Monitors mit '300G' aus dem APPLE-Monitor heraus, sollte dann die gesamte 3. Page mit 'BSAVE AIM.START300,A$300,L$100' unter dem gleichen Filenamen auf die Diskette geschrieben werden, da dann alle Vektoren automatisch richtig gesetzt werden.

**'HELLO'-Programm zur selbständigen Initialisierung**

Mit den zuvor eingegebenen Programmen ist der wesentliche Teil der Monitorübertragung schon geschehen, allerdings läßt sich der AIM-Monitor nicht ohne weiteres aus der BASIC-Kommandoebene laden. Dies muß vielmehr aus einem Programm heraus erfolgen. Mit einigen Änderungen läßt sich hierzu das 'HELLO'-Programm des DOS-Masters verwenden. Die nachfolgenden BASIC-Programmzeilen geben daher nur diese Änderungen wieder:

```
20   REM     -- START F. AIM-MONITOR
30   TEXT : PRINT
40   HOME
50   PRINT "DOS VERSION 3.3                25.06.81"
60   PRINT : PRINT "APPLE II PLUS    RAMCARD   BOOT AIM-MON."
65   PRINT "    ( 40 K-BYTE   DOS: $7600 ... $9FFF )"
66   PRINT : PRINT "            ( AIM : $AXXX )"
190  REM
200  VTAB 10: PRINT "(LOADING AIM-MONITOR INTO  RAM-CARD)"
210  PRINT CHR$(4);"BLOAD AIM.MON,A$D000"
240  GOTO 1000
350  END
1000 VTAB 15: PRINT "(LOADING 'AIM.START300' --> $300)
1020 PRINT CHR$(4);"BLOAD AIM.START300,A$300"
2050 VTAB 10
2055 PRINT "          ------ & ----->              "
2060 PRINT "BASIC (J)                  AIM (<)"
2065 PRINT "          <----- 2 ------              "
2070 PRINT : PRINT
2075 PRINT "          --- CNTRL'Y -->              "
2080 PRINT "MONIT.(*)                  AIM (<)"
2085 PRINT "          <----- 1 ------              "
2090 PRINT
9999 END
```

**DOS/BASIC-Hilfsprogramme für den AIM 65-EDITOR**

Die nachfolgenden beiden BASIC-Programme sind nicht unbedingt zur Simulation des AIM 65 erforderlich, erlauben aber eine bequeme Handhabung des Editors mit der Diskette. Hierbei können nur ganze Files geladen oder gespeichert werden, mit der Randbedingung, daß der Editorfile immer bei hex 1000 beginnt und maximal bei 7600 endet. Die Files werden als Maschinenfiles mit der Kennzeichnung 'B' auf die Diskette geladen. Um Verwechslungen mit Maschinenprogrammen zu vermeiden, wird jedem dieser Files automatisch ein Zusatz 'EDI.' vorangestellt. Die Programme werden aufgerufen, indem man mit 'Q' den Editor und dann mit '2' den AIM-Monitor verläßt und schließlich mit 'RUN EDI IN' oder 'RUN EDI OUT' startet. Nach Ausführung der Programme befindet man sich wieder in der Kommandoebene des AIM 65-Monitors. Die Programme belegen nur den Bereich von 0800 bis 0FFF, was durch die Anweisung 'HIMEM:4095' gewährleistet wird.

'EDI IN' Hilfsprogramm zum Laden eines Editor-Files von Diskette

```
10  HOME : HIMEM: 4095 : REM Begrenzung bis max. $0FFF
15  AA = 42240: REM   $A500  0-Page Puffer
20  D$ = CHR$(4) : PRINT : PRINT "LADE AIM-EDIT-FILE"
25  PRINT : PRINT "DISKETTE EINLEGEN ... (RET) DRÜCKEN";: GET L$
26  LL$ = " ": PRINT
27  IF L$ = "1" OR L$ = "2" THEN LL$ = ",D" + L$
30  PRINT : PRINT D$;"CATALOG";LL$: PRINT : INPUT "FILE : ";F$
40  PRINT D$;"BLOAD";F$;",A$1000";LL$
45  PRINT " "
50  POKE AA + 227,0: POKE AA + 228,16: REM  TEXT
55  HOME : PRINT "FILE  '";F$;" ' GELADEN": PRINT
60  POKE AA + 229,0: POKE AA + 230,112: REM  END
100 CALL 937: REM  SUCHE EDI - ENDE $3A9
110 PRINT " <T>  ---> AIM-EDITOR": PRINT
150 & : END : REM Sprung zum AIM65 - Monitor
```

Dieses Programm ruft mit 'CALL 937' ein Maschinenprogramm auf, welches das Ende des Textfiles sucht und dann in den Pointer 'END' entsprechend $E1, $E2 überträgt. Das Hilfsprogramm findet noch in der 3. Page einen Platz (siehe unten).

'EDI OUT'  Überspielung eines EDITOR-Files auf Diskette

```
10  HIMEM: 4095 : REM Begrenzung bis max. $0FFF
15  AA = 42240: REM   $A500
20  D$ =  CHR$ (4): PRINT : PRINT "SPEICHERE AIM-EDIT-FILE"
25  PRINT : PRINT "DISKETTE EINLEGEN ... kRETg DRÜCKEN";: GET L$
26  LL$ = " ": PRINT
27  IF L$ = "1" OR L$ = "2" THEN LL$ = ",D" + L$
45  PRINT " "
50  T =  PEEK (AA + 227) +  PEEK (AA + 228) * 256: REM TEXT
60  BOTLN =  PEEK (AA + 225) + PEEK (AA + 226) * 256: REM BOTLN
70  FOR T2 = T TO T + 60: IF  PEEK (T2) k 32 THEN 80
75  PRINT  CHR$ ( PEEK (T2));: NEXT
80  PRINT : PRINT : INPUT "FILE : ";F$
82  T =  INT (T + .5):LE =  INT (BOTLN - T + 2.5)
85  PRINT D$;"BSAVE EDI.";F$;",A";T;",L";LE;LL$
105 PRINT "FILE  '";F$;" ' GESPEICHERT": PRINT
110 PRINT " <T>  ---> AIM-EDITOR": PRINT
150 & : END : REM Sprung zum AIM65 - Monitor
```

Assembler - Hilfsprogramm für das Auffinden des EDITOR-Endes

```
03A9-   20 40 03    JSR   $0340     Umschalten der 0-Page
03AC-   A5 E3       LDA   $E3       'TEXT' nach 'END'
03AE-   85 E1       STA   $E1       übertragen
03B0-   A5 E4       LDA   $E4
03B2-   85 E2       STA   $E2
03B4-   A2 00       LDX   #$00      Index auf 00
03B6-   A1 E1       LDA   ($E1,X)   Suche Endmarke = 00
03B8-   F0 0C       BEQ   $03C6
03BA-   E6 E1       INC   $E1       nächster Platz bis Ende
03BC-   D0 F6       BNE   $03B4
03BE-   E6 E2       INC   $E2
03C0-   A5 E2       LDA   $E2
```

```
03C2-   C9 75      CMP   #$75      größte Endadresse
03C4-   D0 EE      BNE   $03B4
03C6-   A9 00      LDA   #$00      Endmarke setzen
03C8-   81 E1      STA   ($E1,X)
03CA-   20 40 03   JSR   $0340     Austauschen der 0-Page
03CD-   60         RTS
```

**Hardware für USER-VIA-Platine**

Der APPLE II besitzt in seiner Grundausstattung keine Parallelschnittstelle wie der AIM 65. Durch die Anordnung von acht Slots (Steckplätzen für Leiterplatten vom APPLE-Format) und der Adreßdekodierung auf der Mutterplatine ist die Erweiterung um eine solche Schnittstelle denkbar einfach. Für jeden Slot steht am Kontakt Nr. 41 ein 'DEVICE SELECT'-Selectsignal zur Verfügung, welches speziell für die Ansteuerung von Schnittstellenbausteinen vorgesehen ist. Dieses soll als 'Chip-Select' herangezogen werden, ist aber mit dem Taktsignal Φ1 verknüpft und geht daher wenige nsec (Nanosekunden) später in den aktiven Low-Zustand über als das Φ1-Signal selbst. Da aber das ENABLE-Signal für den R6522 aus dem Φ1-Signal durch Invertierung gewonnen werden soll (Φ2 steht nicht zur Verfügung), werden die Zeitbedingungen für diesen relativ komplizierten Baustein verletzt und ein direkter Anschluß des VIA 6522 führt zu einer sehr unsicheren Betriebsweise. Abhilfe schafft die in der Abbildung 1a dargestellte Schaltung, die lediglich eine Verschiebung der positiven Flanke des invertierten O1-Signales erzeugt, Abbildung 1b. In der Anleitung zum Prototypen-Board wird zwar auf mögliche Schwierigkeiten hingewiesen, deren Ursache oder eine Lösung nicht angegeben.

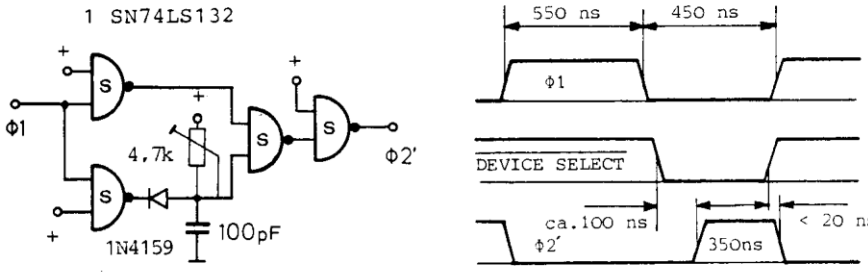



Abb. 1 a: Schaltung des VIA 6522, b: Zeitdiagramm

In der Abbildung 2 ist die einfache Anordnung angegeben, mit der die DOS 3.3-Masterdiskette beim BOOT-Vorgang das RAM-Ende bei 9FFF erkennt, da beim Lesen der Speicherstellen A000 bis BFFF zwangsweise D0 oder D1 auf 'LOW' gelegt und damit Fehler erzeugt werden. Nach Initialisierung eine Slave-Diskette muß diese Schaltung wieder abgebaut werden.

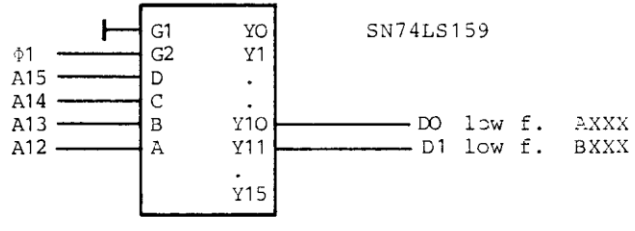



Abb. 2 : Hilfsschaltung für den Bootvorgang

**Aufteilung der Adreßbereiche**

Beim Arbeiten mit dem AIM-Monitor ist immer darauf zu achten, daß nicht die RAM-Bereiche des APPLE-Monitors sowie die des DOS zerstört werden, da dann in der Regel die DOS-Funktionen nicht mehr benutzt werden können. Nachfolgend sind alle wesentlichen Adreßbereiche und ihre Nutzung aufgeführt.

| Adresse von ... bis | | | Nutzung durch | Bemerkung |
|---|---|---|---|---|
| 0000 | 00FF | | AIM / APPLE | 0-Page Austausch mit $A5XX |
| 0100 | 0114 | | AIM | USER-Vektoren,F1,F2,F3 |
| 0115 | 01FF | | AIM / APPLE | Stack (Assembler-Buffer) |
| 0200 | 02FF | | APPLE | 'GETLN' Input Buffer |
| 0300 | 03CF | | AIM / APPLE | Umschaltprogramme |
| 03D0 | 03FF | | APPLE | MON./DOS-Vektoren |
| 0400 | 07FF | | APPLE | Display Text Seite |
| 0800 | 0FFF | | APPLE - BASIC | EDITOR - Hilfsprog. |
| 1000 | 75FF | | AIM | freier RAM - Bereich |
| 7600 | 9FFF | | APPLE - DOS | für 40K-Byte Slave |
| A000 | A3FF | | AIM | frei (USER-VIA) |
| A400 | A497 | | AIM | RIOT 6532 Adressen |
| A498 | A4FF | | - | freier Bereich |
| A500 | A5FF | | AIM / APPLE | 0-Page Puffer |
| A600 | A7FF | | - | freier Bereich |
| A800 | A80F | | AIM | VIA 6522 Printer,Tape |
| A810 | ABBF | | - | freier Bereich |
| AC00 | AC03 | | AIM | PIA 6520 Display |
| AC04 | BFFF | | - | freier Bereich |
| C000 | C07F | | APPLE | System I/O |
| C080 | C08F | *) | AIM / APPLE | Soft-Schalter f. RAM |
| C090 | C7FF | | AIM / APPLE | I/O - Slots |
| C800 | CFFF | | APPLE | I/O - Programme |
| D000 | FFFF | a | APPLE | BASIC, MONITOR |
| D000 | FFFF | b | AIM | ASSEMBLER, MONITOR |

*) bei Adresse Umschaltung auf :

| | |
|---|---|
| C080 | RAM - Karte , Lesezugriff |
| C081 | ROM - Bereich |
| C082 | ROM - Bereich mit Schreibschutz für RAM |
| C083 | RAM - Karte , Schreibzugriff |

Für das Ansprechen dieser Softwareschalter wurden INC-Befehle verwendet, die die Prozessorregister nicht verändern und durch ihren zweimaligen Speicherzugriff (read-modify-write) einen sicheren Betrieb gewährleisten.

