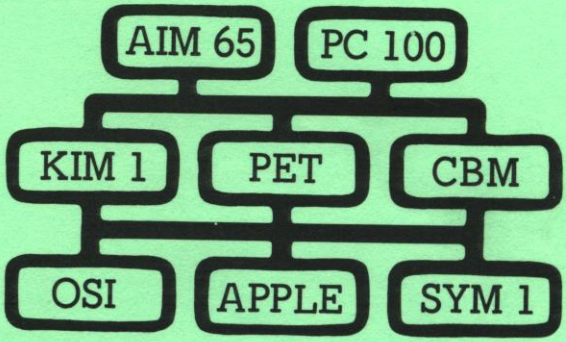
65xx MICRO MAG

COMPUTING · SOFTWARE · HOBBY

DM 8,50 Nr. 21 Oktober 1981



Inhaltsverzeichnis

Umrüstsatz:

6502 - 6809

Die Karte mit 6809-CPU, Taktgenerator und Steckverbindern
wird auf den Sockel der 6502-CPU gesteckt.
Software (EPROM) für Grundmonitor Ende Oktober lieferbar.

Weiterhin lieferbar:

CPU-Karte Nico 65 mit 6502 CPU
CPU-Karte Nico 69 mit 6809 CPU
Combo-Karte mit 8 K RAM und 2 VIAs
Video-Karte
Buskarte
EPROM-Programmiergerät
Experimentierkarte
A/D-Karte
Mini-Digital-Kassettenrekorder
mit Betriebsprogramm und Interface für AIM und 6809

ELEKTRONISCHE BAUGRUPPEN
MICROPROZESSOR-TECHNIK
Dohmann

Im Wiehagen 15 - Tel. (05241) 67480
4830 Gütersloh 1 2

AIM 65 mit Floppy Disk CBM 4040

Obwohl hierzulande viele tausend AIM 65 und PC 100 in der Hand von Anwendern sind, ist der Gebrauch einer Floppy Disk an diesen Computern noch gar nicht so häufig. Dabei gibt es seit mindestens 1 1/2 Jahren entsprechende Anschlußmöglichkeiten, z.B. die Compass oder die GWK-Floppy. In Heft 17 wurde ferner der Betrieb einer OSI-Floppy abgehandelt.

Dieser Aufsatz beschreibt den Betrieb der CBM-Floppy am AIM. Wer sich die Mühe macht, die wenigen benutzten Monitorprogramme sinngemäß nachzucodieren, hat in aller Kürze auch für jeden anderen Rechner mit CPU 6502 eine Floppy Disk in Gebrauch. Das benötigte einfache Interface ist in Heft 19 des 65xx MICRO MAG beschrieben (AIM 65 am IEEE 488-Bus). Die Floppy selbst und ihre Ansprache sind in diesem Heft ausführlich abgehandelt: 'Der Datenverkehr Rechner und CBM-Floppy 4040'. Der Leser möge diese Artikel zum breiteren Verständnis heranziehen.

Der Betrieb einer Floppy bringt dem Betreiber erhebliche Zeitgewinne und auch mehr Ordnung in seine Files. Mit den nachfolgenden Programmen werden pro Sekunde etwa 2K Bytes geladen. Gegenüber dem Laden vom Tonband hat man also die 40fache Geschwindigkeit. Eine Floppy vom Typ 3040 oder 4040 stellt einen peripheren Datenspeicher von etwa 340 KB für den wahlfreien Sofortzugriff zur Verfügung. Da man die Disketten wechseln kann, ist dieser Speicher beliebig erweiterbar, auch ist der Betrieb mehrerer Floppies am IEEE-Bus möglich und schließlich gibt es das Modell 8050 mit 1 MB Speicherkapazität.

Eine auf Floppy abgelegte Datei kann durch SCRATCH wieder entfernt werden, um Raum zu schaffen. Man macht davon während einer Programmentwicklung Gebrauch, um nicht mehr benötigte Zwischenstadien zu entfernen. Andererseits vollzieht man viel häufiger einen sichernden Dump auf die Floppy als auf das Tonband, weil es ja so schnell geht.

Wer seinen AIM häufig benutzt, wird seine Produktivität mit einer Floppy merklich verbessern und die Kosten von gut DM 3000 damit rechtfertigen können. Er wird sich weiter überlegen, ob er nicht die Adreßbereiche der Festwertspeicher von hex B000 bis DFFF mit RAM bestückt, um schnell zwischen den Sprachen und Assembler wechseln zu können. 8K FORTH oder BASIC sind in 4 Sekunden geladen, das geht schneller als das Wechseln von ROMs. Und in diesem Falle wird er mit einem kleinen Trick auch seinen Assembler etwa fünfmal schneller machen und seine Produktivität weiter verbessern.

Die Benutzung einer CBM-Floppy bietet weiter folgende Vorteile: Es ist ein vieltausendfach bewährtes komplettes Gerät mit Gehäuse, Stromversorgung und genormter IEEE-Buchse. Sie ist an jedem Computer mit entsprechendem Businterface betreibbar. Das ist interessant, wenn mehrere Computer betrieben werden (wie auch hier). Man kann sie sogar abwechselnd von jedem Computer am Bus benutzen, wenn nicht zwei zugleich den 'master' mimen wollen. Wenn das passieren könnte, sollte man eine Verriegelungsschaltung benutzen. Auf jeden Fall kann man in einer solchen Konstellation die Files für alle Rechner zugänglich machen. Und die Floppy entlastet den sie betreibenden Computer von aller Verwaltungsarbeit, man muß kein DOS (Disk Operating System) programmieren oder anpassen.

Zu den Programmen: Der Programmabschnitt INITI öffnet die anwenderbestimmte Ein- und Ausgabe (User) für die Floppy. Damit wird es hernach möglich, einen beliebigen Speicherbereich FROM= .. TO=.. mit dem D-Befehl (Dump) abzuspeichern. Man antwortet auf den Prompt OUT= mit 'U'. Jetzt erfolgt die Eingabe eines Dateinamens nach F= (man vermeide Blanks am Schluß, weil sie Bestandteil des Namens werden und beim Laden zu Fehlbedienungen führen mögen). Mit T= wird das Laufwerk (Diskdrive) vorgegeben. Das Programm setzt dabei T=1 zu Laufwerk 0 um und T=2 zu Laufwerk 1. Das Speichern beginnt nach Betätigen der Return-Taste.

Entsprechend erfolgt das Laden, also z.B. (L)IN= U F= PROG1 T= 1 (Return). Es wird dann die Datei 'PROG1' von Laufwerk 0 geladen.

In nachstehenden Listing findet man außer dem schon erwähnten INITI zwei weitere Hauptabschnitte, nämlich das USOINI, welches das Abspeichern besorgt, und ziemlich zum Schluß das USIINI zum Laden.

Die dazwischenliegenden Unterprogramme dienen zum Teil der Interfaceinitialisierung, wie OUTPUT und INPUT, zum Teil der Erzeugung und Abfrage von Steuersignalen für den IEEE-Bus, wie SAMPLE, ATNLO, ATNHI, EOILO, UNTAL, UNLIS, oder sie besorgen allgemeine Dienste im Ablauf der Hauptprogramme, wie HEADER, SEND, ACCEPT, NAMIN oder PRIMAD.

Die Programme können nur ein Anfang sein, um einerseits die weiteren Möglichkeiten des AIM 65 (Editor, BASIC, FORTH, Assembler) einzubinden und um weitere Befehle zu implementieren, die die Floppy verstehen kann. Der Artikel 'Der Datenverkehr ...' in diesem Heft ist dafür eine wertvolle Arbeitsbasis. Zwischenzeitlich kann man sich behelfen, indem man bei Textfiles die in Zeropage ab DF unterhaltenen Pointer beachtet, für BASIC die Pointer ab 73 bis 76 für Programmanfang und -ende und für FORTH den Bereich von 030B bis zur Adresse-1, die durch HEREverwaltet wird. Bei ausreichendem Interesse wird voraussichtlich mindestens die Firma Neudecker in Berlin ab Mitte November 1981 ein Businterface anbieten. Zu diesem Zeitpunkt soll von hier aus auch die erweiterte Treibersoftware zur Verfügung stehen.

```
0000            ;--------------------
0000            ; AIM DUMP & LOAD  TO CBM 4040 FLOPPY
0000            ;
0000            ; V02.09.81-1
0000            ; FUER USERPORT-INTERFACE
0000            ; WIE IN HEFT 19 DES 65XX MICRO MAG BESCHRIEBEN
0000            ;--------------------
0000            ; SYMBOLE
0000 START              =$E00
0000            ;-----------------------
0000            ; ADDRESS THE INTERFACE CHIP
0000 VIA                =$A000          ; USER VIA OF AIM 65
0000                    *=VIA
A000 PORTB              =*              ; PORT TO CONTROL IEEE-BUS
A000 PORTA              =*+1            ; DATAPORT TO BUS
A000 DDRB               =*+2            ; DATA DIRECTION
A000 DDRA               =*+3
A000 ACR                =*+11
A000 PCR                =ACR+1
A000
A000            ;----------------------
A000            ; INTERFACE OPERATORS
A000 WRITE              =$E0            ; MULTIPLEXED BY CB2
A000 READ               =$C0            ; MULTIPLEXED PER CB2-OUTPUT
A000 RISE               =1              ; DETECT RISING EDGE OF ATN
A000 FALL               =0              ; DETECT FALLING EDGE OF ATN
A000 ATNHIG             =$0E            ; OUTPUT ATN=HIGH ON CA2
A000 ATNLOW             =$0C            ; OUTPUT ATN=LOW ON CA2
A000
A000            ;----------------------
A000            ; IEEE BUS OPERATORS
A000 UNLISN             =$3F
A000 UNTALK             =$5F
A000
A000 CLOSED             =$E1            ; CLOSE SAVE COMMAND CHANNEL
A000 CLOSEL             =$E0            ; CLOSE LOAD COMMAND CHANNEL
A000
```

```
A000            ;---------------------
A000            ;ASCII EQUATES
A000 LF         =$0A            ;LINEFEED
A000 CR         =$D             ;CARRIAGE RETURN
A000
A000            ;---------------------
A000            ;REFERENCES TO THE MONITOR
A000 STBYTE     =$E413
A000 PHXY       =$EB9E          ;SAVE REGISTERS
A000 PLXY       =$EBAC          ;RESTORE REGISTERS
A000 MON        =$E956          ;REENTER MONITOR
A000 ADDS1      =$E55D          ;ADD 1 TO STARTADDRESS S1/S1+1
A000 NAME       =$A42E          ;BUFFER FOR FILENAME
A000 S1         =$A41A          ;FROM-ADDRESS
A000 ADDR       =$A41C          ;TO-ADDRESS
A000 STIY       =$A429
A000 TAPIN      =$A434          ;STORES DISKDRIVE
A000 FNAM       =$E8A2          ;GET FILENAME & TAPEDRIVE-#
A000 LDAY       =$EB58          ;SIMULATE ZP-FETCH
A000
A000            ;---------------------
A000            ;KEYFUNCTION F3
A000            *=$112
0112            4C00OE JMP INITI
0115
0115            ;---------------------
0115            ;INITIALIZATION OF USER-OUTPUT
0115            *=START
0E00 INITI
0E00            A915   LDA #<USOINI    ;INITIALIZE USER INPUT
0E02            8D0A01 STA $10A        ;AND OUTPUT VECTORS
0E05            A90E   LDA #>USOINI
0E07            8D0B01 STA $10B
0E0A            A946   LDA #<USIINI
0E0C            8D0801 STA $108
0E0F            A90F   LDA #>USIINI
0E11            8D0901 STA $109
0E14            00     BRK             ;RETURN TO MONITOR
0E15
0E15            ;---------------------
0E15            ;SEND TO IEEE-BUS MAIN USER-OUTPUT
0E15 USOINI
0E15            209EEB JSR PHXY        ;SAVE REGISTERS
0E18            204F0E JSR OUTPUT      ;INIT PORT TO SEND
0E1B
0E1B            20A60E JSR NAMIN       ;GET FILENAME 5 BYTES & DRIVE
0E1E            A200   LDX #0          ;POINT TO START OF OUTTAB
0E20            20EC0E JSR HEADER      ;SEND THE OPENING BYTES
0E23            20C70E JSR PRIMAD      ;ADDRESS DEVICE & CHANNEL 1
0E26            20B80E JSR ATNHI       ;END OF SECONDARY ADDRESS
0E29            AD1AA4 LDA S1          ;START ADDRESS OF PGM, SAL
0E2C            207A0E JSR SEND
0E2F            AD1BA4 LDA S1+1        ;GET SAH OF PROGRAM
0E32            207A0E JSR SEND        ;SEND IT TO BUS
0E35 DU1        201D0F JSR DUMP        ;DUMP PGM BYTES TO BUS
0E38            D0FB   BNE DU1         ;SEE IF WE ARE AT ENDADDRESS
0E3A            20BE0E JSR EOILO       ;LAST BYTE TO BE SENT
0E3D            201D0F JSR DUMP        ;WITH EOI=LOW
0E40            20E00E JSR UNLIS       ;SEND UNLISTEN
```

```
0E43            ;END OF DUMP
0E43
0E43            20C70E JSR PRIMAD       ;CLOSE FILE
0E46            20E00E JSR UNLIS        ;SEND UNLISTEN
0E49            20ACEB JSR PLXY         ;RESTORE REGISTERS
0E4C            4C56E9 JMP MON          ;JUMP TO MONITOR
0E4F
0E4F
0E4F            ---------------------
0E4F            INITIALIZE PORTS
0E4F OUTPUT                             ;BE MASTER ON THE BUS
0E4F            A90F   LDA #$0F         ;INSTALL DATA DIRECTION
0E51            8D02A0 STA DDRB
0E54            8D00A0 STA PORTB        ;,ALL OUTPUTS HIGH
0E57            A9FF   LDA #$FF         ;INSTALL DATAPORT
0E59            8D03A0 STA DDRA         ;ENABLE OUTPUT
0E5C            8D01A0 STA PORTA        ;OUTPUT NEUTRAL ON BUS
0E5F            A9E0   LDA #WRITE       ;MULTIPLEX CHANNELS
0E61            8D0CA0 STA PCR          ;WRITE THE OUTPUT OF CB2
0E64            60     RTS
0E65
0E65 INPUT      A90F   LDA #$0F         ;INSTALL THE INTERFACE
0E67            8D02A0 STA DDRB         ;DATA DIRECTION
0E6A            A90D   LDA #$0D         ;NDAC LOW FOR RECEIVING
0E6C            8D00A0 STA PORTB
0E6F            A900   LDA #0
0E71            8D03A0 STA DDRA         ;PORT A FOR INPUT
0E74            A9C0   LDA #READ        ;MULTIPLEX CHANNELS
0E76            8D0CA0 STA PCR          ;WITH CB2 SIGNAL
0E79            60     RTS
0E7A
0E7A
0E7A            ---------------------
0E7A            SUBROUTINES TO SEND
0E7A SEND
0E7A            49FF   EOR #$FF         ;INVERT THE BYTE ACCORDING BUS
0E7C            8D01A0 STA PORTA        ;WRITE CHAR TO BUS      STANDARD
0E7F            AD00A0 LDA PORTB        ;FLIP THE DAV SIGNAL
0E82            2901   AND #1          ;DONT' TOUCH EOI
0E84            0906   ORA #$06         ;DAV LOW
0E86            8D00A0 STA PORTB
0E89 SEND1
0E89            AD00A0 LDA PORTB
0E8C            2960   AND #$60
0E8E            C920   CMP #$20         ;WAIT FOR NRFD=LOW AND NDAC=HIGH
0E90            D0F7   BNE SEND1        ;BYTE NOT YET ACCEPTED
0E92            A90F   LDA #$0F
0E94            8D00A0 STA PORTB        ;ALL HIGH AGAIN
0E97            A9FF   LDA #$FF
0E99            8D01A0 STA PORTA        ;SEND NEUTRAL
0E9C
0E9C            ---------------------
0E9C            SAMPLE BUS FREE
0E9C SAMPLE
0E9C            AD00A0 LDA PORTB
0E9F            2960   AND #$60
0EA1            C940   CMP #$40
0EA3            D0F7   BNE SAMPLE
0EA5            60     RTS              ;BUS IS READY FOR NEXT BYTE
0EA6
```

```
0EA6            ;--------------------
0EA6            ; INPUT FILENAME + DRIVE
0EA6 NAMIN
0EA6            20A2E8 JSR FNAM         ;GET FILENAME, 5 BYTES
0EA9            AD34A4 LDA TAPIN        ;WHICH DRIVE ?
0EAC            0930   ORA #$30          ;CONVERT TO ASCII
0EAE            8D34A4 STA TAPIN
0EB1            60     RTS
0EB2
0EB2            ;--------------------
0EB2            ; SUBROUTINES WITH INTERFACE COMMANDS
0EB2 ATNLO      A9EC   LDA #WRITE+ATNLOW
0EB4            8D0CA0 STA PCR          ;FLIP CA2 OUTPUT
0EB7            60     RTS
0EB8
0EB8 ATNHI      A9EE   LDA #WRITE+ATNHIG
0EBA            8D0CA0 STA PCR          ;FLIP CA2 OUTPUT
0EBD            60     RTS
0EBE
0EBE EOILO      AD00A0 LDA PORTB        ;GET FORMER STATE
0EC1            29FE   AND #$FE         ;MASK OFF EOI-BIT
0EC3            8D00A0 STA PORTB
0EC6            60     RTS
0EC7
0EC7            ;--------------------
0EC7            ; SUBROUTINES WITH IEEE COMMANDS
0EC7 PRIMAD     20B20E JSR ATNLO        ;SEND PRIM + SEC ADDRESS
0ECA            BD3A0F LDA OUTTAB,X     ;GET NEXT ITEM FROM TABLE
0ECD            E8     INX
0ECE            207A0E JSR SEND         ;SEND PRIMARY ADDRESS
0ED1            BD3A0F LDA OUTTAB,X     ;GET SECONDARY ADDRESS
0ED4            E8     INX
0ED5            207A0E JSR SEND
0ED8            60     RTS
0ED9
0ED9            ;--------------------
0ED9
0ED9 UNTAL      20B20E JSR ATNLO        ;SEND UNTALK
0EDC            A95F   LDA #UNTALK
0EDE            D005   BNE UNLIS1       ;BRANCH ALWAYS
0EE0
0EE0            ;--------------------
0EE0            ; SEND UNLISTEN CMD
0EE0 UNLIS      20B20E JSR ATNLO
0EE3            A93F   LDA #UNLISN
0EE5 UNLIS1
0EE5            207A0E JSR SEND
0EE8            20B80E JSR ATNHI        ;FLIP ATN TO HIGH
0EEB            60     RTS
0EEC
0EEC            ;--------------------
0EEC            ; GENERAL ADMINISTRATION OF MAIN PROGRAM
0EEC            ;--------------------
0EEC            ; SEND DRIVE-#, DELIMITER + NAME TO BUS
0EEC HEADER                             ;OPEN THE FILE
0EEC            20B20E JSR ATNLO
0EEF            209C0E JSR SAMPLE       ;BUS FREE ?
0EF2            20C70E JSR PRIMAD       ;SEND PRIM. & SEC. ADDRESS
0EF5            20B80E JSR ATNHI        ;ATN HIGH AGAIN
```

```
0EF8            AD34A4 LDA TAPIN       ;GET DRIVE 0 OR 1
0EFB            207A0E JSR SEND        ;ADDRESS DRIVE
0EFE            A93A   LDA #':      ;DELIMITER
0F00            207A0E JSR SEND
0F03            A000   LDY #0          ;ADDRESSER TO NAME
0F05 NLOOP      B92EA4 LDA NAME,Y      ;ADDRESS THE FILE
0F08            207A0E JSR SEND
0F0B            C8     INY
0F0C            C004   CPY #4          ;LAST CHAR ?
0F0E            D0F5   BNE NLOOP
0F10            20BE0E JSR EOILO       ;LAST BYTE TO BE SENT
0F13            B92EA4 LDA NAME,Y
0F16            207A0E JSR SEND        ;WITH EOI=LOW
0F19            20E00E JSR UNLIS       ;END OF OPENING COMMAND STRING
0F1C            60     RTS
0F1D
0F1D            ----------------------
0F1D            GET CURRENT BYTE & DUMP TO BUS
0F1D
0F1D DUMP       A91A   LDA #<S1
0F1F            A000   LDY #0
0F21            2058EB JSR LDAY        ;GET BYTE BY POINTER-SIMULATION
0F24            207A0E JSR SEND        ;SEND IT TO BUS
0F27            205DE5 JSR ADDS1       ;INCREMENT CURRENT POINTER
0F2A
0F2A            ----------------------
0F2A            SEE IF WE ARE AT ENDADDRESS (STATUS=0)
0F2A ENDADD     38     SEC             ;PREPARE SUBTRACTION
0F2B            AD1CA4 LDA ADDR        ;TO, LOW
0F2E            ED1AA4 SBC S1          ;CURRENT, LOW
0F31            D006   BNE E1          ;RETURN NOT EQUAL
0F33            AD1DA4 LDA ADDR+1      ;HIGH
0F36            ED1BA4 SBC S1+1
0F39 E1
0F39            60     RTS
0F3A
0F3A            ----------------------
0F3A            PRIMARY + SEC. ADDRESSES FOR DUMP IN CONSECUTION
0F3A OUTTAB     28     .BYT $28,$F1,$28,$61,$28,CLOSED
0F3B            F1
0F3C            28
0F3D            61
0F3E            28
0F3F            E1
0F40            ----------------------
0F40            PRIMARY & SEC. ADDRESSES FOR LOAD IN CONSECUTION
0F40 INTAB      28     .BYT $28,$F0,$48,$60,$28,CLOSEL
0F41            F0
0F42            48
0F43            60
0F44            28
0F45            E0
0F46
0F46
0F46
```

```
                ;---------------------
0F46            ;INPUT ROUTINES TO LOAD FROM FLOPPY
0F46 USIINI     209EEB JSR PHXY         ;SAVE REGISTERS
0F49            204F0E JSR OUTPUT       ;PREPARE PORT FOR COMMANDS
0F4C            20A60E JSR NAMIN        ;GET FILENAME + DRIVE-#
0F4F            A206   LDX #INTAB-OUTTAB   ;POINT TO LOAD COMMANDS
0F51            20EC0E JSR HEADER       ;OPEN FILE
0F54            20C70E JSR PRIMAD       ;FLOPPY NOW IS TALKER
0F57            20B80E JSR ATNHI        ;ATN HIGH AGAIN
0F5A            20650E JSR INPUT        ;ENABLE INPUT ON PORTA
0F5D            20B80F JSR ACC1         ;GET SAL
0F60            8D1CA4 STA ADDR         ;DEPOSIT START OF PROGRAM
0F63            20B80F JSR ACC1         ;GET SAH
0F66            8D1DA4 STA ADDR+1       ;SAH
0F69
0F69            ;---------------------
0F69            ;DATA ACQUISITION MAIN LOOP
0F69 ACCEPT     20B80F JSR ACC1         ;GET CURRENT PROGRAM BYTE
0F6C            08     PHP              ;SAVE EOI-STATE
0F6D            2016E4 JSR STBYTE+3     ;STORE TO MEMORY
0F70            28     PLP              ;CHECK FOR EOI
0F71            B0F6   BCS ACCEPT       ;NOT LAST BYTE
0F73
0F73            ;---------------------
0F73            ;NOW CLOSE ALL
0F73            204F0E JSR OUTPUT       ;FLIP THE INTERFACE
0F76            20D90E JSR UNTAL        ;MAKE 'EM UNTALK
0F79            20C70E JSR PRIMAD
0F7C            20B80E JSR ATNHI        ;ATN HIGH AGAIN
0F7F            20E00E JSR UNLIS        ;MAKE 'EM UNLISTEN
0F82            20ACEB JSR PLXY         ;RESTORE INDEXES
0F85            4C56E9 JMP MON          ;RETURN TO MONITOR
0F88
0F88            ;---------------------
0F88            ;GENERAL ADMINISTRATION OF INPUT
0F88 ACC1       2C00A0 BIT PORTB        ;WAIT FOR DAV LOW
0F8B            30FB   BMI ACC1         ;STILL HIGH
0F8D            AD01A0 LDA PORTA        ;GET INVERTED DATABYTE FROM BUS
0F90            48     PHA              ;SAVE IT UP TO RTS
0F91            A909   LDA #9           ;HANDSHAKE
0F93            8D00A0 STA PORTB        ;NRFD=LOW, NDAC=LOW
0F96            18     CLC              ;GET A DEFINITE EOI-STATE
0F97            AD00A0 LDA PORTB        ;SAVE EOI-STATE IN THE CARRY
0F9A            2910   AND #$10
0F9C            F001   BEQ ACC3         ;EOI WAS LOW, CARRY REMAINS LOW
0F9E            38     SEC              ;MARK EOI HIGH
0F9F ACC3
0F9F            A90B   LDA #$B          ;HANDSHAKE NRFD=LOW, NDAC=HIGH
0FA1            8D00A0 STA PORTB
0FA4 WART       2C00A0 BIT PORTB        ;WAIT FOR DAV TO GO HIGH AGAIN
0FA7            10FB   BPL WART
0FA9            A90D   LDA #$D          ;READYNESS: NRFD=HIGH, NDAC=LOW
0FAB            8D00A0 STA PORTB
0FAE            68     PLA              ;RETURN DATABYTE
0FAF            49FF   EOR #$FF         ;INVERT TO NORMAL REPRESENTATION
0FB1            60     RTS              ;RETURN WITH CHARACTER IN ACC.
0FB2                   .END
0FB2             ERRORS= 000                                    R.L.
```

Datenverkehr Rechner und CBM-Floppy 4040

Einleitung

In den vergangenen Jahren sind viele tausend Floppy Disks an Commodore-Rechner angeschlossen worden. Zu diesem bewährten Datenspeicher wurde gleichwohl nur wenig in den Zeitschriften veröffentlicht. Dieser Aufsatz behandelt daher die Befehle, die die Floppy versteht und dokumentiert den sie begleitenden Informationsaustausch auf dem IEEE 488-Bus. Wir werden dabei sehr schnell bemerken, daß die Floppy ein Peripheriegerät mit eigener Intelligenz ist, das den Rechner von aller Organisationsarbeit entlastet. Diese Dokumentation will nicht nur bei den CBM-Betreibern Verständnis für das vermitteln, was auf dem Bus 'spielt', sondern will im Zusammenhang mit den Aufsätzen 'AIM 65 am IEEE 488-Bus' (Heft 19) und 'AIM 65 mit Floppy CBM 4040' (in diesem Heft) die Wege aufzeigen, wie man letztlich jeden Computer mit der Floppy betreiben kann.