**Hinweise für weitere Ergänzungen**

Eine der nützlichen Einrichtungen zur Programmerprobung ist die Fähigkeit des AIM 65, Assemblerprogramme im Einzelschrittverfahren zu testen, im Trace-Modus zu verfolgen oder Soft-Breakpoints zu setzen. Alle diese Möglichkeiten beruhen aber darauf, daß der Prozessor vor jeder Befehlsausführung einen NMI-Impuls erhält und damit in das entsprechende Single-Step-Programm einsteigt. Das NMI-Signal wird durch eine Adreßauswahllogik auf der AIM 65-Platine erzeugt und bei geschlossenem Single-Step-Schalter dem NMI-Eingang des Prozessors zugeführt. Mit der in Abbildung 3 angegebenen Schaltung, die sich noch leicht auf die Schnittstellenplatine integrieren läßt,

kann sogar ein beliebig einstellbarer Adreßbereich von jeweils 4K-Byte der Single-Step-Ausführung unterworfen werden. Hierzu genügen ein Vergleicher und ein Binärschalter sowie zwei Gatter.

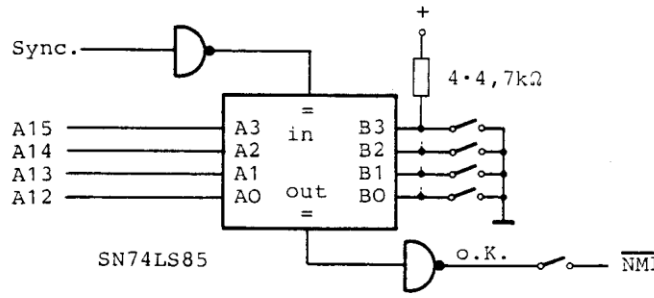



Abb. 3 : Schaltung für Single - Step mit 4bit - Vergleicher

**Bedienungshinweise**

Das Anbooten des neuen Systems geschieht in gewohnter Form durch Einlagen der Diskette und Einschalten oder durch Eingabe von 'PR#6'. Der Bildschirm wird vom HELLO-Programm gelöscht und zeigt dann die möglichen Übergänge zwischen den Systemen. Mit '&' und 'RETURN' wird der AIM-Monitor über die Kaltreset-Routine initialisiert. Jedes Nochmalige Einsteigen in den Monitor bewirkt einen Warmreset. Mit den Tasten '1' und '2', die im AIM 65 dem Tonbandschalter vorbehalten sind, kann wahlweise in den APPLE-Monitor oder in die Kommandoebene des APPLESOFT -BASIC zurückgesprungen werden.

Der AIM 65 meldet sich mit seinem vertrauten Monitor-Prompt und kann wie gewohnt bedient werden. Die Funktion der 'DEL'-Taste wird von der Taste mit dem rückwärtigen Pfeil ( ← ) wahrgenommen. Es ist zu beachten, daß das überfahrene Zeichen im Gegensatz zum AIM-Display nicht gelöscht wird. Die Betätigungtaste mit dem Pfeil ( → ) kann nicht benutzt werden; eine solche Funktion ist im AIM-Monitor auch nicht vorgesehen.

Beim Aufruf des Editors mit (E) ist die untere Grenze (FROM=) mindestens mit hex 800 anzugeben, bei der Verwendung der Hilfsprogramme mit hex 1000. Die obere Grenze ist unbedingt manuell zu setzen und sollte 7500 nicht übersteigen. Ein automatisches Setzen mit 'SPACE' oder 'Return' führt zur Zerstörung des DOS, kann aber bei Betrieb mit Kassettenrecorder und bei großem Speicherbedarf durchaus nützlich sein. Das Hilfsprogramm 'EDI IN' zerstört einen zuvor im RAM befindlichen Editor-File, setzt aber die Zeiger TEXT, END und BOTLN des Editors automatisch im Bereich von 1000 bis maximal 7500. Bei Verwendung der Encoderplatine für die Groß-/Kleinschreibung ist zu beachten, daß der AIM 65-Monitor nur Großbuchstaben als Kommandozeichen anerkennt, der Editor aber einwandfrei den gesamten Zeichenvorrat verarbeitet. *)

Beim Aufruf des Assemblers ist nur die Assemblierung aus dem Speicher in den Speicher vorgesehen, was für die meisten Programme bis zu 4K-Byte Objekte-Code ausreichen mag, wobei sich der Bereich von B000 bis BFFF als Symbolspeicher anbietet. Die Löschung der Display-Ausgabe für jedes Zeichen des Quellprogrammes ergibt eine erhebliche Beschleunigung der Assemblierungsgeschwindigkeit.

Die USER-Vektoren können in gewohnter Weise benutzt werden. solange sie nicht Einfluß auf den DILINK-Vektor nehmen. Die Funktion des DILINK-Vektors übernehmen jetzt die Speicherstellen 031E und 031F.

Die erstellte Slave-Diskette ist mit dem 'COPYA'-Programm des DOS 3.3-Masters kopierbar.

*) Der hier vorgestellte Implementierungsvorschlag stellt nur eine gewisse Minimallösung dar und kann beliebig ergänzt werden. Bei umfangreicheren Modifikationen empfiehlt es sich unbedingt,

einen Änderungsfile im AIM-Monitor anzulegen und jeweils in den AIM.MON.ORIGINAL-File zu assemblieren.

Literatur:

/1/ Rockwell International — AIM 65 USER'S GUIDE; Oct. 1978, Docoment No. 29650 N36
/2/ Rockwell International — AIM Monitor Program Listing; Document No. 29650 N36L, 1978
/3/ Kornnagel, U., Krohn, G. und Bach, P. — Datenaustausch zwischen zwei Mikrocomputersystemen; 65xx MICRO MAG Nr. 15 bis 17, Ahrensburg, 1981
/4/ Löhr, Roland — AIM 65 am IEEE 488-Bus, 65xx MICRO MAG Nr. 19, Juni 1981
/5/ APPLE Computer Inc. — The DOS Manual; DOS 3.3, Cupertino, California, 1979, APPLE Product No. A2L0036 (030-0115-00)
/6/ APPLE Computer Inc. — APPLE II REFERENCE MANUAL; Cupertino, California, 1979 APPLE Product No. A2L0001A (030-0004-011)

Ω

Christian Streicher, 8034 Germering

# Object-Code Editor (AIM)

In der Fachliteratur stößt man häufig auf nützliche Dienstleistungsprogramme, die aus Platzgründen nicht im Quelltext abgedruckt sind. Da bei diesen Programmen jedoch häufig Systemanpassungen notwendig sind, um sie mit einem vorhandenen DOS oder TV-Interface zu verbinden, liegt der Wunsch nahe, den Object-Code mit etwas mehr Konfort bearbeiten zu können, als es der AIM-Dis-Assembler gestattet. Die gebräuchlichste Form der direkten (wahlfreien) Datenaufbereitung ist das Editieren. Es müßte also möglich sein, ein Maschinenprogramm nach Kriterien bearbeiten zu können, die denen des Editierens ähnlich sind. Das vorliegende Programm ist eine Kombination von Mini-Assembler, Disassembler und Editor.

**Initialisierung**

Das Starten des Programms erfolgt durch Drücken der Benutzerfunktionstaste F1. Anschließend wird der Text 'OBJECT-CODE EDITOR' als Kennung ausgegeben. Nun müssen Start- und Endadresse des Object-Code Puffers auf die Fragen FROM= und TO= eingegeben werden. Wenn keine Eingabe erfolgt ist (CR), wird der zuletzt initialisierte Wert angenommen. Dies dient zur Vereinfachung des Wiedereintritts nach eventuellen Probeläufen. Es ist dabei zu beachten, daß sich die Endadresse nach INSERT-Befehlen verändern kann!

Das Programm ist jetzt im Befehlseingabemodus. Jedes eingegebene Zeichen wird mit einer Kennzeichnung beantwortet und ausgeführt. Ungültige Kommandos erzeugen eine Fehlermeldung. Es ist dafür zu sorgen, daß das in Zelle hex 0006 stehende Flag beim Erststart den Wert 00 hat.

**Der Befehlssatz**

Der Befehlssatz des OBJECT-CODE EDITORS kann in zwei Gruppen eingeteilt werden (die nachfolgend in den eckigen Klammern stehenden Zeichen geben die Syntax der Kommandos an):

**1. Byteorientierte Kommandos**

Die nachfolgenden Kommandos bearbeiten lediglich ein Byte. Die Einführung dieser Kommandogruppe ist durch das Verhalten des AIM-Disassemblers bedingt. Es ist mit erheblichem Mehraufwand an Programmlänge jedoch durchaus möglich, diese Kommandos wortorientiert zu gestalten.

U – UP
Setzen des Zeilenzählers auf das nächstniedrige Byte und Disassemblierung des sich daraus ergebenden Wortes.

D – DOWN
Setzen des Zeilenzählers auf das nächsthöhere Byte und Disassemblierung.

T – TOP
Setzen des Zeilenzählers auf den Pufferanfang

B -- BOTTOM
Setzen des Zeilenzählers auf das Pufferende.

(SPC) -- SHOW LINE
Disassemblierung des Bytes, auf das der Zeilenzähler augenblicklich eingestellt ist (ggfs. Bildung eines Wortes).

* -- SET LINE COUNTER
Setzen des Zeilenzählers [ =<*>=HEX ADRESSE ].

**2. Wortorientierte Befehle**

Diese Befehlsgruppe bearbeitet jeweils ein Wort, bestehend aus 1, 2 oder 3 Bytes.

K – KILL
Löschen einer definierten Anzahl von Worten [ =<K> , / ANZAHL DER WORTE ]
Die Worte werden nicht durch Verschieben der nachfolgenden Befehle gelöscht, da sich dadurch alle absoluten Adressen ändern würden. Das Entfernen der Befehle aus dem Programm erfolgt durch Überschreiben mit NOP-Befehlen. Die gelöschten Zeilen werden disassembliert und zur Kontrolle ausgegeben.

L – LIST
Disassemblierung einer definierten Anzahl von Worten [ =<L> , / ANZAHL DER WORTE ]

E – ENTER
Eingabe einer beliebig langen Folge von ASCII-Zeichen [ =<E> , 'ASCII' ].
Der Eingabestring wird mit (CR) abgeschlossen und anschließend disassembliert.

Die beiden letzten Kommandos müssen durch einen Wiedereintritt in den Object-Code Editor ausgeführt werden, da der Assembler des AIM (Benutzung unter I-Kommando) nur mit (ESC) verlassen werden kann.

I – INSERT
Einfügen einer beliebigen Anzahl von Befehlen vor den Befehl, auf den der Zeilenzähler zeigt
[ =<I> , ADDR ... , (ESC) , F1-TASTE ].
Nach dem Drücken der I-Taste befindet man sich in der AIM-Mnemo-Eingabe. Sind alle gewünschten Befehle eingegeben, verläßt man diesen Modus durch (ESC). Anschließend wird mit der F1-Taste der eigentliche Editiervorgang ausgelöst. Dieser besteht darin, ein Link in Form eines JSR-Befehls über den dem Zeilenzähler nachstehenden Befehl zu schreiben. Da die 6502 byteorientiert arbeitet, sorgt eine Befehlslängenauswertung dafür, daß dem JSR-Befehl mindestens 3, maximal jedoch 5 Bytes zur Verfügung gestellt werden. Die nicht benötigten Bytes werden mit NOP-Befehlen überschrieben. Beim Wiedereintritt in den Object-Code Editor werden alle herausgenommenen Befehle an das Ende des vom Programmierer eingegebenen 'Unterprogramms' angehängt und mit einem RTS abgeschlossen. Der INSERT-Befehl arbeitet daher viruell, denn der logische Programmfluß bleibt erhalten, der physikalische Adreßraum wird jedoch unterbrochen. Es ist daher darauf zu achten, daß der nächste relative Sprung mindestens 5 Bytes hinter der echten Adresse des INSERT-Befehles steht. Die im Eingabemodus angezeigte Adresse entspricht der physikalischen Adresse des erzeugten 'Unterprogramms'.

F – FIND
Suchen und Disassemblieren einer bestimmten Anzahl von gleichen Worten
[ =<F> , 0007 SUCHSTRING , (ESC) , F1-TASTE , / ANZ. Z. FIND. WORTE ].
Nach dem Drücken der Taste FIND wird ein Suchkriterium in Mnemo-Form in einen Puffer (0007) assembliert. Bei Eingabe von mehr als einem Wort wird nur das zuerst eingegebene Wort als Suchkriterium anerkannt. Anschließend wird der Eingabemodus mit (ESC) verlassen. Beim Wiedereintritt in den Editor fragt das Programm mit '/' die Anzahl der Worte ab, die gesucht werden sollen. Daraufhin werden das Suchkriterium und alle gefundenen Worte disassembliert. Der Programmzeiger zeigt anschließend auf das letzte gefundene Wort.

```
0007  0000              ;*** ARBEITSSPEICHER ***

0008  0000                    *=$0
0009  0000              START  *=*+2          ; CODE-ANFANG
0010  0002              CURREN *=*+2          ; ZEIGERPOSITION
0011  0004              END    *=*+2          ; CODE-ENDE
0012  0006              FLAG   *=*+1          ; FIND-FLAG
0013  0007              BYTES  *=*+6          ; SUCH-CODE SPEICHER
0014  000D              TEMP   *=*+2          ; ZWISCHENSPEICHER
0015  000F              CR     =$0D


0016  000F              ;*** MONITOR-ROUTINEN ***

0017  000F              COUNT  =$A419
0018  000F              DONE   =$E790
0019  000F              PSL1   =$E837
0020  000F              GCNT   =$E785
0021  000F              SAVPC  =$A425
0022  000F              FROM   =$E7A3
0023  000F              CKSUMS =$A41E
0024  000F              ADDR   =$A41C
0025  000F              WRITAZ =$EA42
0026  000F              BLANK  =$E83E
0027  000F              ADDIN  =$EAAE
0028  000F              TO     =$E7A7
0029  000F              PATCH8 =$FEBC
0030  000F              QM     =$E7D4
0031  000F              CRLF   =$E3F0
0032  000F              OUTPUT =$E97A
0033  000F              DISASM =$F46C
0034  000F              RDRUB  =$E95F
0035  000F              JD1    =$E723
0036  000F              MNEENT =$FB9E
0037  000F              LENGTH =$EA
0038  000F              RCHEK  =$E907

0039  000F                    *=$10C
0040  010C  4C 00 02          JMP ENTRY       ; START VIA F1-TASTE


PAGE 02    PROGRAMM-VERZWEIGER
LINE # LOC  CODE              LINE

0042  010F                    *=$200

0043  0200  A5 06       ENTRY  LDA FLAG        ; TESTE FIND-FLAG
0044  0202  F0 2B               BEQ STRT0       ; --> KEIN FIND-WIEDEREINTI
0045  0204  10 03               BPL *+5         ; ENTRITT NACH 'INSERT'
0046  0206  4C D6 03            JMP FND1
0047  0209  AD 25 A4            LDA SAVPC       ; ENDE AUF PC-ADDRESSE
0048  020C  85 04               STA END         ; (VON MNEENT)
0049  020E  AD 26 A4            LDA SAVPC+1
0050  0211  85 05               STA END+1
0051  0213  A0 00               LDY #0          ; PUFFERENDE KORRIGIEREN
0052  0215  B9 07 00    COR     LDA BYTES,Y
0053  0218  91 04               STA (END),Y
0054  021A  C4 0D               CPY TEMP        ; ALLES KOPIERT ?
```

```
0055  021C  F0 03        BEQ COR1
0056  021E  C8           INY              ; ..NEIN, NAECHSTES BYTE
0057  021F  D0 F4        BNE COR          ; IMMER
0058  0221  18     COR1  CLC
0059  0222  A5 04        LDA END          ; UPDATE PUFFERENDE
0060  0224  65 0D        ADC TEMP
0061  0226  85 04        STA END
0062  0228  90 02        BCC *+4
0063  022A  E6 05        INC END+1
0064  022C  4C 83 02     JMP STRT06       ; NAECHSTES KOMMANDO

0065  022F  A2 00  STRT0 LDX #0           ; DRUCKE 'OBJECT-EDITOR'
0066  0231  BD 3D 04     LDA TEXT,X
0067  0234  F0 06        BEQ STRT01       ; STRING-ENDE ZEICHEN
0068  0236  20 7A E9     JSR OUTPUT
0069  0239  E8           INX
0070  023A  D0 F5        BNE STRT0+2      ; UNBEDINGTER SPRUNG

0071  023C  20 A3 E7  STRT01 JSR FROM      ; RAM STARTADDRESSE
0072  023F  B0 FB        BCS STRT01       ; EINGABEFEHLER !
0073  0241  AD 1E A4     LDA CKSUMS       ; ZAHL EINGEGEBEN ?
0074  0244  D0 0D        BNE STRT02
0075  0246  AD 1C A4     LDA ADDR         ; JA, KOPIERE IN ARBEITSSP
0076  0249  85 00        STA START
0077  024B  AD 1D A4     LDA ADDR+1
0078  024E  85 01        STA START+1
0079  0250  4C 5A 02     JMP STRT03
0080  0253  A5 01  STRT02 LDA START+1     ; DRUCKE ALTE STARTADDR
0081  0255  A6 00        LDX START
0082  0257  20 42 EA     JSR WRITAZ
0083  025A  A5 00  STRT03 LDA START       ; SETZE START=CURRENT
0084  025C  85 02        STA CURREN
0085  025E  A5 01        LDA START+1
0086  0260  85 03        STA CURREN+1
0087  0262  20 3E E8     JSR BLANK
0088  0265  20 A7 E7  STRT04 JSR TO        ; RAM-ENDADDRESSE
0089  0268  B0 FB        BCS STRT04                        BLEIBT
0090  026A  AD 1E A4     LDA CKSUMS
0091  026D  D0 0D        BNE STRT05       ; WENN ALTE ENDADDRESSE
0092  026F  AD 1C A4     LDA ADDR         ; ADDRESSE IN ARBEITSSP
0093  0272  85 04        STA END          ; ...KOPIEREN
0094  0274  AD 1D A4     LDA ADDR+1
0095  0277  85 05        STA END+1
0096  0279  4C 83 02     JMP STRT06

0097  027C  A5 05  STRT05 LDA END+1       ; DRUCKE ALTE ENDADDRESSE
0098  027E  A6 04        LDX END
0099  0280  20 42 EA     JSR WRITAZ
0100  0283  20 F0 E9  STRT06 JSR CRLF
0101  0286  20 6E 04     JSR DISAS        ; DRUCKE ERSTE ZEILE
0102  0289  A2 FF  INP0  LDX ##FF         ; STACKPOINTER RUECKSETZEN
0103  028B  9A           TXS

0104  028C  E8           INX
0105  028D  86 06        STX FLAG         ; & FLAG LOESCHEN
0106  028F  20 95 02     JSR INP1
0107  0292  4C 89 02     JMP INP0
```

```
0108  0295  20 F0 E9   INP1   JSR CRLF
0109  0298  20 BC FE          JSR PATCH8      ; KOMMANDOEINGABE
0110  029B  A2 0A              LDX #10
0111  029D  DD 4D 04   LOOP1  CMP COMM,X      ; KOMMANDO GEFUNDEN ?
0112  02A0  F0 0C              BEQ FOUND
0113  02A2  CA                 DEX             ; NEIN, VERSUCHE
0114  02A3  10 F8              BPL LOOP1         NAECHSTES
0115  02A5  20 D4 E7   ERROR  JSR QM          ; KOMMANDO UNGUELTIG !!
0116  02A8  20 F0 E9          JSR CRLF
0117  02AB  4C 95 02          JMP INP1

0118  02AE  8A          FOUND  TXA             ; BERECHNUNG DER
0119  02AF  0A                 ASL A             KOMMANDOADRESSE
0120  02B0  AA                 TAX             ; KOMMANDOADDRESSE LADE
0121  02B1  BD 59 04           LDA JMPTBL+1,X    LADEN
0122  02B4  48                 PHA
0123  02B5  BD 58 04           LDA JMPTBL,X
0124  02B8  48                 PHA
0125  02B9  4C F0 E9          JMP CRLF        ; AUSFUEHRUNG MIT 'RTS'

0127  02BC              ;*** K - KILL ( ZEILENWEISE ) ***

0128  02BC  20 7F 04   KILL   JSR LINES       ; ANZAHL DER ZEILEN
0129  02BF  20 6E 04   LOOP02 JSR DISAS       ; DISASSEMBLIERE JEDE
0130  02C2  20 F0 E9          JSR CRLF          ZEILE
0131  02C5  A4 EA              LDY LENGTH      ; BEFEHL MIT 'NOP'
0132  02C7  A9 EA              LDA #$EA          UEBERSCHREIBEN
0133  02C9  91 02              STA (CURREN),Y
0134  02CB  88                 DEY
0135  02CC  10 FB              BPL *-3
0136  02CE  E6 EA              INC LENGTH
0137  02D0  A5 EA              LDA LENGTH      ; ZEIGER UM LENGTH
0138  02D2  20 9D 04           JSR ADD           ERHOEHEN
0139  02D5  20 92 04           JSR LOAD        ; CURREN --> SAVPC
0140  02D8  20 90 E7           JSR DONE        ; NAECHSTE ZEILE
0141  02DB  D0 E2              BNE LOOP02
0142  02DD  60                 RTS


0143  02DE              ;*** D - DISPLAY ONE LINE ***

0144  02DE  20 6E 04   DISPL  JSR DISAS       ; ZEILE DISASEMBLIEREN
0145  02E1  60                 RTS             ; FERTIG


0146  02E2              ;*** I - INSERT LINES ***

0147  02E2  A9 03      INSERT LDA #3           ; 3 BYTES FUER 'JSR'
0148  02E4  85 0D              STA TEMP          RESERVIEREN
0149  02E6  A9 00              LDA #0
0150  02E8  85 0E              STA TEMP+1
0151  02EA  20 6C F4   INS1   JSR DISASM
0152  02ED  20 F0 E9          JSR CRLF
0153  02F0  A0 00              LDY #0
0154  02F2  A6 0E              LDX TEMP+1
0155  02F4  B1 02      INS2   LDA (CURREN),Y ; ALTEN BEFEHL SICHERN
0156  02F6  95 07              STA BYTES,X
0157  02F8  E8                 INX
```

```
0158  02F9  C8                INY                          KOPIERT ?
0159  02FA  C4 EA             CPY LENGTH      ; BEFEHL VOLLSTAENDIG
0160  02FC  F0 F6             BEQ INS2        ; ..NEIN, WEITER
0161  02FE  90 F4             BCC INS2
0162  0300  E6 EA             INC LENGTH      ; 0,1,2 --> 1,2,3
0163  0302  A5 EA             LDA LENGTH
0164  0304  20 9D 04          JSR ADD         ; UPDATE PUFFERZEIGER
0165  0307  20 92 04          JSR LOAD
0166  030A  86 0E             STX TEMP+1
0167  030C  A5 0D             LDA TEMP        ; # DER BYTES
0168  030E  38                SEC               DEKREMENTIEREN
0169  030F  E5 EA             SBC LENGTH
0170  0311  85 0D             STA TEMP
0171  0313  F0 02             BEQ *+4
0172  0315  10 D3             BPL INS1        ; WENN < 3, WEITERLOESCHEN
0173  0317  A9 60             LDA #$60        ; 'RTS'-CODE
0174  0319  95 07             STA BYTES,X
0175  031B  86 0D             STX TEMP                     -BEFEHL
0176  031D  85 06             STA FLAG        ; KENNUNG FUER INSERT
0177  031F  E6 04             INC END         ; ENDEZEIGER ERHOEHEN
0178  0321  D0 02             BNE *+4
0179  0323  E6 05             INC END+1
0180  0325  38                SEC             ; PUFFERZEIGER WIEDERHERST
0181  0326  A5 02             LDA CURREN
0182  0328  E5 0D             SBC TEMP
0183  032A  85 02             STA CURREN
0184  032C  A5 03             LDA CURREN+1
0185  032E  E9 00             SBC #0

0186  0330  85 03             STA CURREN+1
0187  0332  A9 20             LDA #$20        ; 'JSR' EINFUEGEN
0188  0334  A0 00             LDY #0
0189  0336  91 02             STA (CURREN),Y
0190  0338  A5 04             LDA END
0191  033A  C8                INY
0192  033B  91 02             STA (CURREN),Y
0193  033D  A5 05             LDA END+1
0194  033F  C8                INY
0195  0340  91 02             STA (CURREN),Y
0196  0342  A9 EA             LDA #$EA        ; DEN REST MIT 'NOP'
0197  0344  C8                INY               BESETZEN
0198  0345  C4 0D       INS3  CPY TEMP
0199  0347  B0 05             BCS INS4        ; FERTIG
0200  0349  91 02             STA (CURREN),Y
0201  034B  C8                INY
0202  034C  D0 F7             BNE INS3        ; IMMER
0203  034E  A5 04       INS4  LDA END         ; ENDEZEIGER IN PC
0204  0350  8D 25 A4          STA SAVPC
0205  0353  A5 05             LDA END+1
0206  0355  8D 26 A4          STA SAVPC+1
0207  0358  4C 9E FB          JMP MNEENT      ; IN DIE MNEMO-EINGABE


0208  035B                ;*** L - LIST DISASSEMBLIERTEN CODE ***

0209  035B  20 92 04  LIST  JSR LOAD          ; ZEILENZEIGER KOPIEREN
0210  035E  20 17 E7        JSR $E717       ; DISASSEMBLIERUNG
0211  0361  AD 25 A4        LDA SAVPC       ; POINTER AUF NAECHSTEN
0212  0364  85 02           STA CURREN      ; ..BEFEHL SETZEN
```