DOS- und BASIC-Versionen

Zur gerätemäßigen Umgebung: Es gibt die Floppies 2040, 3040, 4040 und 8050. Die Unterscheidung liegt im Betriebssystem, in der Spurenanordnung und in der Speicherkapazität. Die 2040 und die 3040 kann man als gleichwertig betrachten. Der Sprung setzt bei der 4040 ein, in die das DOS 2 implementiert ist. Dieses hat die angenehme Eigenschaft, daß die Laufwerke nach dem Einschalten oder nach dem Tausch einer Diskette nicht initialisiert werden müssen. DOS 2 unterstützt daneben die sog. relativen Files, auf deren Datensätze man lesend und schreibend einen wahlfreien Zugriff hat. - Die drei erstgenannten Floppies haben eine Nutzkapazität von etwa 170 000 Bytes je Laufwerk, dh. 340 KB je Gerät. Die 8050 speichert 512 KB je Laufwerk bzw. 1 MB je Gerät. Eine ältere Floppy kann mit dem bei Commodore-Händlern beziehbaren ROM- und Interfacesatz auf DOS 2 aufgerüstet werden. Sie wird dadurch zu einer 4040. Mit DOS 2 kann man von DOS 1 beschriebene Disketten weiterhin lesen, jedoch dort nichts hinzuschreiben, was ja nach entsprechendem Umkopieren keinerlei Behinderung bedeutet.

Parallel mit der Floppy-Entwicklung wurden das BASIC und das Betriebssystem der Rechner aufgebessert. Mit dem PET 2001 (BASIC 2) kann man keine Floppy betreiben, das geht erst ab BASIC 3. Aber man kann den PET mit 28poligen ROMs auf dieses BASIC umstellen. Auch für die Rechner der Serie 3000 gibt es ROMs zum Umrüsten auf BASIC 4. Dieses bietet erweiterte BASIC-Befehle, die die bequeme Ansprache der Floppies ermöglichen. Es ist daher für alle Umstellungen zu empfehlen. Neu sind insbesondere die Statements DOPEN (Eröffnung einer Datei auf Diskette), DCLOSE (Schließen), DSAVE und DLOAD (Programme abspeichern und laden), SCRATCH (eine Datei löschen), RENAME (umbenennen), DIRECTORY oder CATALOG (Inhaltsverzeichnis der Disketten anzeigen), APPEND (Anhängen von Daten an eine sequentielle Datei), dann RECORD für relative Files, HEADER (Diskette formatieren), COLLECT (Entfernen nicht ordentlich geschlossener Files), BACKUP (identische Kopie ziehen, Datensicherung), COPY (Files kopieren) und CONCAT (Verbinden zweier Files). - Bei den Beipielen dieses Aufsatzes werden vorwiegend Statements aus BASIC 4 benutzt.

Hardware der Floppies

Alle genannten Geräteversionen haben zwei Laufwerke (drives), die intern die Nummern 0 (rechtes Laufwerk) und 1 tragen. Jedes kann gezielt angesprochen werden. Die Floppies haben ein IEEE-488-Businterface mit genormten 24pol. Gerätestecker. Im Lieferzustand sind sie auf die Geräteadresse 8 eingestellt, mit der sie per Primäradresse vom Computer angesprochen werden. Diese Adresse kann intern verändert werden. Damit wird es möglich, mehrere Floppies an einem Bus zu betreiben.

Die eigene Intelligenz der Geräte wird durch zwei CPUs (6502 und 6504), durch Interfacebausteine sowie durch ein Betriebssystem in Festwertspeichern gewährleistet. Daneben gibt es einen RAMspeicher von 4 KB, der teils dem System und teils als Datenpuffer dient.

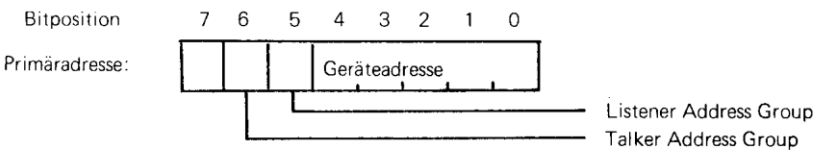
Kommandos

Der Computer spricht die Floppy mit kurzen ASCII-Strings an, die auf den 8 Datenleitungen des IEEE-Busses gesendet werden. Es gibt also keine Eigenheiten, die mit der benutzten Prozessorfamilie und ihren Befehlen zusammenhängen. M.a.W.: Jeder Prozessor kann die Floppy benutzen, wenn er ein IEEE-Interface besitzt.

Die Primäradresse

Zur Auswahl des Gerätes und seiner Funktion sendet der Computer (master) eine sog. Primäradresse und als nächstes eine Sekundäradresse. Was bedeutet das nun? Die Primäradresse enthält als wesentlichen Bestandteil die physikalische Geräteadresse (die im Gerät selbst zur Erkennung seiner Selektion eingestellt ist). Bei der Floppy ist sie im allgemeinen also '8', für Drucker verwendet man häufig die '4'.

Die Geräteadresse ist in den rechten 5 Bit der auf den Bus ausgesandten Primäradresse codiert, also



In diesen 5 Bit lassen sich also physikalische Geräteadressen von 0 bis maximal 30 unterbringen. Die Adresse 31 ist ein Sonderfall (s.u.). Ein gesetztes Bit 5 fordert das Gerät zum Zuhören auf (Listen Adress Group, LAG), ein gesetztes Bit 6 fordert das Gerät zur Datenabgabe auf (Talk Adress Group, TAG). Bit 5 und 6 dürfen also nicht zusammen gesetzt sein.

Die Primäradresse hex 3F fordert alle angeschlossenen Geräte auf, nicht mehr zuzuhören (UNLISTEN). Ein hex 5F heißt UNTALK, keine weiteren Daten abgeben. Diese Kombinationen sind die zuvor genannten Sonderfälle. Primäradressen kleiner als hex 20 werden auf dem IEEE-Bus als Adressed Command Group (hex 00 bis 0F) und als Universal Command Group benutzt (hex 10-1F).

Die Sekundäradresse

Das auf die Primäradresse folgende Byte ist die Sekundäradresse. Sie steuert am Peripheriegerät verschiedene Funktionen, bei einem Drucker z.B. die Schriftgröße oder Formatierung. Nicht alle am IEEE-Bus anschließbaren Geräte benutzen eine Sekundäradresse. Bei Betrieb der CBM-Floppy kommt sie aber regelmäßig zur Anwendung.

Bei Commodore-Rechnern dienen mit hex 'F' beginnende Sekundäradressen (also F0 bis FF) der Eröffnung einer Datei (OPEN-Statement). Sie werden bei der ersten Adressierung des Gerätes und seines Files benutzt.

Entsprechend dienen mit hex 'E' beginnende Sekundäradressen (also E0 bis EF) dem Schließen einer Datei (CLOSE).

Sekundäradressen im Bereich hex 60 bis 7F kommen dann zum Einsatz, wenn ein Datentransport vom/zum File stattfindet.

Was bedeutet nun die Sekundäradresse für die Floppy? Sie eröffnet einen Datenkanal und ordnet ihm eine Nummer zu, die das DOS der Floppy beachtet. Dazu folgendes: Die 4K-RAMspeicher der Floppy sind als 16 Datenpuffer zu je 256 Byte gegliedert. Einen Teil davon benutzt die Floppy selbst für die BAM (Verzeichnis der verfügbaren Diskettensektoren/Blocks), für den Diskcontroller und für den Kommandokanal, der die Sekundäradresse 15 (hex 'F') hat.

Die Ansprache des Kanales 0 (Sekundäradresse mit Null endend) wird automatisch als Laden eines Programmes im Zusammenhang mit DLOAD interpretiert. Kanal 1 ist dem DSAVE zugeordnet, dem Abspeichern von Programmen also. Für das Laden und Speichern von Daten stehen die Kanal-

nummern 2 bis 14 zur Verfügung (15 ist der Kommandokanal). Es dürfen bis zu 10 Kanäle gleichzeitig benutzt werden, dabei belegen eröffnete sequentielle Files jeweils 2 Datenpuffer und relative Files jeweils 3. Man kann also maximal 5 sequentielle oder 3 relative Fikes zugleich offen haben oder eine dazwischenliegende Kombination von 3 + 1 oder 2 + 2 dieser Files (DOS 2). Das DOS nimmt eine Zuordnung der erreichbaren physischen Kanäle zu den Files selbständig vor.

Bei BASIC 4 der CBM-Rechner gelten folgende Besonderheiten: Der Kommandokanal 15 ist bei Ansprache der Diskette rechnerseitig automatisch geöffnet. Bei den Statements DOPEN muß man keine Sekundäradresse angeben, BASIC 4 übernimmt die Kanalverwaltung und sendet, beginnend mit 2 für das erste eröffnete File, entsprechende Sekundäradressen aus, siehe Beispiel 9.

Die Steuerinformation

Bei der Übersendung von Kommandos und am Ende einer Übertragung wird das auf dem IEEE-Bus gesendete Byte von Steuerinformation auf anderen Leitungen begleitet, die ihm die Qualität verleihen:

ATN = LOW Es handelt sich um Kommandobytes, nämlich Primär-/Sekundäradresse bzw. um hex 3F oder 5F, UNLISTEN oder UNTALK

EOI = LOW Es handelt sich um das letzte gesendete Byte

Die drei Signale DAV (Data Valid), NRFD (Not Ready For Data) und NDAC (Data Not Accepted) sind reine Handshake-Signale, die zur Übermittlung jeder Art Bytes gehören.

Die prinzipielle Abfolge des Datenverkehrs

In der nachfolgenden Dokumentation sehen wir folgende grundsätzlichen Abläufe (H = High Level, L = Low Level):

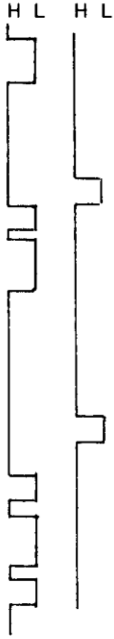
DSAVE-Befehl

Signal ATN	Signal EOI	Byte (hex)	Bedeutung
H L	H L		
		28	Primäradresse: Device 8, LISTEN, Beginn des OPEN
		F1	Sekundäradresse: OPEN Kanal 1 (SAVE-Kanal)
		30	Diskdrive 0 (30 = ASCII 0)
		3A	: Doppelpunkt als Befehlstrenner
		xx	1. ASCII-Byte des Filename
		xx	Letztes Byte des Filename mit EOI=LOW
		3F	UNLISTEN (vom Rechner), Ende der OPEN-Sequenz
		28	Primäradresse: Device 8, LISTEN
		61	Sekundäradresse: Transport auf Kanal 1
		xx	Startadresse Low des Programmes (SAL)
		xx	Startadresse High des Programmes (SAH)
		xx	Das Programm als direkter Speicherabzug. Bei BASIC-programmen Befehle also als 'tokens'
		xx	Letztes Programmbyte, Ende der Übertragung
		3F	UNLISTEN (vom Rechner)
		28	Primäradresse: Device 8, LISTEN
		E1	CLOSE Kanal 1
		3F	UNLISTEN

Man merke sich also: Das Signal ATN ist nur während der Primär- und Sekundäradresse aktiv auf LOW, ferner bei UNLISTEN und UNTALK.

Das Ende einer Übertragung, sei es nun beim OPEN des Files oder am Ende eines Datenstromes, wird vom jeweiligen Absender durch EOI = LOW gekennzeichnet. Beim nachfolgenden DLOAD stammt das erste aktive EOI vom Computer am Ende seiner OPEN-Phase. Das zweite aktive EOI stammt von der Diskette. Dieses EOI erkennend, übernimmt der Computer wieder die Führung.

DLOAD-Befehl



Signal ATN H L	Signal EOI H L	Byte (hex)	Bedeutung
		28	Primäradresse: Device 8, LISTEN, Beginn des OPEN
		F0	Sekundäradresse: OPEN Kanal 0 (Ladekanal)
		30	Diskdrive 0 (30 = ASCII 0)
		3A	: Doppelpunkt als Befehlstrenner
		xx	1. ASCII-Byte des Filename
		xx	Letztes Byte des Filename mit EOI=LOW
		3F	UNLISTEN (vom Rechner), Ende der OPEN-Sequenz
		48	Primäradresse: Device 8, TALK
		60	Sekundäradresse: Transport auf Kanal 0, Ladekanal
		xx	Aussendung von der Floppy: Startadresse Low des Programmes (SAL)
		xx	Startadresse High des Programmes (SAH)
		xx	Das Programm als direkter Speicherabzug.
		xx	Letztes Programmbyte, Ende der Übertragung von der Floppy. Die Floppy sendet EOI.
		5F	UNTALK (vom Rechner)
		28	Primäradresse: Device 8, LISTEN
		E0	CLOSE Kanal 0
		3F	UNLISTEN

Bei den vielen nachfolgenden Beispielen sehen wir vor dem Doppelpunkt als Befehlstrenner gelegentlich noch weitere Bytes, die den Befehl spezifizieren.

Den Protokollen ist weiterhin zu entnehmen, daß bei manchen Befehlen Datenströme gesendet werden (DSAVE, DLOAD, VERIFY), bei anderen einzelne Strings (PRINT INPUT) mit bzw. bis hin zum Stringtrenner OD (Carriage Return) und bei noch anderen einzelne Bytes, deren Vereinnahmung laufend durch UNTALK und neuerlichem TALK unterbrochen wird, so beim GET und bei CATALOG.

Beispiel 1: DSAVE

```
10 DSAVE"TESTPROGRAMM"
20 PRINT"ABCDEF"
30 PRINT"123456"
```

Dieser Programmtext wird auf Floppy geschrieben

Protokoll des Datenverkehres, hexadezimale Bytes

<M>=1000 28 F1 30	Device 8, LISTEN, Drive 0
< > 1004 3A 54 45 53	: (als Trenner) TES
< > 1008 54 50 52 4F	TPRO
< > 100C 47 52 41 4D	GRAM
< > 1010 4D 3F 28 61	M, UNLISTEN, Device 8, Listen, Kanal 1
< > 1014 01 04 15 04	SAL, SAH, Link
< > 1018 0A 00 D5 22	Zeilenummer, token,"
< > 101C 54 45 53 54	TEST
< > 1020 50 52 4F 47	PROG
< > 1024 52 41 4D 4D	RAMM
< > 1028 22 00 23 04	",Begrenzer, Link
< > 102C 14 00 99 22	Zeilennummer 20, token,"
< > 1030 41 42 43 44	ABCD
< > 1034 45 46 22 00	EF", Begrenzer (00)
< > 1038 31 04 1E 00	Link, Zeilennr. 30
< > 103C 99 22 31 32	token, "12
< > 1040 33 34 35 36	3456
< > 1044 22 00 00 00	", Begrenzer, Link 00 00
< > 1048 3F 28 E1 3F	UNLISTEN, CLOSE Kanal 1, UNLISTEN

Beispiel 2: VERIFY

```
VERIFY"*",8
```

Der Befehl VERIFY bewirkt einen Vergleich zwischen abgespeichertem Programm und dem noch im Speicher befindlichen. "*" als Filename nimmt auf das letzte gespeicherte Programm Bezug.

<M>=1000 28 F0 2A 3F	Device 8, LISTEN, Kanal 0 * UNLISTEN
< > 1004 48 60 01 04	Device 8, TALK, Kanal 0, SAL SAH von Floppy
< > 1008 15 04 0A 00	Programmtext wie in Beispiel 1
< > 100C D5 22 54 45	
< > 1010 53 54 50 52	
< > 1014 4F 47 52 41	
< > 1018 4D 4D 22 00	
< > 101C 23 04 14 00	
< > 1020 99 22 41 42	
< > 1024 43 44 45 46	
< > 1028 22 00 31 04	
< > 102C 1E 00 99 22	
< > 1030 31 32 33 34	
< > 1034 35 36 22 00	
< > 1038 00 00 5F 28	Ende des Programms, UNTALK, LISTEN Dev. 8
< > 103C E0 3F	CLOSE Kanal 0, UNLISTEN

Beispiel 3: DLOAD

```
DLOAD"TEST*"
```

DLOAD bewirkt das Laden des namentlich aufgerufenen Programmes. Zur Abkürzung des Namens darf ein "*" verwendet werden.

<M>=1000 28 F0 30 3A	OPEN-Sequenz: De. 8, Kanal 0, Drive 0, Trenner
< > 1004 54 45 53 54	TEST

```
< > 1008 2A 3F 48 60
< > 100C 01 04 15 04
< > 1010 0A 00 D5 22
< > 1014 54 45 53 54
< > 1018 50 52 4F 47
< > 101C 52 41 4D 4D
< > 1020 22 00 23 04
< > 1024 14 00 99 22
< > 1028 41 42 43 44
< > 102C 45 46 22 00
< > 1030 31 04 1E 00
< > 1034 99 22 31 32
< > 1038 33 34 35 36
< > 103C 22 00 00 00
< > 1040 5F 28 E0 3F
```

* UNLISTEN Dev. 8 TALK, Kanal 0
SAL SAH und Programmbytes wie in den Beipielen 1 und 2

Pgrogrammende
UNTALK, CLOSE Kanal 0, UNLISTEN

Beispiel 4: Eröffnung einer sequentiellen Datei

```
10 DOPEN#5,"DATATEST",W
20 PRINT#5,"AIM 65"     W
30 DCLOSE#5
```

Logisches File 5 auf dem Rechner, Name
W= Eröffnen zum Beschreiben. Schreibe String
Schließen der Datei

```
<M>=1000 28 F2 30 3A
< > 1004 44 41 54 41
< > 1008 54 45 53 54
< > 100C 2C 57 2C 53
< > 1010 3F 28 62 41
< > 1014 49 4D 20 36
< > 1018 35 0D 3F 28
< > 101C E2 3F
```

Dev. 8, LISTEN, OPEN Kanal 2 (vom Rechner zugeteilt), Drive 0, Befehlstrenner
DATATEST
, W, S (sequentiell)
UNLISTEN, LISTEN, Kanal 2 A
IM 6
5 (CR) UNLISTEN, CLOSE
Kanal 2, UNLISTEN

Beispiel 5: Lesen aus einer sequentiellen Datei

```
10 DOPEN#5,"DATATEST"
20 INPUT#5,A$
25 PRINTA$
30 IFST=64THEN50
40 GOTO20
50 DCLOSE#5
```

Logisches File 5 des Rechners
Holen eines Strings von der Floppy
Ausdruck auf dem Rechner
Prüfung auf Dateiende (ST=Statusvariable)

```
<M>=1000 28 F2 30 3A
< > 1004 44 41 54 41
< > 1008 54 45 53 54
< > 100C 3F 48 62 41
< > 1010 49 4D 20 36
< > 1014 35 0D 5F 28
< > 1018 E2 3F
```

Dev. 8 LISTEN, Kanal 0, Drive 0 :
DATA
TEST
UNLISTEN, Dev. 8 TALK A
IM 6
5 (CR) UNTALK CLOSE
Kanal 2, UNLISTEN

Beispiel 6: Lesen aus einer sequentiellel Datei

```
10 OPEN7,8,6,"0:DATATEST,S,R"
20 GET#7,A$
30 IFST=64THENCLOSE7:END
40 PRINTA$;
50 GOTO20
```

Formulierung z.B. nach BASIC 3: log. File 7, Device 8, Kanal 6, Drive 0, Name sequentielles Lesen. Der GET-Befehl holt 1 Zeichen. Zeile 30 prüft auf Dateiende

```
<M>=1000     28 F6 30
< > 1004 3A 44 41 54
< > 1008 41 54 45 53
< > 100C 54 2C 53 2C
< > 1010 52 3F 48 66
< > 1014 41 5F 48 66
< > 1018 49 5F 48 66
< > 101C 4D 5F 48 66
< > 1020 20 5F 48 66
< > 1024 36 5F 48 66
< > 1028 35 5F 48 66
< > 102C 0D 5F 28 E6
< > 1030 3F
```

OPEN Dev. 8, Kanal 6 Drive 0
:DAT
ATES
T,S,
R UNLISTEN TALK Dev. 8, Kanal 6
A UNTALK TALK Kanal 6
I UNTALK TALK Kanal 6
M UNTALK TALK Kanal 6
(Space) UNTALK TALK Kanal 6
6 UNTALK TALK Kanal 6
5 UNTALK TALK Kanal 6
(CR) UNTALK CLOSE Kanal 6
UNLISTEN

Beispiel 7: APPEND, Anhängen eines Strings an eine sequentielle Datei

```
10 APPEND#9,"DATATEST"
20 PRINT#9,"COMMODORE"
30 DCLOSE#9
```

Log. File 9, Name des bestehenden Files
Der anzuhängende String

```
<M>=1000 28 F2 30 3A
< > 1004 44 41 54 41
< > 1008 54 45 53 54
< > 100C 2C 41 3F 28
< > 1010 62 43 4F 4D
< > 1014 4D 4F 44 4F
< > 1018 52 45 0D 3F
< > 101C 28 E2 3F
```

Dev. 8 LISTEN Kanal 2, Drive 0 :
DATA
TEST
,A (APPEND) UNLISTEN Dev. 8 Listen
Kanal 2 COM
MODO
RE (CR) UNLISTEN
CLOSE Kanal 2 UNLISTEN

Beispiel 8: Lesen der durch APPEND erweiterten Datei

```
10 OPEN6,8,6,"DATATEST"
20 GET#6,A$
30 IFST=64THENCLOSE6:END
40 PRINTA$;
50 GOTO20
```

Programm wie im Beispiel 6

```
<M>=1000     28 F6 44
< > 1004 41 54 41 54
< > 1008 45 53 54 3F
< > 100C 48 66 41 5F
< > 1010 48 66 49 5F
< > 1014 48 66 4D 5F
< > 1018 48 66 20 5F
< > 101C 48 66 36 5F
< > 1020 48 66 35 5F
< > 1024 48 66 0D 5F
< > 1028 48 66 43 5F
< > 102C 48 66 4F 5F
< > 1030 48 66 4D 5F
< > 1034 48 66 4D 5F
< > 1038 48 66 4F 5F
< > 103C 48 66 44 5F
< > 1040 48 66 4F 5F
< > 1044 48 66 52 5F
< > 1048 48 66 45 5F
< > 104C 48 66 0D 5F
< > 1050 28 E6 3F
```

OPEN Dev. 8 Kanal 6, kein Drive D
ATAT
EST UNLISTEN
TALK Kanal 6 A UNTALK usw.

I
M
(Space)
6
5
(CR)
C
O
M
M
O
D
O
R
E
(CR) UNTALK
CLOSE Kanal 6 UNLISTEN

Beispiel 9: Eröffnung mehrerer Datafiles

```
 10 DOPEN#2,"DATA1"
 20 DOPEN#3,"DATA2"
 30 DOPEN#4,"RELATIVE"
```

```
<M>=1000 28 F2 30 3A
< > 1004 44 41 54 41
< > 1008 31 3F 28 F3
< > 100C 30 3A 44 41
< > 1010 54 41 32 3F
< > 1014 28 F4 30 3A
< > 1018 52 45 4C 41
< > 101C 54 49 56 45
< > 1020 3F
```

In den 3 OPEN-Sequenzen weist BASIC 4 automatisch die Datenkanäle 2, 3 und 4 zu (F2, F3, F4)

Beispiel 10: Anlegen einer relativen Datei

```
 10 DOPEN#5,"RELATIVE",L10
 20 RECORD#5,200
 30 PRINT#5,"ABCDEFGHIJK"
 40 DCLOSE#5
READY.
```

log. File 5, Name, Länge: 10 Bytes je Datensatz
Positionieren auf Satz 200
Durch dieses erste Beschreiben werden 200 Datensätze angelegt (man schreibt also in den letzten Satz)

```
<M>=1000 28 F2 30 3A
< > 1004 52 45 4C 41
< > 1008 54 49 56 45
< > 100C 2C 4C 2C 0A
< > 1010 3F 28 6F 50
< > 1014 62 C8 00 01
< > 1018 3F 28 62 41
< > 101C 42 43 44 45
< > 1020 46 47 48 49
< > 1024 4A 4B 0D 3F
```

Dev. 8 LISTEN, Kanal 2, Drive 0 :
RELA
TIVE
,L (Länge) , 0A=10 (Satzlänge)
UNLISTEN, LISTEN Kommandokanal 15, P
b (offensichtl. RECORD- Befehl) C8 00=200, Byte 1
UNLISTEN LISTEN KANAL 2, A
BCDE
FGHI
JK (CR) UNLISTEN

Beispiel 11: Lesen aus angewähltem Datensatz in einer relativen Datei

```
 10 DOPEN#5,"RELATIVE"
 20 RECORD#5,200,4
 30 GET#5,A$
 35 PRINTA$
 40 DCLOSE#5
READY.
```

Positioniere auf Datensatz 200, dort Byte 4
Hole 1 Byte

```
<M>=1000 28 F2 30 3A
< > 1004 52 45 4C 41
< > 1008 54 49 56 45
< > 100C 3F 28 6F 50
< > 1010 62 C8 00 04
< > 1014 3F 48 62 44
< > 1018 5F 28 E2 3F
```

OPEN Dev. 8, Kanal 2, Drive 0 :
RELA
TIVE
UNLISTEN, Dev. 8 Kommandokanal
Pb=positioniere C8 00=200, Byte 4
UNLISTEN TALK Kanal 2, es kommt das 'D'
UNTALK CLOSE Kanal 2 UNLISTEN

Beispiel 12: Verbindung zweier Dateien mit CONCATENATE

```
 10 CONCAT D0,"DATATEST"TO D0,"DATA"
READY
```

DATATEST soll an DATA angehängt werden

```
<M>=1000 3F 28 6F 43      Dev. 8, Kommandokanal C (Concatenate)
< > 1004 30 3A 44 41      Drive 0 : DA
< > 1008 54 41 3D 30      TA= Drive 0
< > 100C 3A 44 41 54      : DAT
< > 1010 41 2C 30 3A      TA= Drive 0 :
< > 1014 44 41 54 41      DATA
< > 1018 54 45 53 54      TEST
< > 101C 3F               UNLISTEN
```

Beispiel 13: Löschen einer Datei mit SCRATCH

```
 10 SCRATCH"DATA",D0      Befehl, Name, Drive 0
READY

<M>=1000 28 6F 53 30      Dev. 8, Kommandocanal S0 (Scratch Drive 0)
< > 1004 3A 44 41 54      : DAT
< > 1008 41 3F            A UNLISTEN
```