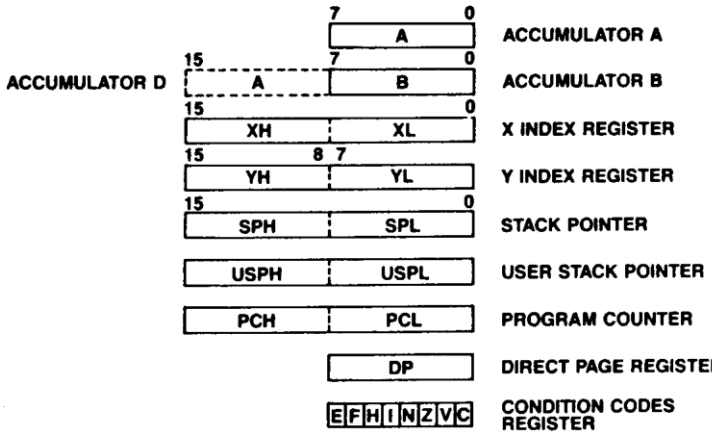
```
0213  0366  AD 26 A4        LDA SAVPC+1
0214  0369  85 03           STA CURREN+1
0215  036B  60              RTS             ; IN DIE BEFEHLSSCHLEIFE


0216  036C                  ;*** U - UP STELLE ZEIGER AUF NAECHSTES BYTE **

0217  036C  38        UP    SEC             ; CURREN = CURREN-1
0218  036D  A5 02           LDA CURREN
0219  036F  E9 01           SBC #01
0220  0371  85 02           STA CURREN
0221  0373  A5 03           LDA CURREN+1
0222  0375  E9 00           SBC #00
0223  0377  85 03           STA CURREN+1
0224  0379  20 6E 04        JSR DISAS       ; DISASSEMBLIERE DIE
0225  037C  60              RTS               ZEILE


0226  037D                  ;*** D - DOWN DISASSEMBLIERE NAECHSTES BYTE ***

0227  037D  A9 01     DOWN  LDA #01         ; CURREN = CURREN+1
0228  037F  20 9D 04        JSR ADD
0229  0382  20 6E 04        JSR DISAS       ; DISASSEMBLIERE DIE
0230  0385  60              RTS               ZEILE

0231  0386                  ;*** E - ENTER ASCII-STRING ***

0232  0386  A0 00     ENTER LDY #00
0233  0388  A9 27           LDA #$27        ; DRUCKE (')
0234  038A  48              PHA             ; SICHERN FUER SPAETER
0235  038B  20 7A E9        JSR OUTPUT
0236  038E  20 5F E9  ENT1  JSR RDRUB       ; EINGABE VOM KEYBOARD
0237  0391  C9 0D           CMP #CR         ; ENDEZEICHEN IST (CR)
0238  0393  F0 05           BEQ ENDENT
0239  0395  91 02           STA (CURREN),Y  ; ZEICHEN ABSPEICHERN
0240  0397  C8              INY             ; ZEIGER ERHOEHEN

0241  0398  D0 F4           BNE ENT1        ; UNBEDINGTER SPRUNG
0242  039A  C0 2C     ENDENT CPY #$2C       ; NICHT ERLAUBTER WERT ?
0243  039C  D0 01           BNE *+3
0244  039E  C8              INY             ; JA, KORREKTUR ERFORDERL
0245  039F  8C 19 A4        STY COUNT
0246  03A2  20 92 04        JSR LOAD        ; ZEIGER KOPIEREN
0247  03A5  68              PLA             ; DRUCKE ZWEITES (')
0248  03A6  20 7A E9        JSR OUTPUT
0249  03A9  20 F0 E9        JSR CRLF
0250  03AC  98              TYA
0251  03AD  20 9D 04        JSR ADD         ; CURREN = CURREN+ASCI
0252  03B0  20 23 E7        JSR JD1         ; DISASSEMBLIERE STRING
0253  03B3  60              RTS


0254  03B4                  ;*** T - TOP SETZE ZEIGER AUF RAM ANFANG ***

0255  03B4  A5 00     TOP   LDA START       ; ZEIGER AUF RAM-START
0256  03B6  85 02           STA CURREN
0257  03B8  A5 01           LDA START+1
0258  03BA  85 03           STA CURREN+1
```

```
0259  03BC  20 6E 04        JSR DISAS       ; DISASSEMBLIERE DIE ZEILE
0260  03BF  60              RTS


0261  03C0                 ;*** B - BUTTOM SETZE ZEIGER AUF LETZTES BYTE ***

0262  03C0  A5 04        BUTTOM LDA END        ; ZEIGER AUF RAM-ENDE SETZEN
0263  03C2  85 02               STA CURREN
0264  03C4  A5 05               LDA END+1
0265  03C6  85 03               STA CURREN+1
0266  03C8  20 6E 04           JSR DISAS       ; DISASSEMBLIERE DIE ZEILE
0267  03CB  60                 RTS


0268  03CC                 ;*** F - FIND FINDE ALLE BEFEHLE DIE GLEICH DER *>
0269  03CC                 ;*** EINGABE SIND ***

0270  03CC  A9 FF         FIND   LDA ##FF       ; SETZE FIND-FLAG
0271  03CE  85 06                STA FLAG                          -PUFFER
0272  03D0  20 74 04             JSR LDB       ; INITIALISIERE VERGLEICHS
0273  03D3  4C 9E FB             JMP MNEENT    ; IN DIE MNEMONISCHE EINGABE
                                                  EINGABE
0274  03D6                 ;--- EINTRITT NACH DER EINGABE ---

0275  03D6  20 74 04      FND1   JSR LDB       ; EINGABEPUFFER INITIALIS
0276  03D9  20 7F 04             JSR LINES     ; ANZAHL DER ZEILEN
0277  03DC  CE 19 A4             DEC COUNT
0278  03DF  20 6C F4             JSR DISASM    ; SUCHKRITERIUM AUSGEBEN
0279  03E2  20 F0 E9             JSR CRLF
0280  03E5  A0 00         FND02  LDY #00
0281  03E7  B1 02         FND12  LDA (CURREN),Y ; NAECHSTES ZEICHEN LADEN
0282  03E9  D9 07 00             CMP BYTES,Y   ; GLEICH SUCHKRITERIUM ?
0283  03EC  F0 1F                BEQ FND3      ; JA
0284  03EE  E6 02         FND13  INC CURREN    ; ZEIGER ERHOEHEN
0285  03F0  D0 02                BNE *+4
0286  03F2  E6 03                INC CURREN+1
0287  03F4  A5 02                LDA CURREN    ; ENDE ERREICHT ?
0288  03F6  C5 04                CMP END
0289  03F8  D0 EB                BNE FND02     ; NEIN, WEITER
0290  03FA  A5 03                LDA CURREN+1
0291  03FC  C5 05                CMP END+1
0292  03FE  D0 E5                BNE FND02
0293  0400  AD 25 A4      FND5   LDA SAVPC     ; ZEIGER AUF LETZTES..
0294  0403  85 02                STA CURREN    ; ..GEFUNDENES WORT
0295  0405  AD 26 A4             LDA SAVPC+1
0296  0408  85 03                STA CURREN+1
0297  040A  4C 89 02             JMP INP0      ; IN DIE KOMMANDOSCHLEIFE

0298  040D  C4 EA         FND3   CPY LENGTH    ; ALLE BYTES GLEICH ?
0299  040F  F0 03                BEQ FND4      ; JA, STRING GEFUNDEN
0300  0411  C8                   INY           ; VERGLEICHE NAECHSTES BYTE
0301  0412  D0 D3                BNE FND12     ; IMMER
0302  0414  20 6E 04      FND4   JSR DISAS     ; DISASSEMBLIERE GEFUNDENE
0303  0417  20 90 E7             JSR DONE      ; ALLE GEFUNDEN ?          ZEILE
0304  041A  F0 E4                BEQ FND5      ; JA, ABBRUCH
0305  041C  20 07 E9             JSR RCHEK
0306  041F  20 F0 E9             JSR CRLF
0307  0422  4C EE 03             JMP FND13     ; NAECHSTES BYTE
```

```
0308  0425                ;*** * - SETZE PROGRAMMZAEHLER ***

0309  0425  20 AE EA   PCIN    JSR ADDIN       ; ADDRESSE EINGEBEN
0310  0428  B0 54             BCS ADERR       ; EINGABEFEHLER !
0311  042A  20 F0 E9          JSR CRLF
0312  042D  AD 1C A4          LDA ADDR        ; EINGABE = CURREN
0313  0430  85 02              STA CURREN
0314  0432  AD 1D A4          LDA ADDR+1
0315  0435  85 03              STA CURREN+1
0316  0437  20 92 04          JSR LOAD        ; SYSTEM-ZAEHLER LADEN
0317  043A  4C 6C F4          JMP DISASM      ; ZEILE DISASSEMBLIEREN
0319  043D  0D             TEXT    .BYT CR,'OBJECT-EDITOR',CR,00

0320  044D  4B 20 49   COMM    .BYT`'K ILUDETBF*'

0321  0458  BB 02       JMPTBL .WOR KILL-1
0322  045A  DD 02              .WOR DISPL-1
0323  045C  E1 02              .WOR INSERT-1
0324  045E  5A 03              .WOR LIST-1
0325  0460  6B 03              .WOR UP-1
0326  0462  7C 03              .WOR DOWN-1
0327  0464  85 03              .WOR ENTER-1
0328  0466  B3 03              .WOR TOP-1
0329  0468  BF 03              .WOR BUTTOM-1
0330  046A  CB 03              .WOR FIND-1
0331  046C  24 04              .WOR PCIN-1


0332  046E  20 92 04   DISAS   JSR LOAD        ; ZEIGER KOPIEREN
0333  0471  4C 6C F4          JMP DISASM      ; ZEILE DISASSEMBLIEREN

0334  0474  A9 07       LDB     LDA #<BYTES     ; INITIALISIERT SUCHSTRING
0335  0476  8D 25 A4           STA SAVPC
0336  0479  A9 00              LDA #>BYTES
0337  047B  8D 26 A4           STA SAVPC+1
0338  047E  60         ADERR   RTS

0339  047F  20 37 E8   LINES   JSR PSL1        ; DRUCKT '/'
0340  0482  20 85 E7          JSR GCNT        ; EINGABE EINER ZAHL < 99
0341  0485  AD 19 A4          LDA COUNT       ; PRUEFUNG, OB ZAHL GUELTIG
0342  0488  C9 2C              CMP #$2C
0343  048A  F0 03              BEQ ERR1        ; ZAHL UNGUELTIG !!
0344  048C  4C F0 E9           JMP CRLF
0345  048F  4C A5 02   ERR1    JMP ERROR

0346  0492  A5 02       LOAD    LDA CURREN      ; KOPIERT CURREN NACH SAVPC
0347  0494  8D 25 A4           STA SAVPC
0348  0497  A5 03              LDA CURREN+1
0349  0499  8D 26 A4           STA SAVPC+1
0350  049C  60                 RTS

0351  049D  18         ADD     CLC             ; ADDIERT ACCU UND CURREN
0352  049E  65 02               ADC CURREN
0353  04A0  85 02               STA CURREN
0354  04A2  90 02               BCC *+4
0355  04A4  E6 03               INC CURREN+1
0356  04A6  60                 RTS
                        .END
```

# MC 6809: Befehle (2)

Im Abschnitt 'Befehle (1)' in Heft 19 wurden die Befehle für die 8 Bit breiten Register Akku A und Akku B besprochen und auch die Befehle für den 16 Bit breiten Akku D.




**16-Bit-Befehle für die Indexregister X, Y, S und U**

Außer dem Programmzähler hat der 6809 noch folgende 16 Bit breiten Register: Die Indexregister X und Y sowie den Hardware-Stackpointer S und den User-Stackpointer U (Anwender-Stackpointer). Wie wir bei den Adressierungsarten noch sehen werden, können alle diese Register, also auch die Stackpointer S und U, zur direkten und indirekten Indizierung der meisten Befehle benutzt werden. - Für die Register selbst stehen folgende Befehle zur Verfügung, die jeweils 2 Bytes als Operanden haben:

| | | | | |
|---|---|---|---|---|
| LDX | LDY | LDS | LDU | Lade Register mit 16-Bit-Operanden |
| STX | STY | STS | STU | Abspeichern aus Register, 16-Bit-Operand |
| CMPX | CMPY | CMPS | CMPU | Vergleiche Register mit 16-Bit-Operand. |
| | Adressierungsarten: | | | Immediate (Direktoperand), außer beim Abspeichern<br>Direct Page<br>Extended (absolute Adressierung)<br>Indiziert |

Es gibt weitere Ladebefehle für diese Register, nämlich

| | | | | |
|---|---|---|---|---|
| LEAX | LEAY | LEAS | LEAU | Lade Register mit einer effektiven Adresse. |
| | Adressierungsart: | | | Indiziert |

Diese Befehle benutzen die weitgreifenden Indizierungsmöglichkeiten des 6809. Die Möglichkeit der Indizierung bedeutet ja, daß die CPU effektive Adressen berechnen kann. Während nun bei den übrigen indizierbaren Befehlen diese berechnete Adresse auf den Adreßbus ausgesandt wird, um den Operanden an seinem Speicherplatz zu erreichen, wird bei den Befehlen LEAX usw. diese berechnete Adresse nur in das angesprochene Indexregister übertragen, wo sie für Folgebefehle als eine Adreßbasis weiterbenutzt werden kann. Die hier liegenden Programmierungsmöglichkeiten werden in späteren Abschnitten erläutert. Es ist aber schon jetzt auf Besonderheiten hinzuweisen, die sich aus der Architektur des 6809 ergeben:

Manche Befehle haben einen Operationscode von 2-3 Bytes, für die dann auch eine entsprechend längere Ausführungszeit benötigt wird. Das sog. POSTBYTE (Folgebyte) wird in indizierten Befehlen verwendet. Es bestimmt die Art der Indizierung.

Außerdem arbeitet der 6809 bei manchen Befehlen mit einem PREBYTE, einem vorauslaufenden Byte. Es bewirkt, wenn man so will, die Umschaltung der Befehlsinterpretation. Ein solches Prebyte wird für weniger häufig benutzte Befehle eingesetzt, nämlich für die 'long branches' (weite Verzweigungen), für die Software-Interrupts SWI2 und SWI3 und für die in diesem Abschnitt erwähnten Befehle LDY, STY, CMPY, LDS, STS, CMPS sowie LDU, STU, CMPU, nicht jedoch für LDX, STX und CMPX.

Das Indexregister X hat also hinsichtlich Befehlslänge und Ausführungszeit eine bevorzugte Stellung. Hinsichtlich ihres Befehlsvorrates (Laden, Speichern, Vergleichen) und der mit ihnen möglichen Indizierungen sind die Register X, Y, S und U jedoch gleichwertig, wenn man vom bereits erwähnten Befehl ABX absieht: Addiere Inhalt von Akku B vorzeichenfrei zu X.

Natürlich hat der 6809 spezielle Stackbefehle, die den aktuellen Stand der Stackpointer H und U für Zwecke der Registerablage und -wiederherstellung benutzen.