Beispiel 14: Kopieren einer Datei mit COPY

```
 10 COPY"DATA1"TO"DATA2"  bestehende Datei DATA 1
                          als DATA2 kopieren
READY.

<M>=1000 28 6F 43 30      Dev. 8, Kommandocanal C 0 =COPY Drive 0
< > 1004 3A 44 41 54      : DAT
< > 1008 41 32 3D 30      A2, Drive 0
< > 100C 3A 44 41 54      : DAT
< > 1010 41 31 3F         A1 UNLISTEN
```

Beispiel 15: Entfernen nicht ordnungsmäßig geschlossener Dateien mit COLLECT

```
 10 COLLECT D0            Drive 0 ist angesprochen
READY.

<M>=1000 28 6F 56 30      Device 8, Kommandokanal V=VALIDATE
< > 1004 3F               Drive 0 UNLISTEN
```

Beispiel 16: Ziehen einer identischen Kopie der Diskette mit BACKUP

```
 10 BACKUP D0 TO D1       Original in Drive 0, Kopie in 1
READY.

<M>=1000 28 6F 44 31      Dev. 8, Kommandokanal D=DUPLICATE
< > 1004 3D 30 3F         Drive 1 = Drive 0 UNLISTEN
```

Beispiel 17: Umbenennung einer Datei mit RENAME

```
 10 RENAME "DATA1" TO "DEMO"   Der neue Name ist DEMO
READY.

<M>=1000 28 6F 52 30      Device 8, Kommandokanal R=RENAME, Drive 0
< > 1004 3A 44 45 4D      : DEM
< > 1008 4F 3D 30 3A      O= Drive 0 :
< > 100C 44 41 54 41      DATA
< > 1010 31 3F            1 UNLISTEN
```

Beispiel 18: Formatieren einer Diskette (NEW)

```
    HEADER"IEETEST",D0,I15          Befehl, Diskettenname, Drive 0, Identität 15
READY

<M>=1000 28 6F 4E 30                Dev. 8, Kommandokanal N=NEW Drive 0
< > 1004 3A 49 45 45                :IEE
< > 1008 54 45 53 54                TEST
< > 100C 2C 31 35 3F                ,15 UNLISTEN
< > 1010 48 6F 30 30                TALK Kommandokanal 00 (Rückmeldung)
< > 1014 2C 20 4F 4B                , OK
< > 1018 2C 30 30 2C                , 00,
< > 101C 30 30 0D 5F                00 (CR) UNTALK
```

Anhang

Die CBM-Floppies verstehen nicht nur die genannten Befehle, sondern auch weitere, die der interessierte Lesser den entsprechenden Handbüchern entnehmen möge. Man kann gezielt einzelne Diskettensektoren lesen und beschreiben. Der BLOCK-EXECUTE-Befehl erlaubt es, ein erweitertes DOS auf der Diskette zu unterhalten und als Programm im Datenpuffer ablaufen zu lassen. Ferner kann der Pufferzeiger der Kanäle, der auf das nächste anzusprechende Byte zeigt, für gezielte Zugriffe umgesetzt werden, um z.B. auf bestimmte Datenfelder aufzusetzen. Man kann gezielt in das Memory der Diskette einschreiben, es auslesen und sogar Unterprogramme dort ablaufen lassen, die die Kontrolle mit RTS an das DOS zurückgeben.Und schließlich gibt es anwenderbestimmte Kommandos, die nach einer Standard-Sprungleiste ausführen.

R.L.

Robert Rudolph, 2000 Hamburg 63

PRINTUSING für CBM

In kommerziell eingesetzten CBM-Konfigurationen wird häufig statt eines Matrixdruckers der CBM-Familie ein Typenraddrucker eingesetzt. Bei der Ausgabe umfangreicher Tabellen kann daher nicht auf die komfortablen Formatierfähigkeiten der CBM-Drucker zugegriffen werden. Eine PRINTUSING-Routine, wie sie in vielen BASIC-Dialekten existiert (etwa TRS 80 L2) wird benötigt.

Für den CBM gibt es seit kurzem BASIC-Erweiterungen auf dem Markt, die auch ein PRINTUSING enthalten, etwa DISC-O-PRO von Skyles Electric. Die dem Autor bekannten Versionen haben jedoch zwei manchmal entscheidende Einschränkungen:

- Zahlen, die betragsmäßig kleiner als 0.01 sind, werden falsch verarbeitet
- Anzahl von Formatfeldern und zu druckenden Werten müssen gleich sein.

Das im Folgenden vorgestellte Programm beseitigt beide Einschränkungen. Es ist aus Geschwindigkeitsgründen in Assembler (MAE) geschrieben und läuft auf CBM 30xx, 40xx und 80xx mit jeweils nur geringfügigen Änderungen (Umstellen einiger Systemadressen, siehe Programmlisting).

Aufrufsyntax:

PRINTUSING wird wie folgt aufgerufen (optionale Teile in `[..]`)

```
SYS(PU)[#],F$,D$[(I)][,FP[,MA]]
```

Hierbei bedeuten im einzelnen:

PU — Adresse, ab der die Maschinenroutine PRINTUSING abgelegt worden ist, im Beispiellisting also 2053 bzw. 8275 dezimal.

d — Dateinummer, wenn IEEE-Gerät angesprochen werden soll. Wie bei dem BASIC PRINT-Befehl darf '#d' fehlen, dann wird das durch CMD eingestellte Gerät (bzw. der Bildschirm) genommen.

F$ — String (oder Stringausdruck), der die Formatanweisungen enthält. Die Syntax der Formatanweisungen ist weiter unten erklärt.

D$(i) — String (Ausdruck oder Feld), der zu druckende Werte enthält. Ist MA gleich 1, darf D$(i) ein einfacher String(ausdruck) sein. Ist MA >1 und enthält F$ mehr als ein Formatfeld (s.u.), so muß D$(i) ein Stringfeld sein - es werden dannn soviele Feldelemente (ab einschließlich Index i) gedruckt, wie F$ (noch abzuarbeitende) Formatfelder enthält oder bis MA-Werte gedruckt worden sind.

FP — Formatpointer: Spalte im Formatstring, ab der das Format bearbeitet werden soll. Fehlt FP, wird 0 angenommen. Dann werden alle Formatfelder abgearbeitet. Wenn FP angegeben wurde, enthält FP am Ende des PRINTUSING-Befehls die der zuletzt verarbeiteten folgende Spalte. Damit ist eine lückenlose Weiterverarbeitung des Formats Wert für Wert möglich.

MA — Maximale Anzahl zu druckender Strings. Fehlt bereits FP, wird 99 angenommen. Existiert FP, aber fehlt MA, wird MA auf 1 gesetzt.

Fehlt FP, wird beim Erreichen des Formatendes ein CR ($0D) gedruckt, aber kein LF ($0A). Ist FP angegeben, wird auch beim Erreichen des Formatendes kein CR ausgegeben.

Der Formatstring F$ dient als Ausgabemaske. Grundsätzlich wird für jedes Zeichen in F$ genau ein Zeichen gedruckt. Hierbei werden alle Zeichen außer '@' und '#' kopiert. '@' und '#' fungieren hier als Steuerzeichen und bewirken, daß Werte aus D$(i) statt '@' bzw. '#' gedruckt werden.

Eine lückenlose Folge von n mal '@' definiert ein Stringfeld der Länge n. Vor dem entsprechenden D$-Element werden die ersten n Zeichen eingesetzt; ggfs. wird mit Blanks aufgefüllt.

Eine lückenlose Folge von '#' definiert ein Zahlenfeld ohne Dezimalpunkt. Der Wert des entsprechenden D$-Elements wird ohne Dezimalpunkt rechtsbündig ausgedruckt. Negative Vorzeichen werden ausgegeben, positive nicht.

Eine Folge von '#', die genau einmal '.' enthält, definiert ein Zahlenfeld mit Dezimalpunkt. '.' darf hierbei auch als erstes oder letztes Zeichen der Folge auftreten. Der Wert des entsprechenden D$-Elements wird in üblicher Dezimalbruchschreibweise ausgegeben, der Dezimalpunkt steht hierbei an der im Formatstring bezeichneten Stelle.

Bei Überlauf des Formatfeldes gibt der Rechner '*' im ganzen Formatfeld aus.

Etwaige Unklarheiten in Aufruf- und Formatsyntax wird hoffentlich das kleine BASIC-Beispielprogramm beseitigen.

Noch ein Hinweis für Betreiber anderer 6502-Rechner: Die Anpassung des Programms ist grundsätzlich möglich, setzt aber wegen der Vielzahl der Einsprünge in das Betriebssystem eine sehr gute Kenntnis des Betriebssystems voraus.