**Tausch und Übertragung zwischen Maschinenregistern**

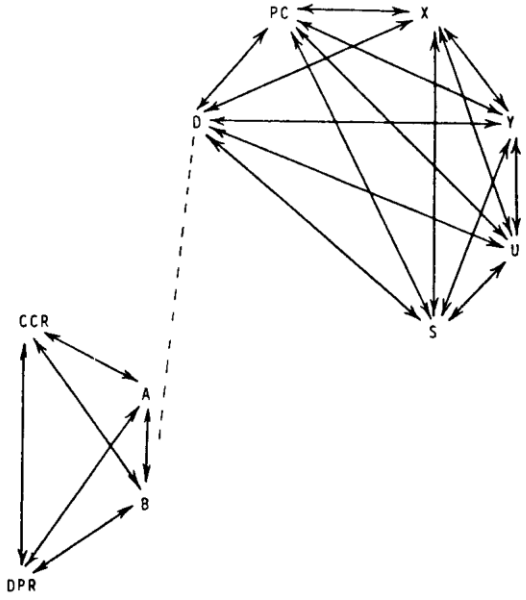
Zwischen jeweils gleich breiten Registern (8 bzw. 16 Bit) ist ein gegenseitiger Austausch mit dem Befehl

EXG (Exchange Registers) möglich.

Der Befehl

TFR (Transfer Register to Register) legt eine Kopie aus dem Quellregister in das Zielregister ab.

Für die Befehle EXG und TFR bestehen damit folgende Austausch- und Transfermöglichkeiten:



Die Adressierungsart für EXG und TFR ist 'inherent'. Auf den Opcode folgt ein Direktoperand (Postbyte), der die anzusprechenden Register in je einem Halbbyte bezeichnet:

```
0000 = D              1000 = A
0001 = X              1001 = B
0010 = Y              1010 = CCR
0011 = U              1011 = DP
0100 = S              1100 = undefiniert
0101 = PC             1101 = undefiniert
0110 = undefiniert    1110 = undefiniert
0111 = undefiniert    1111 = undefiniert
```

Beim Befehl TFR bezeichnet das vordere Halbbyte das Quellregister, das hintere das Zielregister:

| Direktoperand des TFR | Quell- | Zielregister |
|---|---|---|

Aus den vorstehenden Darstellungen geht hervor, daß es für den Registertausch bzw. für die Übertragung der Registerinhalte keine Beschränkungen gibt (außer zu beachtender gleicher Breite). Insbesondere wäre es also möglich, durch einen Befehl z.B. TFR X PC das Programm an einer Stelle fortzusetzen, deren Adresse im Register X enthalten ist. Das entspräche einem unbedingten Sprungbefehl (JMP).

Für das Direct Page Register (DP) gibt es keinen besonderen Ladebefehl. Beim Einschalten des Prozessors wird es zwar zu 00 initialisiert. Wenn man seinen Inhalt später ändern möchte, wird man es aus Akku A oder B mit TFR beschreiben. Will man nur vorübergehend eine Speicherseite zur Direct Page machen, so bietet sich der Befehl EXG für das DP an.

Mit dem Befehl TFR läßt sich z.B. auch der Prozessorstatus (CC-Register) für eine spätere Verwendung retten, wenn man nicht von der Möglichkeit der Ablage auf dem Stack Gebrauch machen kann oder will.

**Stackbefehle**

Der Hardware-STack S und der Anwender-STack U wurden bereits in ihrer Eigenschaft als Indexregister angesprochen. Beide Stacks müssen vom Programmierer initialisiert werden. Das geschieht im allgemeinen durch die Befehle LDS #Adresse bzw. LDU #Adresse. Als 'Adresse' darf dabei eine beliebige (auch 'krumme') vorhandene RAM-Adresse gewählt werden. Die Stacks initialisiert man für Bereiche, in denen sie andere Programmteile und sich selbst nicht beeinflussen können. Ein Stackbereich darf im praktischen Einsatz beliebig groß werden und auch überlappend in Speicherseiten liegen (anders beim 6502: Der Stack liegt immer in Speicherseite 01 und wird beim RESET zu 01FF initialisiert).

Und noch ein Hinweis für 6502-Programmierer: Die Stackpointer weisen beim 6809 immer auf die letzte benutzte Speicherstelle hin. M.a.W.: Die Stackpointer werden vor einer Ablage zunächst erniedrigt, bzw. nach dem Entstacken erhöht.

Die Stackbefehle sind:

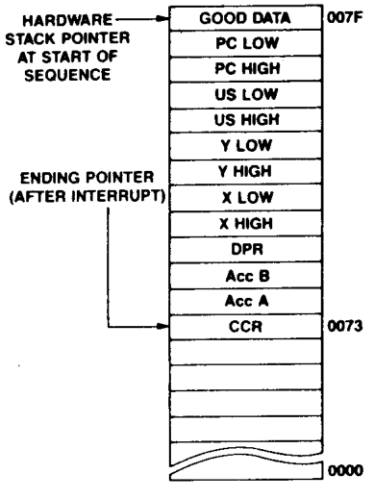
| | |
|---|---|
| PSHS | Push Registers on the Hardware Stack, Registerablage auf dem Hardware-Stack |
| PULS | Pull Registers from the Hardware Stack, Register entstacken |
| PSHU | Push Registers on the User Stack, Registerablage auf den Anwender-Stack |
| PULU | Pull Registers from the User Stack, Register entstacken |

Adressierungsart: Memory mit Registerliste im Direktoperand.

Der Direktoprand ist das zweite Byte im Befehl. Man kann ihn als 'binäre Strichliste' oder Registerliste bezeichnen, jede Bitposition kennzeichnet ein Maschinenregister, das ge- oder entstackt werden soll. M.a.W.: Mit einem einzigen Befehl können 0 bis 8 Register transportiert werden. Erwähnenswert ist weiterhin die Automatik, derzufolge der Stackpointer für 8 Bit breite Register um 1 und für 16 Bit breite Register um 2 verändert wird. Die Direktoperanden haben folgende Zuweisung der Register zu den Bitpositionen:

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Bitposition |
|---|---|---|---|---|---|---|---|---|
| PC | U | Y | X | DP | B | A | CC | Direktoperand Hardware-Stack |
| PC | S | Y | X | DP | B | A | CC | Direktoperand Anwender-Stack |

Bei dieser Zuweisung ist es nur logisch, daß der Hardware-Stackpointer nicht auf den Hardware-Stack abgelegt werden kann (wozu auch?) und der Anwender-Stackpointer nicht auf den Anwender-Stack. - Die Ablage der Register auf den Stacks und die Entnahme von dort werden immer in einer festgelegten Reihenfolge vollzogen:




Diese geordnete Ablage zusammen mit der bequemen Notierung der zu stackenden/entstackenden Registerliste macht die Rettung von Registern und die Stackverwaltung einfach und sicher.

Der Hardware-Stack S wird im Falle eines **Unterprogrammaufrufes** automatisch (und wie gewohnt) von der CPU mit der Rückkehradresse des Programmzählers beschickt. Der **schnelle Interrupt** FIRQ führt zur Rettung des Programmzählers und des Status (der FIRQ setzt u.a. zum Zeichen dessen das FIRQ-Flag). **Alle übrigen Interrupte** retten sämtliche 8 Registerinhalte (außer dem Hardware-Stackpointer selbst) auf den Hardware-Stack und setzen dafür das Entire-Flag (entire= gesamt). Der Anwender-Stack bleibt von diesen Ereignissen also vollkommen unberührt und er eignet damit hervorragend zur Parameterübergabe. Zur Verdeutlichung folgende Schemata:

Wenn man beim 6502 eine Parameterübergabe über den Stack vollziehen wollte, dann mußte man nach einem JSR-Befehl 'hinter' die Rückkehradresse greifen, wenn man an die Parameter heran wollte. Beim 6809 sind solche Kunstgriffe nicht notwendig, wenn man den Anwender-Stack beschickt. Und in vielen Fällen wird man Parameter gar nicht auf den Stack transportieren müssen, man setzt stattdessen den Stackpointer U auf den Anfang der Parameterliste, wenn die Parameter hintereinander aufgereiht sind.

Strukturierte Programmiersprachen sind oft 'stackorientiert' mit Verwendung 'lokaler Variablen', Variablen also, die nur in einer als Unterprogramm aufgerufenen Prozedur oder in einem Block erreichbar und ansprechbar sind und die mit dem Verlassen der Prozedur verloren gehen. Der 6809 ist für diese Sprachen insofern vorbereitet, als man nach einem Unterprogrammaufruf (der die Rückkehradresse auf dem Hardware-Stack sichert) den Hardware-Stackpointer mit dem LEAS-Be-

fehl um die Menge Bytes herunterzieht, die man für die lokalen Variablen oder den Arbeitsspeicher benötigt. Durch entsprechendes Umladen bewegt man sich in diesem Bereich dann mit dem Anwender-Stackpointer. Weitere Prozedur-(Unterprogramm-)aufrufe oder Interrupts werden dann den Variablenbereich nicht zerstören, weil der Hardware-Stackpointer in sichere Entfernung gebracht wurde. Wir kommen darauf zurück.

**Befehle zur Beeinflussung des Prozessorstatus**

Vom 6502 kennen wir verschiedene Befehle, die im Prozessorstatus Flags setzen bzw. löschen, z.B. SEC, CLC (Set Carry, Clear Carry Flag). Der 6809 hat keine entsprechenden Befehle, stattdessen

ORCC der Direktoperand dieses Befehles, das Postbyte, bestimmt mit seinen 1-er Bitpositionen, welche Bits (Flags) im Prozessorstatus logisch hinzugeODERt werden sollen.

ANDCC Der Direktoperand bestimmt mit seinen 0-er Bits, welche Flags des Status mit logisch AND zu 0 gelöscht werden.

Adressierungsart: Immediate durch das Postbyte.

Der Direktoperand stellt wieder eine binäre 'Strichliste' für die entsprechenden Positionen des CC-Registers dar. Der Programmierer überläßt die Herstellung der entsprechenden Maske seinem hoffentlich entsprechend leistungsfähigen Assembler.

**Verzweigungsbefehle**

Der 6809 kennt zwei Arten von Verzweigungsbefehlen, die 'short Branches' wie beim 6502 oder 6800 mit +127 bzw. −128 Sprungweite relativ zum aktuellen Befehlszählerstand, und die 'long branches' mit beliebiger, den ganzen Speicher überdeckender Sprungweite. Bei der ersten Art hat der Direktoperand 1 Byte, bei der zweiten hat er 2 Bytes, den wir uns wieder als vorzeichenbehaftete Zahl vorzustellen haben. Die Berechnung der Direktoperanden überläßt man wiederum seinem Assembler.

Folgende Erklärungen sind noch notwendig: Kurze Verzweigungen haben 2 Bytes (Operationscode plus Offset). Die weniger häufig benutzten 'long branches' haben (mit 2 Ausnahmen) ein Prebyte (hex 10) zur Umschlatung der Befehlsinterpretation, Darauf folgt der Operationscode der Verzweigung und schließlich der Offset in 2 Bytes. Long branches zählen mithin 4 Byte.

Die erwähnten Ausnahmen betreffen die häufiger benutzten Befehle LBRA (Long Branch Always, unbedingte Verzweigung) und LBSR (Long Branch to Subroutine, Unterprogrammverzweigung), die ohne Prebyte implementiert sind und mithin etwas schneller ausführen. Die weiten Verzweigungen erlauben die speicherplatzunabhängige Adressierung von Zielen relativ zum Programmzählerstand. Wie wir bei den indizierten Befehlen noch sehen werden, können auch Operanden relativ zum Programmzählerstand adressiert werden, so daß auch von dieser Seite her vollverschieblicher Code Programmiert werden kann.

Im 6809 sind die Verzweigungsbedingungen gegenüber den älteren Prozessoren erheblich vermehrt worden. Man kommt daher nach Vergleichen Register gegen Operand stets mit einem einzigen Verzweigungsbefehl aus egal, ob die Operanden als vorzeichenfreie oder vorzeichenbehaftete Zahlen von 8 oder 16 Bit zu interpretieren sind. Wir rufen in diesem Zusammenhang in Erinnerung,

daß nach Subtraktionen oder Vergleichen (die ja als nur 'gedachte' Subtraktion den Status setzen) das Carry-Flag wie folgt beeinflußt wird:

Carry = 0 Es wurde nicht geborgt
Carry = 1 Es wurde geborgt.

Nachstehend zunächst die unbedingten Verzweigungsbefehle.

| | | |
|---|---|---|
| BRA | LBRA | Branch Always, unbedingte Verzweigung |
| BRN | LBRN | Branch Never, unbedingte Nicht-Verzweigung |
| BSR | LBSR | Branch to Subroutine, unbedingte Verzweigung zu einem Unterprogramm. Die Rückkehradresse wird auf den Hardware-Stack gerettet. |

Zu der folgenden Aufstellung der bedingten Verzweigungen ist zu bemerken: Die Gruppierung erfolgte nach Anwendungsgebieten, dadurch sind einige Widerholungen enthalten. Für alle genannten Befehle gibt es auch 'long branches', die mnemonisch mit einem vorgestellten 'L' abgekürzt werden z.B. LBPL (Long Branch on Plus). Die Aufstellung zeigt auch, in Abhängigkeit von welchem Statusflag verzweigt wird. Bei den Verzweigungen für vorzeichenbehaftete Zahlen werden bool'sche Produkte mehrerer Flags ausgewertet.