Wer sich die Mühe des Abtippens sparen möchte, kann vom Autor eine Diskette (CBM 3040/4040) mit allen Quell- und Maschinencodes beziehen durch Überweisung von DM 25,-- auf das Postscheckkonto Hamburg, Nr. 447618-208.

```
0010 ; ********************************
0020 ; *                              *
0030 ; *         BSLABELS.4           *
0040 ; *                              *
0050 ; *  BETRIEBSSYSTEMEINSPRUENGE   *
0060 ; *  FUER BETRIEBSSYSTEM 4       *
0070 ; *                              *
0080 ; *  (80 - ZEICHEN-BILDSCHIRM)   *
0090 ; *                              *
0100 ; *                              *
0110 ; ********************************
0120 TALK        .DE $F770            ; SETZE IEEE-EINGABE
0130             ; MIT DATEINUMMER IN X
0140 UNTALK      .DE 61870
0150             ; ALLE IEEE-EINGABEGERAETE ABSCHALTEN
0160 LISTEN      .DE $FFC9            ;AUSGABE SETZEN
0170 UNLISTEN    .DE $FFCC            ;NORMAL - I/O
0180 BWERT       .DE $BD98            ; AUS BASIC-AUSDRUCK IN FA
0190             ; WERT AUS BASIC-AUSDRUCK IN FAC SPEICHERN
0200 GARCOL      .DE 45984            ; REINIGT STRINGBEREICH
0210 CHGET       .DE $70              ; HOLT 1 BASIC-ZEICHEN
0220 CHGOT       .DE $76              ; HOLT ALTES BASIC-ZEICHEN
0230 INSERT      .DE $B350            ; DIVERSE PARAMETER
0240 CURSOR      .DE 196              ; DIE CURSORPOSITION
0250 FACSPE      .DE 52490
0260             ; FAC IN SPEICHER, IN (X,Y) STEHT DIE ADRESSE
0270 PRINTCHAR   .DE $FFD2            ; 1 ZEICHEN DRUCKEN
0280 VASUC       .DE 49451
0290             ; HOLT VARIABLENPOINTER AUS BASIC
0300 STRFAC      .DE $C8EB
0310 MUL10       .DE 52248            ; FAC := FAC * 10
0320 ADD0.5      .DE 51583            ; FAC := FAC + 0.5
0330 FACSTR      .DE $CF93
0340             ; FAC IN STRING AB $0100 WANDELN
0350 STRDR1      .DE 47908
0360             ; STRING AB ($1F,$20) MIT LAENGE X DRUCKEN
0370 INTFAC      .DE 50364
0380             ; WANDELT (A,Y) (A=HIGH BYTE) IN FAC
0390 SPEFAC      .DE $CCD8
0400             ; HOLT SPEICHERWERT AUS (A,Y) INS FAC
0410 INT         .DE $CE02
0420             ;SCHNEIDET NACHKOMMASTELLEN AB
0430 FLINT       .DE 49898            ; WANDELT FAC IN INTEGER
0440 ; ********************************
0450 ; *                              *
0460 ; *  WESENTLICHE BS-VARIABLEN    *
0470 ; *                              *
0480 ; *  (WERDEN VON BS-ROUTINEN     *
0490 ; *  ALS PARAMETER BENUTZT)      *
0500 ; *                              *
0510 ; *  VERSION BS 4                *
0520 ; *                              *
0530 ; *                              *
0540 ; ********************************
0550 VAADR       .DE 68               ;  ERGEBNIS VON VASUC
0560 STGRE       .DE 48
0570             ;  UNTERGRENZE STRINGBEREICH (= LETZTER DEF. ST
0580 VAGRE       .DE 46
0590             ; OBERE GRENZE VARIABLENLISTE (ENDE FELDER)
0600 INADR       .DE $61
0610             ;INTEGERWERT (HIGH) NACH WANDLUNG DURCH FLINT)
0620 HILF        .DE 203
0630             ; HILFSADRESSE IN ZEROPAGE FUER ALLES MOEGLICHE
```

```
                    0640 ; ********************************
                    0650 ; *                              *
                    0660 ; *    WICHTIGE HILFSROUTINEN    *
                    0670 ; *                              *
                    0680 ; *                              *
                    0690 ; *    ALS INTERFACE ZUM BS      *
                    0700 ; *                              *
                    0710 ; *    VERSION:  BS 4            *
                    0720 ; *                              *
                    0730 ; *                              *
                    0740 ; *                              *
                    0750 ; *                              *
                    0760 ; ********************************
2000- 20 70 00      0770 GETVA      JSR CHGET             ; HOLT VARIABLENPOINTER UN
2003- 20 2B C1      0780            JSR VASUC
2006- A0 00         0790            LDY #0
2008- 60            0800            RTS
                    0810            ;
2009- 20 70 00      0820 WERT       JSR CHGET
                    0830            ; HOLT 1 BYTE-AUSDRUCK AUS BASIC
200C- 20 98 BD      0840            JSR BWERT
200F- 20 EA C2      0850 WERTPLUS6  JSR FLINT
2012- A5 62         0860            LDA *INADR+1
2014- 60            0870            RTS
                    0880            ;
2015- 85 CB         0890 VASAVE     STA *HILF
                    0900            ; VAADR HOLEN UND IN ZP-ADRESSE,
                    0910            ;DIE IN A STEHT, RETTEN
2017- 20 00 20      0920            JSR GETVA
201A- 84 CC         0930            STY *HILF+1
201C- A5 44         0940            LDA *VAADR
201E- 91 CB         0950            STA (HILF),Y
2020- C8            0960            INY
2021- A5 45         0970            LDA *VAADR+1
2023- 91 CB         0980            STA (HILF),Y
2025- 60            0990            RTS
                    1000            ;
2026- 85 CB         1010 STRHOL     STA *HILF
                    1020            ; STRING HOLEN (AUCH AUSDRUCK)
2028- 20 70 00      1030            JSR CHGET             ;TRENNZ. UEBERLESEN
202B- 20 98 BD      1040            JSR BWERT             ; STRINGPTR IN (97/98)
202E- A5 61         1050            LDA *97
2030- 85 44         1060            STA *VAADR
2032- A5 62         1070            LDA *98
2034- 85 45         1080            STA *VAADR+1
2036- A0 00         1090            LDY #0
2038- 84 CC         1100            STY *HILF+1
203A- B1 44         1110 STRL1      LDA (VAADR),Y
203C- 91 CB         1120            STA (HILF),Y
203E- C8            1130            INY
203F- C0 03         1140            CPY #3
2041- D0 F7         1150            BNE STRL1
2043- A9 16         1160            LDA #22
2045- 8D 13 00      1170            STA 19                ; LOESCHT STRING-STACK
2048- 60            1180            RTS
2049- A5 44         1190 WERTGOT    LDA *VAADR            ; HOLT REAL-ADRESSE
204B- A4 45         1200            LDY *VAADR+1
204D- 20 D8 CC      1210            JSR SPEFAC
2050- 4C 0F 20      1220            JMP WERTPLUS6         ; WEITER IN WERT
                    0080            .FI "ZEROPAGE"


02C6  3108-33CE  ZEROPAGE
```

```
                0010 ; ********************************
                0020 ; *                              *
                0030 ; *                              *
                0040 ; *         ZEROPAGE             *
                0050 ; *                              *
                0060 ; *  ADRESSEN BIS $FF, DIE       *
                0070 ; *  VOM BS FREUNDLICHERWEISE    *
                0080 ; *  NICHT IMMER BENOETIGT       *
                0090 ; *  WERDEN UND DAHER RELATIV    *
                0100 ; *  UNKTITISCH VERWENDBAR SIND  *
                0110 ; *                              *
                0120 ; *                              *
                0130 ; ********************************
                0140 ENDFLAG    .DE 25
                0150 MAXANZ     .DE 26
                0160 SCANADR    .DE 199
                0170 LAENGE     .DE 27
                0180 SLAENGE    .DE 28
                0190 ANZNULL    .DE 29
                0200 FORMAT     .DE 180              ; BIS 182
                0210 DATEN      .DE 183
                0220 SCANPTR    .DE 190
                0230 PUNKTPOS   .DE 192
                0240 BANZ       .DE 186
                0250 IEEEAUS    .DE 189
                0090            .FI "PRINTUSING"

1DF9  3108-4F01  PRINTUSING

                0010 ;
                0020 ;
                0030 ; ********************************
                0040 ; *                              *
                0050 ; *                              *
                0060 ; *                              *
                0070 ; *         PRINTUSING           *
                0080 ; *                              *
                0090 ; *                              *
                0100 ; *  FORMATIERTE AUSGABE VON     *
                0110 ; *  ZAHLEN FUER C64-RECHNER      *
                0120 ; *                              *
                0130 ; *                              *
                0140 ; *  COPYRIGHT: R. RUDOLPH       *
                0150 ; *             LENTERSWEG 46    *
                0160 ; *             2000 HAMBUEG 63  *
                0170 ; *                              *
                0180 ; ********************************
                0190 ;
                0200 ;
2053- A2 00     0210 PU.ENTRY   LDX #0               ;FUER IEEEAUS
2055- 20 76 00  0220            JSR CHGOT
2058- C9 23     0230            CMP #'#
205A- D0 09     0240            BNE BILDSCHIRM       ; NORMAL-AUSGABE NEHMEN
205C- 20 09 20  0250            JSR WERT             ; HOLT DATEINUMMER AUS BAS
205F- AA        0260            TAX
2060- 20 C9 FF  0270            JSR LISTEN           ; SETZT AUSGABE-DATEI
2063- A2 FF     0280            LDX #255
2065- 86 BD     0290 BILDSCHIRM STX *IEEEAUS
2067- A9 B4     0300            LDA #FORMAT
2069- 20 26 20  0310            JSR STRHOL
206C- A9 B7     0320            LDA #DATEN
206E- 20 15 20  0330            JSR VASAVE
2071- A0 00     0340            LDY #0
2073- 84 BE     0350            STY *SCANPTR
2075- 84 19     0360            STY *ENDFLAG
2077- A0 63     0370            LDY #99
2079- 84 1A     0380            STY *MAXANZ
```

```
207B- 20 76 00  0390            JSR CHGOT
207E- C9 2C     0400            CMP #',
2080- D0 1B     0410            BNE FORMATSCAN
2082- 85 19     0420 TEILFORMAT STA #ENDFLAG
2084- A9 C7     0430            LDA #SCANADR
2086- 20 15 20  0440            JSR VASAVE
2089- 20 49 20  0450            JSR WERTGOT
                0460 ; HOLT DEN WERT VON MIT VAGET GESICHERTER VARIABLEN
208C- 85 BE     0470            STA #SCANPTR
208E- 20 76 00  0480            JSR CHGOT
2091- A0 01     0490            LDY #1                  ;MAXANZ VOREINSTELLUNG
2093- C9 2C     0500            CMP #',
2095- D0 04     0510            BNE STAMAXANZ
2097- 20 09 20  0520            JSR WERT
                0530 ; HOLT ANZAHL ZU DRUCKENDE ELEMENTE AUS BASIC
209A- A8        0540            TAY
209B- 84 1A     0550 STAMAXANZ  STY #MAXANZ
209D- A4 BE     0560 FORMATSCAN LDY #SCANPTR
                0570 ; IN Y UM OFFSET FUER INDIR. ZUGRIFF ZU HABEN
209F- C4 B4     0580            CPY #FORMAT
20A1- 90 07     0590            BCC PLUS1               ; FORMAT ABGEARBEITET
20A3- A0 00     0600            LDY #0
20A5- 84 BE     0610            STY #SCANPTR            ;FORMAT RUECKSETZEN
20A7- 4C 27 22  0620 ENDE1      JMP ENDE
20AA- B1 B5     0630 PLUS1      LDA (FORMAT+1),Y
20AC- C9 23     0640            CMP #'#
20AE- F0 1C     0650            BEQ ZAHL
20B0- C9 40     0660            CMP #'@
20B2- D0 03     0670            BNE P2
20B4- 4C D0 21  0680            JMP STRING
20B7- C9 2E     0690 P2         CMP #'.
20B9- D0 09     0700            BNE AUSGABE
20BB- C8        0710            INY
20BC- B1 B5     0720            LDA (FORMAT+1),Y
20BE- C9 23     0730            CMP #'#
20C0- F0 09     0740            BEQ ZAHL2
20C2- B1 B5     0750            LDA (FORMAT+1),Y
20C4- 20 D2 FF  0760 AUSGABE    JSR PRINTCHAR
20C7- E6 BE     0770            INC #SCANPTR
20C9- D0 D2     0780            BNE FORMATSCAN          ; UNBEDINGTER SPRUNG
20CB- 88        0790 ZAHL2      DEY                     ; AUF '.' ZURUECKSTELLEN
20CC- A2 00     0800 ZAHL       LDX #00                 ; ZAEHKLT FORMATFELDLAENGE
20CE- 86 C0     0810            STX #PUNKTPOS
20D0- C6 1A     0820            DEC #MAXANZ
20D2- 30 D3     0830            BMI ENDE1
20D4- B1 B5     0840 ZAHL1      LDA (FORMAT+1),Y
20D6- C9 23     0850            CMP #'#
20D8- F0 08     0860            BEQ FELD1
20DA- C9 2E     0870            CMP #'.
20DC- D0 0A     0880            BNE ZAHLAUS
                0890 ; JETZT PUNKTPOSITION GEFUNDEN = POS. DES PUNKTS IM FELD
20DE- E8        0900            INX
20DF- 86 C0     0910            STX #PUNKTPOS
20E1- CA        0920            DEX
20E2- E8        0930 FELD1      INX
20E3- C8        0940            INY
20E4- C4 B4     0950            CPY #FORMAT
20E6- D0 EC     0960            BNE ZAHL1
20E8- 84 BE     0970 ZAHLAUS    STY #SCANPTR
                0980 ; IN X: LAENGE DES DATENFELDS
                0990 ; IN PUNKTPOS: STELLE DES DEZIMALPUNKTS (=0: KEIN DEZIMALP
                1000 ; JETZT KOMMT:
                1010 ;
                1020 ; - DATENSTRING ZUGREIFEN
                1030 ; -    -"-      IN ZAHL WANDELN
                1040 ; -    -"-      MIT 10↑ ANZ. NACHKOMMASTELLEN MULTIPLIZIER
                1050 ; - REST NACH '.' ABSCHNEIDEN (INT-FUNKTION AUS  BASIC)
                1060 ; - IN STRING ZURUECKWANDELN
```

```
                    1070 ; - FORMATIEREN ('0', '.' UND ' ' GEEIGNET EINFUEGEN)
                    1080 ; - ADRESSE UND LAENGE VON ERGEBNISSTRING SETZEN
                    1090 ; - ERGEBNISSTRING AUSGEBEN
                    1100 ; - ENDE ZAHLFORMAT: SPRUNG IN SCAN-ROUTINE
                    1110 ;
                    1120 ;
                    1130 ;DATENSTRING: IN DATEN+1 STEHT STRINGPTR: AUFRUF VAL-ROUTI
20EA- 86 1B         1140            STX #LAENGE
20EC- 20 0A 22      1150            JSR NEXTDATA
20EF- 8A            1160            TXA                     ;LAENGE -> ACCU
20F0- 20 EB C8      1170            JSR STRFAC
                    1180 ; WANDELT IN ZAHL, DIE IN FAC ABGELEGT WIRD
                    1190 ;JETZT ANZ. NACHKOMMASTELLEN BERECHNEN
20F3- A5 C0         1200            LDA #PUNKTPOS
20F5- F0 13         1210            BEQ KEINPUNKT
20F7- A5 1B         1220            LDA #LAENGE
20F9- 38            1230            SEC
20FA- E5 C0         1240            SBC #PUNKTPOS
20FC- 85 CB         1250            STA #HILF
20FE- 85 1D         1260            STA #ANZNULL
2100- C6 CB         1270 MSCHLEIFE  DEC #HILF
2102- 30 06         1280            BMI KEINPUNKT
2104- 20 18 CC      1290            JSR MUL10               ; FAC := FAC * 10
2107- 4C 00 21      1300            JMP MSCHLEIFE
210A- 20 7F C9      1310 KEINPUNKT  JSR ADD0.5              ; FAC := FAC + 0.5
210D- 20 02 CE      1320            JSR INT                 ; NACHKOMMASTELLEN ABSCHNE
                    1330 ; JETZT IN STRING ZURUECKWANDELN
2110- 20 93 CF      1340            JSR FACSTR
                    1350 ; ACHTUNG: E-FORMAT BEI FAC > 1E9
                    1360 ; DIES FUEHRT BEI ZU GROSSEN ZAHLEN ZU FEHLERN
                    1370 ; ZAHLENSTRING LIEGT AB $0100
                    1380 ; ABSCHLUSS DES STRINGS MIT ENDE-KENNUNG $00!
                    1390 ; JETZT STRINGLAENGE ERRECHNEN
2113- A2 00         1400            LDX #0
2115- A9 00         1410            LDA #0
2117- E8            1420 SSCHLEIFE  INX
2118- DD 00 01      1430            CMP $0100,X
211B- D0 FA         1440            BNE SSCHLEIFE
                    1450 ;IN X: LAENGE DES STRINGS MIT VORZEICHEN
211D- 86 1C         1460            STX #SLAENGE            ; STRINGLAENGE AKTUELLE ZA
                    1470 ; ANZAHL NACHKOMMA NULLEN BESTIMMEN
211F- A5 1D         1480            LDA #ANZNULL
2121- 18            1490            CLC
2122- 69 02         1500            ADC #2
2124- E5 1C         1510            SBC #SLAENGE
2126- 85 1D         1520            STA #ANZNULL
2128- A5 C0         1530            LDA #PUNKTPOS
212A- F0 71         1540            BEQ OHNEPUNKT
                    1550 ; NUR BLANKS UND ZIFFERN AUSGEBEN !
212C- A5 1B         1560            LDA #LAENGE             ; BERECHNEN #BLANKS
212E- 18            1570            CLC
212F- E5 1C         1580            SBC #SLAENGE
2131- B0 02         1590            BCS L11
2133- A9 00         1600            LDA #0
2135- 85 BA         1610 L11        STA #BANZ               ; ANZAHL BLANKS
2137- AA            1620            TAX
2138- E8            1630            INX
2139- E8            1640            INX