**Bedingte Verzweigungen**

**Einfache Verzweigungsbefehle**

| MNEMONIC | BRANCHES IF | |
|---|---|---|
| BPL | N = 0 | Plus |
| BMI | N = 1 | Minus |
| BNE | Z = 0 | Not Equal |
| BEQ | Z = 1 | Equal |
| BVC | V = 0 | Overflow Clear |
| BVS | V = 1 | Overflow Set |
| BCC/BHS | C = 0 | Carry Clear/Higher or Same |
| BCS/BLO | C = 1 | Carry Set/Lower |

**Verzweigungsbefehle für vorzeichenfreie Zahlen**

| MNEMONIC CODE | BRANCH CONDITION ALGEBRAIC | BOOLEAN | |
|---|---|---|---|
| BLO/BCS | (ACCX) < (M) | C = 1 | Lower/Carry Set |
| BLS | (ACCX) ≤ (M) | C + Z = 1 | Less or Same |
| BHS/BCC | (ACCX) ≥ (M) | C = 0 | Higher or Same/Carry Clear |
| BHI | (ACCX) > (M) | C + Z = 0 | Higher |
| BEQ | (ACCX) = (M) | Z = 1 | Euql |
| BNE | (ACCX) ≠ (M) | Z = 0 | Not Equal |

**Verzweigungsbefehle für vorzeichenbehaftete Zahlen**

| MNEMONIC CODE | BRANCH CONDITION ALGEBRAIC | BOOLEAN | |
|---|---|---|---|
| BLT | (ACCX) < (M) | N ⊕ V = 1 | Less than |
| BLE | (ACCX) ≤ (M) | Z + [N ⊕ V] = 1 | Less than or Equal |
| BGE | (ACCX) ≥ (M) | N ⊕ V = 0 | Greater or Equal |
| BGT | (ACCX) > (M) | Z + [N ⊕ V] = 0 | Greater than |
| BEQ | (ACCX) = (M) | Z = 1 | Equal |
| BNE | (ACCX) ≠ (M) | Z = 0 | Not Equal |

**Sprungbefehle, Unterprogramme, Rückkehr**

Wie wir gesehen haben, lassen sich Programme voll verschieblich schreiben, wenn man statt absoluter Sprünge relative Verzweigungen, ggfs. auch als 'long branches' benutzt (BRA, LBRA). Das Gleiche gilt für Unterprogrammaufrufe (BSR, LBSR). Gleichwohl hat der 6809 auch Sprung- und Unterprogrammbefehle, nämlich:

JMP Jump to Effective Address
JSR Jump to Subroutine at Effective Address.

Adressierungssarten: Direct Page - Extended - Indiziert.

Diese Befehle kommen vor allem dann zum Einsatz, wenn man Programmteile aufruft, die immer am gleichen Platz residieren (Monitor, Firmware). - Die Möglichkeit der direkten und indirekten Indizierung der Befehle erlaubt die Benutzung von 'Sprungleisten' (vergleichbar in BASIC: ON X= 0, 2, 4, ... GOTO oder GOSUB).

Für den Rücksprung aus Unterprogrammen, die mit BSR oder JSR angesprungen wurden, gibt es den üblichen Befehl:

RTS Return from Subroutine

Auch der Befehl

RTI Return from Interrupt wirkt erwartungsgemäß. Er stellt den Status und Programmzähler wieder her, wenn das Entire-Flag 0 war (nach einem FIRQ). RTI stellt alle Register wieder her für die übrigen Interrupts (wenn das E-Flag 1 war).

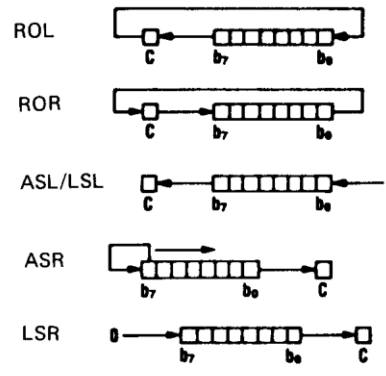
**Befehle, die direkt auf Speicherzellen wirken**

Diese Befehle beeinflussen außer dem Status kein weiteres Maschinenregister. In der 65er Familie werden sie treffend mit 'read/modify/write' bezeichnet. Der Befehlssatz entspricht genau der ersten für die Akkumulatoren genannten Befehlsgruppe (Heft 19, Seite 22):

NEG Bildung des 2er-Komplementes (der negativen Zahl)
COM Bildung des 1er-Komplementes
CLR Setze zu 00
INC Inkrementiere um 1
DEC Dekrementiere um 1
ROL Zirkularverschiebung nach links unter Einschluß des Carry Flag
ROR Zirkularverschiebung nach rechts unter Einschluß des Carry Flag
ASR Arithmetische Rechtsverschiebung unter Erhaltung des Vorzeichenbits
LSL Logische Linksverschiebung
ASL Arithmetische Linksverschiebung

LSR Logische Rechtsverschiebung
TST Test auf Null und Vorzeichen

Adressierungsarten:
Direct Page
Extended (absolut)
Indiziert




Wirkung der Verschiebebefehle

**Verschiedene Befehle**

NOP No Operation
SWI Software-Interrupt (Break), vektoriert über FFFA/FFFB
SWI2 Software-Interrupt mit Prebyte hex 10, vektoriert über FFF4/FFF5
SWI3 Software-Interrupt mit Prebyte hex 11, vektoriert über FFF2/FFF3

SYNC Synchronize to External Event. Dieser Befehl hält den Prozessor an. Adreß- und Datenbus werden hochohmig gefahren (tri-state). Wenn jetzt ein Interrupt eintrifft beendet der Prozessor den Wartestand und schreitet wie folgt fort:

Wenn der Interrupt maskiert war oder wenn das Interruptsignal weniger als 3 Zyklen ansteht, fährt die MPU mit der nächsten Instruktion fort.

Wenn Interrupt zugelassen war und wenn sein Signal 3 und mehr Zyklen ansteht, tritt der Prozessor in die Interruptbearbeitung ein und stackt zunächst die Register.

Der SYNC-Befehl ist für Datentransfers hoher Geschwindigkeit vorgesehen, bei denen eine DMA-fähige Einheit oder ein I/O-Baustein ohne Interruptbetrieb bedient werden.

CWAI Clear and Wait for Interrupt. Das Postbyte/Direktoperand ANDed den Prozessorstatus, um den zugelassenen Interrupt zu definieren: =FF kein Interrupt, =EF IRQ zugelassen, =BF FIRQ zugelassen, =AF beide zugelassen. Der gesamte Registersatz wird schon vor dem Interruptereignis gestackt (Entire-Flag=1). Wenn danach ein nicht maskierter Interrupt eintrifft, werden keine weiteren Register gestackt, stattdessen wird sofort auf den Interrupthandler vektoriert. Wenn der Interrupt ein FIRQ war, wird mit RTI gleichwohl vollkommen entstackt, weil des E-Flag 1 war.

Die nachfolgenden Aufstellungen geben einen Gesamtüberblick über den Befehlssatz.

**6809-Befehle nach Funktionsgruppen und beteiligten Einheiten**

Bis auf die wenigen gekennzeichneten Befehle beeinflussen alle den Prozessorstatus CC

| Memory | Akku A | Akku B | Akku D | Indexregister X | Y | S | U | Status | Funktionsgruppe |
|---|---|---|---|---|---|---|---|---|---|
| x | LDA | LDB | LDD | LDX | LDY | LDS | LDU | | Laden und Speichern |
| CLR | \|CLRA | \|CLRB | | | | | | | |
| | | | | LEAX | LEAY | LEAS*) | LEAU*) | | (nur indizierte Adressierung) |
| | | | | | | | | ORCC | |
| | | | | | | | | ANDCC | |
| x | STA | STB | STD | STX | STY | STS | STU | | |
| x | ADDA | ADDB | ADDD | | | | | | Arithmetik |
| x | ADCA | ADCB | | | | | | | |
| x | SUBA | SUBB | SUBD | | | | | | |
| x | SBCA | SBCB | | | | | | | |
| | \|DAA | | | | | | | | |
| | \|MUL | | | | | | | | |
| | \|SEX | x | | | | | | | |
| | | x | | \|ABX | | | | | (verändert den Status nicht) |
| x | CMPA | CMPB | CMPD | CMPX | CMPY | CMPS | CMPU | | Vergleich und Test |
| TST | \|TSTA | \|TSTB | | | | | | | |
| | BITA | BITB | | | | | | | |
| | ANDA | ANDB | | | | | | | Logische Befehle |
| | ORA | ORB | | | | | | | |
| | EORA | EORB | | | | | | | |
| NEG | \|NEGA | \|NEGB | | | | | | | Datenveränderung |
| COM | \|COMA | \|COMB | | | | | | | |
| INC | \|INCA | \|INCB | | | | | | | |
| DEC | \|DECA | \|DECB | | | | | | | |
| LSR | \|LSRA | \|LSRB | | | | | | | Verschiebebefehle |
| ASR | \|ASRA | \|ASRB | | | | | | | |
| ROR | \|RORA | \|RORB | | | | | | | |
| ASL | \|ASLA | \|ASLB | | | | | | | |
| LSL | \|LSLA | \|LSLB | | | | | | | |
| ROL | \|ROLA | \|ROLB | | | | | | | |

Mit | gekennzeichnete Befehle:
Adressierungsart Inherent bzw. Akku

| | | | | | | | | | Adressierungsarten |
|---|---|---|---|---|---|---|---|---|---|
| | x | x | x | x | x | x | x | x | Immediate (Direktoperand) |
| x | x | x | x | x | x | x | x | | Direct Page |
| x | x | x | x | x | x | x | x | | Extended (absolut) |
| x | x | x | x | x | x | x | x | | Indiziert |

*) ohne Statusbeeinflussung

**Registertausch - Registertransfer**

EXG TFR — Adressierungsart: Inherent mit Direktoperand

**Stackbefehle**

PSHS - PULS PSHU - PULU — Adressierungsart: Memory mit Direktoperand

**Verzweigungsbefehle (alle Befehle auch als 'long branch')**

BRA - BRN BSR — Adressierungsart: Relativ zum Programmzähler
BCC - BCS
BVC - BVS
BNE - BEQ
BPL - BMI
BHI - BLS
BHS - BLO
BGT - BLE
BGE - BLT

Diverse grafische Darstellungen wurden der Motorola-Dokumentation zum 6809 entnommen.

**Sprung- und Rückkehrbefehle**

JMP JSR — Adressierungsarten: Direct Page, Extended, indiziert
RTS RTI — Inherent

**Verschiedene Befehle**

NOP — Adressierungsart: Inherent
SWI SWI2 SWI3 — Absolut indirekt
SYNC
CWAI — Adressierungsart: Memory Immediate Ω

Wird fortgesetzt. R. L.

**Berichtigung: FORTH-Disassembler in Heft 19, Seite 24**

Für das einwandfreie Arbeiten des Disassemblers sind im BASIC-Programm für CBM die Zeilen 87 und 90 in der nachstehenden Formulierung zu verwenden. Geringfügig wurde auch in den Zeilen 25, 70-75, 83 und 84 sowie in 330 und 550 abgeändert.

```
25 GOSUB840:PC=07-CX

70 A=CODE:GOSUB1330:CODE$=A$:WO$=""
71 IFCODE$="B756"THENWO$="TCALL"
72 IFCODE$="B7DE"THENWO$="USER "
73 IFCODE$="B7AE"THENWO$="VARIABLE"
74 IFCODE$="B792"THENWO$="CONSTANT"
75 PRINT#4,PR;N;"  ";NAME$;SPC(16-N);LINK$;"  ";CODE$;"  ";WO$


83 A=PC+N+5+CX:GOSUB 1330
84 PRINT#4,A$;" : ";
85 FORI=PC+N+5TOPX-1
87 J=J+1:IFJ=32THENA=I+CX:GOSUB1330:PRINT#4:PRINT#4,A$;" : ";:J=0
90 B=PEEK(I):GOSUB1240


330 A=0+CX

550 A=0+2+BR+CX
```

# Software-Besprechung

**Das SM-Kit wurde als Profiwerkzeug für Programmierer der Commodore-Rechner** vom Softwareverbund Micro Computer GmbH in München entwickelt.Es wird seit einiger Zeit auch beim Autor eingesetzt und darf als außerordentlich nützliche und angenehme Programmierhilfe bezeichnet werden. Es führt das bisher nur sehr selten verliehene Qualitätssiegel 'von Commodore empfohlen' und erobert mit 50% seines Absatzes sogar den US-Markt. Für Rechner mit BASIC 4 erhält der Benutzer ein maskenprogrammiertes ROM mit 4K ab Adresse hex 9000, für BASIC 3 ein EPROM nebst einer sorgfältig aufgemachten 57seiten Anleitung mit Bedienungsbeispielen. Das SM-Kit dient der Programmierung, nicht dem Bereitstellen zusätzlicher BASIC-Befehle. Seine wesentlichen Dienste sind: REPEAT der Tastenfunktionen, AUTO (Zeilennummern) mit der Besonderheit seiner Ein- und Ausschaltung. Statements ab erster Schreibstelle aktivieren AUTO ab jetzigem Nummernbereich, Statements ab folgender Stelle schalten es aus. Die DELETE-Funktion arbeitet wie LIST, nämlich von ... bis. NUMBER (Renumber) kann ebenfalls von ... bis Zeile gesteuert werden (im Gegensatz z.B. zum Toolkit), erzeugt eine Referenzliste und verhindert die unbeabsichtigte Programmzerstörung.

Bei aktiviertem SM-Kit werden Fehler zur RUN-Zeit automatisch mit fehlerhaftem Statement und aufgesetzter Cursorposition gezeigt. Die FIND-Funktion ist für das Editieren nützlich. Man kann Suchbereiche definieren. Gefunden werden entweder BASIC-Worte und Variable oder Textketten nach REM. Durch geschickte Wahl des Suchstrings z.B. mit %, dem Dollarzeichen oder = können Variable nach Typ oder die Zuweisungen gefunden werden. Auch diese Funktion hat ein REPEAT. Der DUMP ermöglicht die gezielte Ausgabe von Variablen-Inhalten. Für die Protokollierung solcher Ausgaben gibt es ein Hardcopy auf Drucker oder Floppy. Der TRACE mit Vorwahl des Startpunktes zeigt volle Statements und kann im Single Step auch Variableninhalte ausgeben.