213A- E4 C0         1650            CPX #PUNKTPOS           ;MAXIMUM #BLANKS BER.
213C- 90 07         1660            BCC L2
213E- A6 C0         1670            LDX #PUNKTPOS
2140- CA            1680            DEX
2141- F0 01         1690            BEQ L22
2143- CA            1700            DEX
2144- 8A            1710 L22        TXA
2145- 85 BA         1720 L2         STA #BANZ
                    1730 ; JETZT DRUCKSCHLEIFE: LAENGE MAL JE EIN ZEICHEN DRUCKEN
```

```
2147- A0 00     1740            LDY #0
2149- A2 00     1750            LDX #0
                1760 ; X ENTHAELT ANZAHL BEREITS GEDRUCKTE ZEICHEN
                1770 ; JETZT DRUCKEN   M I T   DEZIMALPUNKT
214B- C4 BA     1780 DSCHLEIFE  CPY *BANZ               ; BEGINN DRUCKSCHLEIFE
214D- B0 04     1790            BCS KEINBLANK           ; ALLE BLANKS GEDRUCKT
214F- A9 20     1800            LDA #$20                ; ASCII-CODE FUER ' ' LADE
2151- 90 3F     1810            BCC DRUCK               ; UNBED. SPRUNG
2153- E0 00     1820 KEINBLANK  CPX #0                  ; OB VORZEICHEN ZU DRUCKEN
2155- D0 1B     1830            BNE DZIFFER             ; KEIN VORZEICHEN DRUCKEN
2157- A5 1B     1840            LDA *LAENGE
2159- E8        1850            INX
215A- C5 1C     1860            CMP *SLAENGE            ; OB VORZEICHEN PLATZ HAT
215C- F0 14     1870            BEQ DZIFFER             ; SCHON ZIFFER DRUCKEN
215E- 08        1880            PHP
215F- A5 C0     1890            LDA *PUNKTPOS
2161- C9 01     1900            CMP #1
2163- D0 07     1910            BNE VZDRUCK
                1920            ;PUNKT AN 1. POSITION!
2165- 28        1930            PLP
2166- CA        1940            DEX
2167- 90 3E     1950            BCC STERNDRUCK
2169- E8        1960            INX
216A- B0 06     1970            BCS DZIFFER
216C- 28        1980 VZDRUCK    PLP
216D- CA        1990            DEX
216E- B0 02     2000            BCS DZIFFER
                2010 ; LAENGE > ZAHLLAENGE - VORZEICHEN DRUCKEN
2170- 90 35     2020            BCC STERNDRUCK
2172- C8        2030 DZIFFER    INY
2173- C4 C0     2040            CPY *PUNKTPOS
2175- 08        2050            PHP
2176- 88        2060            DEY
2177- 28        2070            PLP
2178- D0 05     2080            BNE L3                  ; NORMALE ZIFFER DRUCKEN
217A- A9 2E     2090            LDA #'.
217C- 4C 92 21  2100            JMP DRUCK
217F- 90 09     2110 L3         BCC L31                 ; NOCH VORKOMMA
2181- C6 1D     2120            DEC *ANZNULL
2183- 30 05     2130            BMI L31
2185- A9 30     2140 L30        LDA #'0
2187- 4C 92 21  2150            JMP DRUCK
218A- E4 1C     2160 L31        CPX *SLAENGE
218C- E8        2170            INX
218D- B0 F6     2180            BCS L30
                2190 ; MIT '0 RECHTS AUFFUELLEN
218F- BD FF 00  2200            LDA $FF,X
2192- 20 D2 FF  2210 DRUCK      JSR PRINTCHAR
2195- C8        2220            INY                     ; DA 1 CHAR GEDRUCKT WURDE
2196- C4 1B     2230            CPY *LAENGE
2198- 90 B1     2240            BCC DSCHLEIFE           ; NAECHSTES ZEICHEN DRUCKE
219A- 4C 9D 20  2250            JMP FORMATSCAN          ; ZUR GROSSEN DRUCKSCHLEIF
219D- 8A        2260 OHNEPUNKT  TXA
219E- A2 01     2270            LDX #1
                2280 ; INITIALISIERT ZAEHLER STRINGZEICHEN AB $0100
21A0- 18        2290            CLC
21A1- E5 1B     2300            SBC *LAENGE
21A3- F0 1D     2310            BEQ L41                 ; ALLES OK
21A5- 90 0C     2320            BCC L40
21A7- A9 2A     2330 STERNDRUCK LDA #'*
21A9- 20 D2 FF  2340            JSR PRINTCHAR
21AC- C6 1B     2350            DEC *LAENGE
21AE- D0 F7     2360            BNE STERNDRUCK
21B0- 4C 9D 20  2370            JMP FORMATSCAN
21B3- A4 1B     2380 L40        LDY *LAENGE
21B5- C4 1C     2390 L401       CPY *SLAENGE
21B7- F0 0A     2400            BEQ L42
21B9- A9 20     2410            LDA #$20                    LADEN
```

```
21BB- 20 D2 FF  2420            JSR PRINTCHAR
21BE- 88        2430            DEY
21BF- 4C B5 21  2440            JMP L401
21C2- E8        2450 L41        INX
21C3- BD FF 00  2460 L42        LDA $FF,X
21C6- 20 D2 FF  2470            JSR PRINTCHAR
21C9- E4 1C     2480            CPX *SLAENGE
21CB- 90 F5     2490            BCC L41
21CD- 4C 9D 20  2500            JMP FORMATSCAN
                2510            ;
21D0- A2 00     2520 STRING     LDX #00             ; ZAEHLT FORMATFELDLAENGE
21D2- C6 1A     2530            DEC *MAXANZ
21D4- 30 51     2540            BMI ENDE
21D6- B1 B5     2550 STRING1    LDA (FORMAT+1),Y
21D8- C9 40     2560            CMP #'@
21DA- D0 06     2570            BNE STRINGAUS
21DC- E8        2580            INX
21DD- C8        2590            INY
21DE- C4 B4     2600            CPY *FORMAT
21E0- D0 F4     2610            BNE STRING1
21E2- 84 BE     2620 STRINGAUS  STY *SCANPTR
21E4- 86 1B     2630            STX *LAENGE
21E6- 20 0A 22  2640            JSR NEXTDATA
21E9- A5 1B     2650            LDA *LAENGE
21EB- 38        2660            SEC
21EC- E5 1C     2670            SBC *SLAENGE
21EE- 85 BA     2680            STA *BANZ           ; # BLANKS HINTENDRAN
21F0- B0 06     2690            BCS L5              ; ALLES OK
21F2- A9 00     2700            LDA #0
21F4- 85 BA     2710            STA *BANZ
21F6- A6 1B     2720            LDX *LAENGE
21F8- 20 24 BB  2730 L5         JSR STRDR1
                2740 ; DRUCKT STRING MIT LAENGE IN X-REGISTER
21FB- C6 BA     2750 L51        DEC *BANZ
21FD- 10 03     2760            BPL P4              ; ' ' FERTIG GEDRUCKT
21FF- 4C 9D 20  2770            JMP FORMATSCAN
2202- A9 20     2780 P4         LDA #$20            ; ' ' LADEN
2204- 20 D2 FF  2790            JSR PRINTCHAR
2207- 4C FB 21  2800            JMP L51
220A- A0 02     2810 NEXTDATA   LDY #2
220C- B1 B7     2820            LDA (DATEN),Y
220E- 85 20     2830            STA *$20
2210- 88        2840            DEY
2211- B1 B7     2850            LDA (DATEN),Y
2213- 85 1F     2860            STA *$1F
2215- 88        2870            DEY
2216- B1 B7     2880            LDA (DATEN),Y
2218- AA        2890            TAX
2219- 86 1C     2900            STX *SLAENGE
221B- A5 B7     2910            LDA *DATEN
221D- 18        2920            CLC
221E- 69 03     2930            ADC #3
2220- 85 B7     2940            STA *DATEN
2222- 90 02     2950            BCC NEXTENDE
2224- E6 B8     2960            INC *DATEN+1
2226- 60        2970 NEXTENDE   RTS
2227- A5 19     2980 ENDE       LDA *ENDFLAG
2229- D0 07     2990            BNE TEILENDE
                3000 ;JETZT ENDE: CR DRUCKEN
222B- A9 0D     3010            LDA #$0D            ; CR LADEN
222D- 20 D2 FF  3020            JSR PRINTCHAR
2230- D0 0E     3030            BNE RETURN          ;UNBED. SPRUNG
2232- A9 00     3040 TEILENDE   LDA #0
2234- A4 BE     3050            LDY *SCANPTR
2236- 20 BC C4  3060            JSR INTFAC          ; SCANPTR -> REAL-ZAHL
2239- A6 C7     3070            LDX *SCANADR
223B- A4 C8     3080            LDY *SCANADR+1
223D- 20 0A CD  3090            JSR FACSPE
```

```
                3100 ; SICHERT SCANPTR IN BASIC-VARIABLE
2240- A5 BD     3110 RETURN     LDA *IEEEAUS
2242- F0 03     3120            BEQ EXIT
2244- 20 CC FF  3130            JSR UNLISTEN          ; IEEE-BUS ABSCHALTEN
2247- 60        3140 EXIT       RTS                   ; ENDE PRINTUSING
                0100            .EN

                0010 ; ********************************
                0020 ; *                              *
                0030 ; *        BSLABELS.3            *
                0040 ; *                              *
                0050 ; *  BETRIEBSSYSTEMEINSPRUENGE   *
                0060 ; *  FUER BETRIEBSSYSTEM 3       *
                0070 ; *                              *
                0080 ; *  (40 - ZEICHEN-BILDSCHIRM)   *
                0090 ; *                              *
                0100 ; *  KOMMENTARE SIEHE BSLABELS.4*
                0110 ; *                              *
                0120 ; ********************************
                0130 TALK       .DE $F770             ; IEEE-EINGABEGERAET MIT D
                0140 UNTALK     .DE $F272             ; ALLE IEEE-EINGABEGERAETE
                0150 LISTEN     .DE $FFC9             ;AUSGABE SETZEN
                0160 UNLISTEN   .DE $FFCC             ;NORMAL - I/O
                0170 BWERT      .DE $CC9F             ; AUS BASIC-AUSDRUCK IN FA
                0180 GARCOL     .DE 49960             ; REINIGT STRINGBEREICH, W
                0190 CHGET      .DE $70               ; HOLT 1 BASIC-ZEICHEN
                0200 CHGOT      .DE $76               ; HOLT ALTES BASIC-ZEICHEN
                0210 DRUADR     .DE 56373             ; ADRESSE IN (X,A) AUS (X=
                0220 INSERT     .DE $C2D8             ; DIVERSE PARAMETER, SIEHE
                0230 CURSOR     .DE 196               ; NUR DIE CURSORPOSITION
                0240 FACSPE     .DE $DAE3             ; FAC IN SPEICHER, IN (X,Y
                0250 PRINTCHAR  .DE $FFD2             ; 1 ZEICHEN AUSGEBEN (STEH
                0260 VASUC      .DE 53101             ;HOLT VARIABLENPOINTER AUS
                0270 STRFAC     .DE $D68F             ; WANDELT STRING IN ($1F,$
                0280 MUL10      .DE 55790             ; FAC = FAC * 10 WIRD GERE
                0290 ADD0.5     .DE 55084             ; FAC = FAC + 0.5 WIRD GER
                0300 FACSTR     .DE $DCE9             ; FAC IN STRING AB $0100 W
                0310 STRDR1     .DE $CA23             ; STRING AB ($1F,$20) MIT
                0320 STRDR2     .DE $DCE6             ; STRING AB (A,Y) DRUCKEN,
                0330 INTFAC     .DE $D26D             ; WANDELT (A,Y) (A=HIGH BY
                0340 SPEFAC     .DE $DAAE             ; HOLT SPEICHERWERT AUS (A
                0350 INT        .DE $DBD8             ;SCHNEIDET NACHKOMMASTELLE
                0360 FLINT      .DE $D09A             ; WANDELT FAC IN INTEGER
                0370 ; ********************************
                0380 ; *                              *
                0390 ; *  WESENTLICHE BS-VARIABLEN    *
                0400 ; *                              *
                0410 ; *  (WERDEN VON BS-ROUTINEN     *
                0420 ; *  ALS PARAMETER BENUTZT)      *
                0430 ; *                              *
                0440 ; *  VERSION BS 3                *
                0450 ; *                              *
                0460 ; *                              *
                0470 ; ********************************
                0480 VARADR     .DE 68                ;  ERGEBNIS VON VASUC
                0490 STGRE      .DE 48                ;  UNTERGRENZE STRINGBEREI
                0500 VAGRE      .DE 46                ;  OBERE GRENZE VARIABLENL
                0510 INADR      .DE $61               ;INTEGERWERT (HIGH) NACH W
                0520 HILF       .DE 203               ; HILFSADRESSE IN ZEROPAGE
                0530 ; ********************************
                0540 ; *                              *
                0550 ; *    WICHTIGE HILFSROUTINEN    *
                0560 ; *                              *
                0570 ; *                              *
                0580 ; *    ALS INTERFACE ZUM BS      *
                0590 ; *                              *
                0600 ; *    VERSION:  BS 3            *
                0610 ; *                              *
                0620 ; *                              *
                0630 ; *                              *
                0640 ; *                              *
                0650 ; ********************************
```

```
2000- 20 70 00  0660 GETVA      JSR CHGET           ; HOLT VARIABLENPOINTER UN
2003- 20 6D CF  0670            JSR VASUC
2006- A0 00     0680            LDY #0
2008- 60        0690            RTS
                0700            ;
2009- 20 70 00  0710 WERT       JSR CHGET           ; HOLT 1 BYTE-AUSDRUCK AUS
200C- 20 9F CC  0720            JSR BWERT
200F- 20 9A D0  0730 WERTPLUS6  JSR FLINT
2012- A5 62     0740            LDA *INADR+1
2014- 60        0750            RTS
                0760            ;
2015- 85 CB     0770 VASAVE     STA *HILF           ; VAADR HOLEN UND IN ZERO-
                0780            ;DIE IN A STEHT, RETTEN
2017- 20 00 20  0790            JSR GETVA
201A- 84 CC     0800            STY *HILF+1
201C- A5 44     0810            LDA *VAADR
201E- 91 CB     0820            STA (HILF),Y
2020- C8        0830            INY
2021- A5 45     0840            LDA *VAADR+1
2023- 91 CB     0850            STA (HILF),Y
2025- 60        0860            RTS
                0870            ;
2026- 85 CB     0880 STRHOL     STA *HILF           ; STRING HOLEN (AUCH AUSDR
2028- 20 70 00  0890            JSR CHGET           ;TRENNZ. UEBERLESEN
202B- 20 9F CC  0900            JSR BWERT           ; HOLT STRINGPTR IN (97/98
202E- A5 61     0910            LDA *97
2030- 85 44     0920            STA *VAADR
2032- A5 62     0930            LDA *98
2034- 85 45     0940            STA *VAADR+1
2036- A0 00     0950            LDY #0
2038- 84 CC     0960            STY *HILF+1
203A- B1 44     0970 STRL1      LDA (VAADR),Y
203C- 91 CB     0980            STA (HILF),Y
203E- C8        0990            INY
203F- C0 03     1000            CPY #3
2041- D0 F7     1010            BNE STRL1
2043- A9 16     1020            LDA #22
2045- 8D 13 00  1030            STA 19              ; SETZT STRING-STACK ZURUE
2048- 60        1040            RTS
2049- A5 44     1050 WERTGOT    LDA *VAADR          ; HOLT ADRESSE VON REAL-ZA
204B- A4 45     1060            LDY *VAADR+1
204D- 20 AE DA  1070            JSR SPEFAC
2050- 4C 0F 20  1080            JMP WERTPLUS6       ; WEITER IN WERT
                0100            .EN
```

```
1000 REM ************************************************************
1010 REM ***
1020 REM ***  DEMOPROGRAMM ZU PRINTUSING
1030 REM ***
1040 REM ***  (C) 1984 BY [illegible]
1050 REM ***
1060 REM ************************************************************
1070 :
1080 :
1090 REM ***  ES IST VORAUSGESETZT, DASS PRINTUSING GELADEN IST, AB
1100 REM ***  $2000 LIEGT UND VOR UEBERSCHREIBEN DURCH BASIC GESCHUETZT IST.
1110 :
1120 :
1130 PU = 8275:REM ***  ANFANGSADRESSE PRINTUSING
1140 :
1150 REM ***  DATENFELD ANLEGEN
1160 DIM D$(20) : FOR I = 1 TO 19 : D$(I) = STR$(10*I + I*0.5) : NEXT
1170 :
1180 DIM S$(4) : S$(1)="HALLO" : S$(2)="EIN ZU LANGER STRING IST DAS"
1190 S$(3) = "500: ZAHL AM ANFANG" : S$(4) = "ZAHL AM ENDE: 600"
1200 :
1210 REM ***  GRUNDFORMATE FUER ZAHL UND STRING
1220 FI$="  >#######<  "
1230 FR$="  >####.##<  "
1240 FS$="  >@@@@@@@<  "
1250 :
```

```
1260 REM ***  AUSGEBEN EINES ZAHLENFELDES AUF DEN SCHIRM
1270 FF$="": FORI=1TO19 : FF$=FF$+FR$ : NEXT
1280 SYS(PU),FF$,D$(1)
1290 :
1300 REM ***  DITO, ABER HUEBSCH IN DREIERKOLONNEN
1310 FOR I = 1 TO 20 STEP 3: SYS(PU),"LINE"+FR$+FR$+FR$,D$(I) : NEXT
1320 :
1330 REM ***  DITO, ABER ALS INTEGERS
1340 FOR I = 1 TO 20 STEP 3: SYS(PU),"LINE"+FI$+FI$+FI$,D$(I) : NEXT
1350 REM ***  AUSGABE VON STRINGS
1360 FOR I = 1 TO 4 : SYS(PU),"STRING: "+FS$,S$(I) : NEXT
1370 :
1380 REM ***  STRINGS IM ZAHLENFORMAT AUSGEBEN
1390 FOR I = 1 TO 4 : SYS(PU),"ZAHL? :"+FR$,S$(I) : NEXT
1400 :
1410 REM ***  DAS GROSSE FORMAT FF$ SCHRITTWEISE ABARBEITEN
1420 FP = 0 : FOR I = 1 TO 20 : SYS(PU),FF$,D$(I),FP : NEXT
1430 :
1440 REM ***  AUSGABE AUF PERIPHERIEGERAET #4 (DRUCKER)
1450 OPEN 4,4
1460 FOR I = 1 TO 4 : SYS(PU)#4,"DRUCKER: "+FS$,S$(I) : NEXT
1470 :
1480 REM ***  GEMISCHTES FORMAT TEILWEISE ABARBEITEN
1490 F$=">>>>>BEGINN    ##.##    STRING @@@@@@@@  LETZTES ELEMENT #####"
1500 FP = 6:REM ***  UM DIE '>'-ZEICHEN NICHT AUSZUGEBEN
1510 SYS(PU)#4,F$,D$(19),FP : REM ***  PRODUZIERT STERNCHEN, DA UEBERLAUF
1520 SYS(PU)#4,F$,S$(2),FP,9 : REM ***  DRUCKT BIS ZUM FORMATENDE
1521 PRINT#4: REM ***  GIBT DIE DATEN AUS - ES FEHLTE NOCH CR
1530 CLOSE 4
1540 REM *** ALLES KLARGEWORDEN?
1550 END
READY.                                                          Ω
```