Außer den DOS-Kurzbefehlen sind die Lade- und Speicherbefehle wichtig und interessant: Beim Laden ist ein APPEND (Anhängen) ab Zeilennummer möglich, auch das verschiebliche Hereinziehen einer Programmdatei ab Speicheradresse. Der MERGE-Befehl mischt beliebige Zeilen eines auf der Diskette liegenden Programmes in ein speicherresidentes hinein. Der SAVE-Befehl vollzieht sofort ein VERIFYund erlaubt es auch, von ... bis Zeile abzuspeichern. Damit sind alle Möglichkeiten gegeben, in Modulen zu programmieren und diese nach Bedarf zu gruppieren. Das in diesem Heft abdruckte Programm für die Cross Reference List für BASIC-Variable stellt auf diesem Wege eine weitere Hilfe für die Verbindung dar. Bezug des SM-Kit durch die Commodore-Händler. Preis (unverbindlich) DM 200,--.

R. L.

# Buchbesprechung

**Flögel, E.: Programmieren in Maschinensprache mit dem 6502. 247 S. Hofacker-Verlag, München 1981, ISBN 3-921682-61-4, DM 49,--.** Das kurz vor Radaktionsschluß eingetroffene Buch führt den Untertitel 'Einführung und Programme' und hat folgende Gliederung: Einführung (Seite 1), Befehlssatz (17), Adressierungsarten (25), Programmentwicklung (35), Assembler-Programmierung (61), Programmbeispiele (67), 6502-Systeme (139), Disassembler (189), Interfacebausteine (203), Fehlersuche in Programmen (223), Anhang mit Befehlsliste (227).

Das umfangreichste Kapitel ist mithin das der Programmbeispiele, gefolgt von den 6502-Systemen. Letzteres bietet für Apple, Atari, AIM, CBM, PET, VC20 und Ohio C1P einige Beispiele für das Programmieren in Maschinensprache. Es kann damit natürlich nicht die erforderlichen Systemhandbücher ersetzen, sondern bestenfalls zu eigenen Vergleichen anregen. Man bemerkt, auch in den Beispielen, eine gewisse Bevorzugung des APPLE und des für ihn benutzten Assemblers. Schmerzlich ist das Folgende: Es gibt eine Standard-Syntax für 6502 Assembler. Sie wird dem Leser in keiner Weise unterbreitet. Weiter: Für den PET/CBM wird nur der kleine im ROM enthaltene Monitor erwähnt, nicht jedoch die Tatsache, daß der Commodore-Assembler und die im Markt verbreiteten Makroassembler den CBM mit hervorragenden Fähigkeiten in der maschinensprachlichen Programmierung versehen.

Die Programmbeispiele schreiten von einfachem (Zeitschleifen, Tabellen, Listen) zu hochentwickelten Routinen fort: Mathematik, Gleitkommaarithmetik, Relocator. Zu den Programmen in Assembler-Listings, versehen auch mit Hexdumps und Flußdiagrammen, vermißt der Rezensent die

Kommentare in den Programmzeilen. Manche mehrseitige Programme stehen diesbezüglich ganz nackt da. Es muß damit die Frage gestellt werden, ob die gewählte Darstellungsform eignet, um das Programmieren in Maschinensprache erlernen zu können. Der Nutzen mag im Einzelfalle mehr darin liegen, eine abtippfähige Vorlage zu erhalten. R. L.

# Produkte

**Die HERA-electronic in Nürnberg bietet eine Reihe von Erweiterungsplatinen für den AIM 65/PC 100 an:** a) Multifunktionsgruppe zur Aufnahme der 6502 CPU (unterseitiger Steckadapter) mit Bufferung der Adreß-, Daten- und Kontrollsignale, voller Dekodierung des Adreßbereiches A000 ... AFFF für weiteres I/O oder für Speicher und einer Erweiterung des Debuggingbereiches um die 4K-Adreßbereiche B, C und D (BASIC/Assembler). Zur Inbetriebnahme sind die TTL-Dekoder auf den Plätzen Z15 und Z19 zu entfernen (DM 168,-). b) Zwei EPROM-Weichen mit 2 bzw. 3 ROM-sockeln und Steckadapter bzw. Flachkabel, die den wahlweisen Betrieb von residenten Festwertspeichern ermöglichen, die für denselben Adreßbereich dekodiert sind, z.B. BASIC/FORTH oder GWK-BASIC/Assembler. Info und Bezug: HERA electronic professional, Hermine Heckl, Gärtnerstr. 12, 85 Nürnberg, T. 0911-34 24 25. Ω

**Der Matrixdrucker EPSON MX 80 F/T kann bequem auf die Zeichensätze des CBM umgestellt werden.** Es handelt sich um ein paralleles IEEE 488–Interface mit genormten IEEE-Stecker und Kabel. Die Platine wird in die dafür vorbereitete Interface-Buchse des EPSON eingesetzt und ist damit betriebsfertig. Der CBM hat bekanntlich einen eigenen ASCII-Zeichensatz und vertauscht unter gemischter Groß- und Kleinschreibung (Poke 59468,14) die Darstellung. In Heft 19 auf Seite 50 wurde eine Routine abgedruckt, die das Problem softwaremäßig löst. Bei dem hier erprobten und besprochenen IEEE-Interface ist eine solche Codewandlung nicht mehr nötig, sie wird vom Interface besorgt. Die deutschen Umlaute und das 'ß' sind nach DIN belegt. Die gemischte Schreibweise wird erzeugt, indem man das Interface mit der Geräteadresse 4 anspricht. Wählt man jedoch die Geräteadresse 20, so wird neben der Großschreibung des CBM auch der Grafik-Zeichensatz des CBM einschl. Steuerzeichen des Cursors druckbar gemacht.

Der Vorteil des Interface darf damit darin gesehen werden, daß es es den EPSON nicht nur an den IEEE-Bus anschließt und die beiden Zeichensätze des CBM druckbar macht, sondern auch das immer benutzte Verbindungskabel enthält. Preis DM 398,- (unverb.) inkl. MwSt. Bezug: Fa. Norbert Stellberg, Blindenaaf 36, 5063 Overath, T. 02206 - 66 44. Ω

**Speziell für CBM-Rechner wird jetzt eine Reihe von Erweiterungsplatinen angeboten,** die aber auch an anderen Rechnern eingesetzt werden kann: Sie ermöglichen es, durch einfaches Umschalten wahlfrei verschiedene ROM-Sätze zu betreiben, z.B. BASIC 3 plus 3 zugehörige Anwender-ROMs oder BASIC 4 + 2 ROMs (2x7 24-polige ROM-Sockel, lötfreier Anschluß). Eine etwas einfachere Ausführung gestattet nur die Umschaltung der 5 BASIC-Steckplätze und bedingt kleine Veränderungen an der Computerplatine. In vielen Fällen wird auch eine weitere Platine nützlich sein, die 3 ROM-Plätze je 4-fach zu bestücken gestattet (extern umschaltbar), um z.B. abwechselnd NETIM, SM-Kit usw. beim CBM oder BASIC und FORTH beim AIM 65 zu betreiben. Auch ein 6-fach Power-Switch für den User-Port ist lieferbar (bis 1 kW, ohmsche Last). Info und Bezug: Hans Jakeschky, Am Redder 2-4, 2071 Hammoor. T. n. 20 h: 045 32 - 73 64. Ω

**SANYO Color-Daten-Monitor CTP 3245-C.** Dieses Gerät mag in vielen Fällen eine Alternative zum reinen Datenmonitor sein. Ad eins handelt es sich um einen zeitgemäßen PAL-Farbfernsehempfänger mit 36 cm Bildschirm und sechs belegbaren Stationstasten, Gewicht 11 kg. Das Gerät wurde hier vorübergehend getestet und lieferte dabei ein einwandfreies farbiges Fernsehbild, auch ohne Dachantenne, wie man es bei einem Gerät zu ca. DM 1.400,- erwarten darf. Ad zwei handelt es sich aber um einen Farb-Monitor mit 9 MHz Bandbreite und Composite-Video-Eingang für den Anschluß an Computer. Für die Benutzung dieser Funktion stand hier leider kein APPLE-Computer oder VIC 20 von Commodore zur Verfügung. Der Sanyo ist jedoch für solche Geräte vorgesehen. Er kann horizontal 340 und vertikal 280 Punkte auflösen (80 Zeichen, 24 Zeilen, 5x7-Matrix) Ad drei ist das Gerät auch ein einfarbiger Monitor mit 'Grünschalter' Beim Betätigen dieser Taste erscheint die Schrift augenschonend grün auf schwarzem Grund. Diese Funktion wurde hier be-

nutzt und für die Textausgabe am Heimcomputer für gut befunden. Man darf natürlich kein so scharfes Bild wie bei einem 18 MHz-Monitor erwarten. Bildschirmarbeitsplätze stellen hohe Anforderungen. - Das ähnliche Modell CTP 3245-V ist als Empfänger-Monitor für Video-Systeme gedacht mit Video/Audio- Eing/Ausgang über 8pol. EIA-Buchse. R.L.

# AIM Spezial (9)

**Fehler im Monitor-Programm des AIM 65:** Herr Michael Beland aus Hamburg weist auf einen Fehler im Unterprogramm HEX hin (EA7D). Es akzeptiert das Sonderzeichen '@' als Ziffer. Nach Ansicht des Herausgebers sollte in Loc. EA90 mit CMP #$41 verglichen werden, um die richtige Filterung der Zifferneingaben zu erhalten. - Vielen Dank für den Hinweis!

**Die System-VIA des AIM ist nicht interruptfähig,** wenigstens nicht auf dem Gerät der Erstserie mit Nr. A00438 (Herausgeber). Es ist nicht bekannt, ob später noch eine Änderung vorgenommen wurde. Aus nicht ersichtlichen Gründen ist Pin 21 der VIA auf Platz Z32, der IRQ-Pin, unbeschaltet. Bei Bedarf sollte man auf der Platinen-Unterseite eine entsprechende Drahtbrücke zum Pin 4 des Z31 verlegen. Nach dieser Beschaltung ist es z.B. möglich, in der seriellen Datenübertragung ein ankommendes Startbit per Interrupt zu erkennen: Man lädt Timer T2 mit 00 vor, schaltet im ACR Bit 5 auf 1, um die Zählung von Impulsen an PB6 zu ermöglichen und schaltet Timer T2 im IER auf Interruptmöglichkeit. Mit der abfallenden Impulsflanke (=Startbit) am seriellen Interface (PB6) hat man damit seinen Interrupt für den Datenempfang.

**Beim Einlesen von Magnetbandcassetten treten immer einmal wieder Lesefehler auf,** auf Grund derer ein File als verloren betrachtet werden muß. Nach den Gesetzen von Murphy treten diese Fehler regelmäßig wie folgt ein: a) bei unersetzlichen Files, die nur einmal angelegt worden sind, b) bei Files großer Länge, hier besonders nach den ersten Sätzen und c) bei Files, die nicht dokumentiert wurden. Eine Rettung (recovery) ist nunmehr möglich. Das nachstehende Beispiel gilt für EditorTextfiles, die vom tape drive 2 gelesen werden. Eine Änderung auf drive 1 ist analog möglich,

```
A9 LDA #$54      'T' FÜR TAPE
8D STA $A412     IN DAS INFLG BRINGEN
A9 LDA #$01      TAPE 2 IN TAPIN SETZEN
8D STA A434      NACH TAPIN
A9 LDA # ...     NUMMER DES FOLGESATZES VERWENDEN
8D STA 0115      NACH BLK, BLOCK-VORGABE
4C JMP $F7D8     FORTSETZUNG BEI INPU 1
```

Handhabung: Beim Auftreten eines Fehlers stoppt der vom AIM ferngesteuerte Rocorder automatisch. Jetzt das Programm eingeben, und zwar mit Benutzung der auf den schadhaften Block folgenden Block-Nummer. Band auf die Lücke oder das Ende des schadhaften Blocks zurückpositionieren, ggfs. akustische Kontrolle, Programm mit G/ starten, ggfs. Vorgang wiederholen. Der Magnetkopf muß also vor oder in den SYNCs des ausgewählten Blockes nach dem Hochlaufen des Bandes in der Sollgeschwindigkeit lesen können. Nach dem Laden muß man ggfs. manuell Trümmer aus dem Textspeicher entfernen und Ergänzungen vornehmen. Undruckbare Zeichen erkennt man mit dem M-Befehl, mit '/' ändert man sie ab. Ggfs. fügt man auch hex 0D für 'CR' ein.

Ω

# Editorial

Nach einem wiederum eigenwilligen Sommer (das Wirtschaften von 4 Milliarden Menschen verändert wohl doch das Klima?) beginnt jetzt die Herbstsaison. Auf den bevorstehenden Messen wird man wiederum viele neue Angebote für den Computer-Anwender finden. Beim vielfältigen Angebot werden die Sichtung des Materials und die Auswahl eines Computers und seiner Erweiterungen oft schon etwas schwierig. Bei den notwendigen Prüfungen sollte man sich besonders vor Augen halten, daß die für ein System benötigte Software letztlich teurer als die Hardware wird, weil ihre Erstellung so zeitaufwendig ist. Software ist im Moment der Erstellung oder Beschaffung fast niemals 'fertig'. Unter den sich wandelnden Anforderungen des Anwenders oder der Vorschriften ist stets eine Software-Pflege notwendig. Dazu gehören Dokumentation und überhaupt gegebene Anpassungsmöglichkeiten. Das gilt für Entwicklungen aus dem eigenen Haus ebenso wie für fertige Programmpakete, die man sich immer zunächst vorführen lassen sollte. Es gilt auch für Software, die man entwickeln läßt. Wem ist schon gedient, wenn der zu befragende Programmierer plötzlich einen anderen Wohnort hat?

Auch Hardware ist nicht problemlos. Wir haben in dieser im Grunde aufstrebenden Branche bereits einige Konkurse von Computergeschäften gesehen. Außer Ungeschick dürfte dabei der von Anfang an bestehende Mangel an Eigenkapital mitgespielt haben. Optimismus und der geringere Preis machen noch nicht die Leistungsfähigkeit aus. Der Händler muß fachgerecht beraten und auch die Risiken überbrücken können, die sich aus der ständigen Veränderung des Marktes und aus den vielleicht auch einmal wieder fallenden Kursen des US-Dollars ergeben können.

Man erlebt es leider auch immer wieder, daß in Kellerbetrieben zusammengestoppelte Hardware im Versand angeboten wird. Man weise solche Ware zurück, Angebot, Ausführung und zugesagte Leistung müssen zueinander passen. Wo das nicht der Fall ist, scheue man sich nicht vor einer Weitergabe berechtigter Kritik.

Seit diesem Jahr gibt es weitere Computerzeitschriften. Das Informationsangebot wird damit grösser und deckt eine Vielzahl von Rechnern und Systemen für ein großes Publikum ab. Das 65xx MICRO MAG wird wie bisher konzentriert für wenige Systeme arbeiten und für sie möglichst viel gehaltvolle Artikel und Software bringen um den Gedankenaustausch in der vorwiegend professionell orientierten Leserschaft zu pflegen. Und man darf davon ausgehen, daß die Autoren für eine Rückfrage zur Verfügung stehen. Auf diesem Wege haben bereits viele gleichgesinnte Betreiber ständigen Kontakt miteinander aufgenommen.

Dieses Heft konnte nicht alle vorliegenden Beiträge und Anregungen aufnehmen. Die Autoren werden noch um etwas Geduld gebeten.

Ω

## Workshops für 6502 & 6809

Die Reihe der erprobten und bewährten zweitätigen Programmierkurse für 6502 und für 6809 wird fortgesetzt:

**Fortgeschrittenen-Workshop 65xx.** 1. Tag: Einführung in die Assemblersprache unter Wiederholung des Befehlssatzes, Benutzung des Text-Editors. 2. Tag: Interface-Programmierung unter Benutzung der Assemblersprache.

**6809 Assembler- und Interfaceprogrammierung.** 1. Tag: Architektur, Register, Signale. Gründliche Vorstellung des Befehlssatzes in Assemblersprache. 2. Tag: Interfaceprogrammierung unter Benutzung der Assemblersprache.
Preis: Jeweils DM 560,-- + MwSt. Im Preis eingeschlossen: Lehrgangsmaterial, 2 Mittagsmahlzeiten und Erfrischungen. Kleine Teilnehmerkreise. Ort: Ahrensburg bei Hamburg. Sonderprospekt auf Anfrage. Termine: Für 6502 am 18./19. Sept. 1981, für 6809 am 2./3. Okt. 1981, jeweils Freitag und Sonnabend von 8.30 bis 17 Uhr.
Anmeldung beim Herausgeber. Schulungen in Firmen nach besonderem Angebot. Anfängerkurse können bei ausreichendem Interesse anberaumt werden.

Ω

**Kleinanzeigen**

**AIM 65 BASIC-Erweiterung (4K)**
Einfaches Editing, TRACE, Befehls-Kurzeingabe, TIME, AUTO, RENUMBER, FRAC etc..
Software DM 168,-- auf CC, EPROM auf Anfrage. Info gegen Rückporto von R. Hammerschmidt, H.-Heine-Str. 4, 4006 Erkrath.

**AIM Floppy Disk (DAIM-Controller mit BASF-Laufwerk)** inkl. Kabeln und DOS DM 1.250,-. H.J. Regge, Bremen, Tel. 0421 - 71 114.

**Antennen-Steuerungsprogramm für Satelliten in zirkularer Bahn ges. für AIM 65 mit 4K-BASIC.** L. Hoorelbeke, ON6HR, Cogelsdreef 31, B-2288 Bouwel-Grobbendonk, T. 015- 51 45 40.

**COMPASS Floppy-Disk-Controller für direkten Anschluß an AIM 65/PC 100 modifiziert,** DOS verbessert, VB 600,- DM (siehe 65xx MICRO MAG Nr. 18, S. 46 ff.). Weitere Systemsoftware VS. M. Helm, Adolf-Ey-Str. 2a, 3393 ClausthalZellerfeld, Ruf 053 23 - 17 07 (Corps Hercynia).

**Diverse Mini-Floppy-Laufwerke BASF 6106 zu verkaufen..** DM 450,--. Fa. Kuno Schöllhorn, Rubensstr. 1, 6700 Ludwigshafen, Tel.: 0621 - 56 83 70.

Ω

# Des Programmierers Traum

Leider wird das „Fehlerspray" für Programme weiterhin ein Traum bleiben.

Das **-Kit** jedoch, mit seinen ausgefeilten Debug-Routinen, können Sie schon heute kaufen. So zeigt das **Trace** des **-Kit** nicht nur jede angelaufene Zeilennummer, vielmehr wird Statement für Statement komplett ausgegeben – auf Wunsch sogar mit aktuellen Variablen-Inhalten. Da die Ausgabe wahlweise auf Bildschirm, Floppy oder Drucker erfolgen kann, erhalten Sie so auf einfache Art ein dynamisches Programmlisting! Doch auch alle anderen Funktionen des **-Kit** sind bis ins Detail ausgefeilt. So haben Sie neben einem echten **Merge** und einem vielseitigen **Number,** noch eine ganze Reihe nützlicher Befehle, sowie ein elegantes Floppyhandling zur Verfügung. Wir haben uns größte Mühe gegeben, die Kapazitäten eines 4 KB-Roms und nicht die Ihrer Brieftasche voll auszunutzen.

EMPFOHLEN DURCH COMMODORE

**Unverbindlich empfohlener Preis DM 200,–**

**Deutschland:** Commodore Fachhändler
**USA:** AB-Computers, 252 Bethlehem-Pike, Colmar, PA 18915
Tel.: (215) 822 77 27
**England:** -Software (U.K.), 24 Long Street, Dursley, Gloucestershire, Tel. 04 53-4 64 96
**Schweiz:** Instant-Soft AG, Stetter Str. 25, CH-5507 Mellingen

Softwareverbund
Microcomputer GmbH
Scherbaumstr. 29, 8000 München 83

# 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

**Aktuelle IC- Preise incl. MWST.,** Mengenrabatt ab 8 Stück pro Typ.
Wir liefern nur Spitzenfabrikate (Fujitsu, NEC, Motorola, TI usw.):

| | | |
|---|---|---|
| 2114L 450 ns = 7,-- | 2114L 200 ns = 9,-- | 2114 CMOS = 15,-- |
| 4116 200 ns = 8,-- | 4116 150 ns = 12,-- | 4164 200 ns = 80,-- |
| 2716 450 ns = 16,-- | 2532 450 ns = 38,-- | 6116 150 ns = 45,-- |
| 2732 450 ns = 38,-- | 6502, 6522, 6532, 6551, 8257-5: à 28,-- | |
| MC 6845 CP = 58,-- | 74LS240, -241,-243,-244, -245, 8T28: à 4,50 | |

TI-93C und TI83C -3mm-Flachprofil-IC-Fassungen: 8/14/16/18 polig: à 0,60 DM. 20/24/28/40 polig: à 1,50 DM. EPROM-Löschlampe mit E27-Schraubsockel DM 65,-.


DIPL.-ING. HORST NEUDECKER

INGENIEURBÜRO FÜR MESSTECHNIK

**Mehringplatz 13** **Tel.: 030 - 614 89 00**
**1000 Berlin 61** **030 - 251 20 00**

# Video-Interface für AIM 65 / PC 100

Europakarte 100x160 mm, Stromaufnahme 5V, 1A, 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 µ P-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 | 2356,- +MWSt= | 2662,28 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.

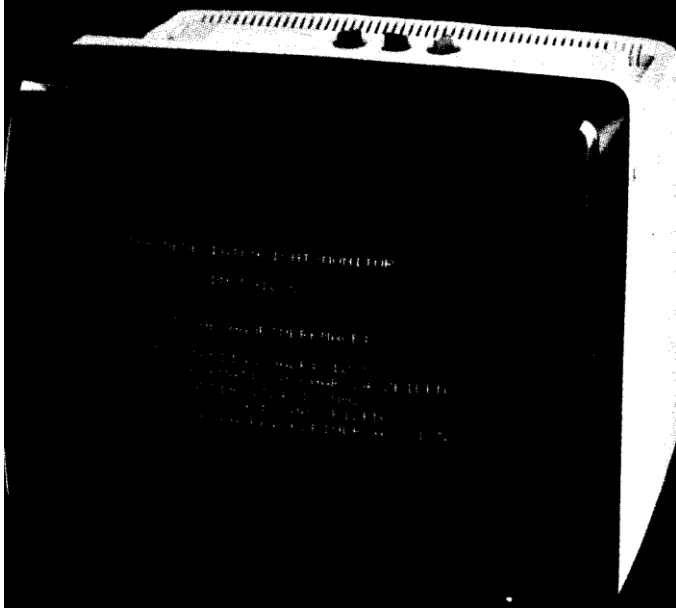
| | | |
|---|---|---|
| Preis für 32 KB | DM 692,- + MWSt = DM 781,96 |
| teilbestückt 16 KB | DM 426,- + MWSt = DM 481,38 |

**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

# Das Warten hat sich gelohnt...



Original Sanyo-Datenmonitor DM 5912/CX – jetzt zu einem noch günstigeren Preis. Ihr Fachhändler erwartet Sie. 31 cm Bildschirm. Leuchtstoff P 31-grün. Standard Video Eingang (CCIR), Gehäuse Kunststoff elfenbein, 220 V Netz

Nähere Informationen mit Händlernachweis:

SANYO VIDEO-SYSTEM

SANYO VIDEO Vertriebs GmbH & Co.
Lange Reihe 29 · 2000 Hamburg 1
Telefon (040) 24 62 66 · Telex 2 174 757


# Treffpunkt BERLIN !

**Samstag, den 5. September 1981**

Vom 4. bis zum 13. 9. 1981 findet in Berlin die Internationale Funkausstellung statt.

In diesem Zusammenhang veranstaltet die Fa. Dipl.-Ing. Horst Neudecker in ihrem Elektronik-Laden, Mehringplatz 13, 1000 Berlin 61, Tel.: 030 - 614 89 00

am 5. 9. 1981 von 10 - 17 Uhr einen Tag der offenen Tür.

An diesem Tag und in diesem Zeitraum wird der Herausgeber der Zeitschrift 65xx MICRO MAG dort anwesend sein, sich über das persönliche Kennenlernen freuen und für die Erörterung von Fachfragen zur Verfügung stehen.

Dem Erstversand dieses Heftes liegt ein Prospekt des Franzis-Verlages, München bei.

# Sanyo Color-Daten-Monitore CTP 3245-C und CTP 3245-V für jedes Programm: Fernsehen, Computer, Video

Dieser neue Typ Color-Daten-Monitor überzeugt durch die vielen Einsatzmöglichkeiten. In der optimalen Bildgröße von 36 cm.

**1** Er empfängt ganz normal jedes Fernsehprogramm. In PAL-Color über 8 Stationstasten.

**2** Modell CTP 3245-C hat einen PAL-Video-Composite-Eingang für jeden Computer. Mit bester Bildschirmauflösung von ≈ 9 MHz

**3** Der zusätzliche „Grünschalter" sorgt für augenschonende Computer-Textdarstellung in Grün auf schwarzem Grund.

**4** Modell CTP 3245-V ist der ideale Empfänger-Monitor für jedes Video-System. Video-Audio-Eingang u. Ausgang über die 8-Pol-EIA-Buchse.

Insgesamt also Color-Daten-Monitore der absoluten Sonderklasse. Und zu einem Preis, der jeden Anwender auf das angenehmste überraschen wird.
Fragen Sie Ihren Computer-Fachhändler. Weitere Informationen über das große Sanyo Daten-Monitor-Programm und Händlernachweis bitte sofort anfordern.

**SANYO**
VIDEO-SYSTEME

SANYO VIDEO Vertriebs GmbH & Co.
Lange Reihe 29 · 2000 Hamburg 1
Telefon (040) 24 62 66 · Telex 2174757

# 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. Beiträge, die nicht besonders gekennzeichnet sind, stammen vom Herausgeber. COPYRIGHT 1981 by Roland Löhr. Alle Rechte vorbehalten, auch die des auszugsweisen Nachdrucks, der Übersetzung, der fotomechanischen Wiedergabe und die der Verbreitung auf magnetischen und sonstigen Trägern. - 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 7-19 können einzeln oder komplett nachgeliefert werden, und zwar zum Endpreis von DM 7,80/Stück.

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 48,--

### Microprocessor Systems Engineering

von Camp, Smay, Triska, Lehrbuch, ca. 640 S., das vor allem das Zusammenwirken von Hardware und Betriebsprogramm für den AIM 65 erklärt. Vergleiche des 6502 mit 6800 und 8080 DM 86,--

Null-Ziehkraft-Sockel für 24-pol. ROMs DM 14,50