Bernhard Kokula, 6700 Ludwigshafen

UNITEX - universale Textausgabe

Texte in Programmen, z.B. zur Bedienerführung, werden üblicherweise am Programmende zusammengefaßt und durch indizierte Adressierung aufgerufen. Dazu hat dann jedes Programm seine eigene Textausgabe-Routine. Bequemer und einfacher wäre es, z.B. im EPROM eine feste Textausgabe zu haben, die man als Subroutine wie den 'NUMA' (AIM 65/PC 100) aufrufen kann. Den auszugebenden Text würde man dann direkt hinter den Aufruf des ausgebenden Programmes als 'BYT-Definition ablegen. Die Lösung hierzu zeigt H.G. Joepgen mit seinem SUPERPRINT, der wiederum aus E. H. Franke's PRINTSTRING abgeleitet ist.

Die grundlegende Idee darin ist, daß durch den Unterprogrammaufruf die Anfangsadresse des Textes auf dem Stack liegt, weil der Text ja unmittelbar hinter dem Aufruf abgelegt worden ist. Damit ist das Problem der 'Text-Table-Adresse' gelöst. Nun ist es kein Problem mehr, während der Textausgabe nach dem Text-Endzeichen zu suchen. Ein indirekter Sprung zur danach folgenden Adresse führt in das aufrufende Programm zurück. Zwei gute Lösungen des Textendes sind nun Joepgens $EA sowie die des AIM/PC 100, beim letzten auszugebenden Zeichen das Bit 7 zu '1' zu setzen, d.h. ASCII-Wert+ 80. Die Methode mit $EA ist bequemer, weil es der NOP-Befehl des Prozessors ist und weil man dieses NOP einfacher schreibt, dafür kostet sie 1 Byte mehr. im Ausgabetext. Die ausgebende Routine wird dafür kürzer. UNITEX ist universal und verarbeitet beide. Durch Weglassen der in den Kommentaren mit (1) und (2) gekennzeichneten Zeilen kann man sie für nur eine der genannten Möglichkeiten verkürzen.

Während beim 'Bit-7=1-END' das Unterprogramm INCREM zweimal aufgerufen wird, kann das beim 'NOP-END' entfallen. Das Inkrement am Ende ist nicht nötig, da 'NOP' ja als Befehl erkannt wird, und es werden 13-19 Zyklen weniger Maschinenzeit gebraucht. Joepgen inkrementiert nun, und das ist seine Idee, das 2. und 3. Byte im Befehl LDA 'TEXTADRESSE', und zwar in der Ausgaberoutine selbst. Das ist aber nicht möglich, wenn die Routine fest in ein EPROM soll. Trotz des Mangels an frei verfügbaren Zeropage-Adressen gibt es zumindest im AIM 65/PC 100 ein immer freies Paar im 'NOWLN' des Editors ($DF/E0). Während des Programmablaufes ist der Editor nicht aktiv, und beim Wiedereintritt mit 'T' wird NOWLN sowieso auf den Editoranfang gestellt (Unterprogramm TOPNO).

Literaturhinweise:

H. Erich Franke: PRTSTR - Einfache Textausgabe in Assemblerprogrammen. Sonderheft 'HOBBYCOMPUTER 2', Franzis-Verlag, 1979.

Hans Georg Joepgen: SPRINT - Dienstroutine zur Ausgabe von Text und Tafeln in '6502 simuliert neue Adressierungsart' aus Sonderheft 'Programme für Kleincomputer und Taschenrechner', FUNKSCHAU-Sonderheft 31, 1980.

Hinweise des Herausgebers: a) Das Unterprogramm INCREM kann entfallen, wenn man statt JSR INCREM programmiert: JSR $ F928. Im AIM/PC 100 wird damit das gleichwertige Programm AD1 des Editors benutzt. b) Andere Computer der Familie benutzen auch Ausgabeprogramme, die die Ausgabe beim Antreffen einer hex 00 beenden. c) Das Prinzip der Parameterübergabe im Anschluß an den Unterprogrammaufruf wurde für den 6502 erstmals als 'MRA Modify Return Address after JSR' in Heft 1 dieser Zeitschrift abgehandelt.

```
0000            ;     ******************************
0000            ;     *                            *
0000            ;     *         > UNITEX <         *
0000            ;     *                            *
0000            ;     *   UNIVERSALE TEXTAUSGABE   *
0000            ;     *     FUER AIM.65/PC 100     *
0000            ;     *                            *
0000            ;     *    V2.1  6.81/B.KOKULA     *
0000            ;     ******************************
0000            ;
0000            ;UEBERNIMMT ASCII-TEXTE DIREKT FOLGEND DEM AUFRUF
0000            ;GIBT SIE UEBER 'OUTALL' AUS UND KEHRT ZUM AUF-
0000            ;RUFENDEN PROGRAMM HINTER DEM TEXT ZURUECK:
0000            ;  WENN ES EIN ASCII+$80 FINDET ODER
0000            ;  DEM ASCII TEXT EIN 'NOP' FOLGT
0000            ;DIE ROUTINE AENDERT KEINE REGISTER AUSSER <A>
0000            ;SIE IST 'ROM-FAEHIG', VARIABEL IST NUR 'NOWLN'
0000            ;DIE ZEROPAGE ADRESSEN WERDEN NUR TEMPORAER VOM
0000            ;EDITOR BENOETIGT UND STOEREN WEDER BASIC,
0000            ;ASSEMBLER, PL/65 NOCH FORTH.
0000
0000            ;NACHFOLGENDE ZEILEN KOENNEN ENTFALLEN WENN:
0000            ;0) (Y) NICHT BEWAHRT WERDEN SOLL
0000            ;1) ALLE TEXTE MIT 'ASCII+$80' ENDEN
0000            ;2) ALLEN TEXTEN EIN 'NOP' FOLGT
0000            ;3) BEI 'NOP-END' KANN 'INCREM' DIREKT IN DIE
0000            ;ROUTINE EINGEBAUT WERDEN, DA NUR 1X BENOETIGT
0000            ;    'UNITEX' IST AUSSERDEM FREI IM GANZEN
0000            ;    ADRESSBEREICH VERSCHIEBBAR OHNE UMRECHNUNG
0000
0000            ;  **** 'START POINTER'
0000
0000 TEXT           =$300           ;AUFRUFADRESSE
0000 TEST           =$400           ;START TESTPROGRAMM
```

```
0000                    *=$10C          ;USER-F1
010C            4C0004  JMP TEST
010F            ---------------------
010F
010F                    *=$DE
00DE SAVEY              *=*+1           ;0) SAVE (Y)
00DF NOWLN              *=*+2           ;INDIR. INDIZ. POINTER
00E1
00E1            ;  **** 'UNIVERSAL TEXT ROUTINE'
00E1
00E1                    *=TEXT
0300            84DE    STY SAVEY       ;0) SAVE Y
0302            A000    LDY #0
0304            68      PLA             ;ADRESSE DES AUFRUFERS
0305            85DF    STA NOWLN
0307            68      PLA             ;DITO HI-BYTE
0308            85E0    STA NOWLN+1
030A UN1        202403  JSR INCREM      ;ADRESS POINTER +1
030D            B1DF    LDA (NOWLN),Y   ;HOLE ASCII DATE
030F            C9EA    CMP #$EA        ;1) OP-CODE 'NOP'
0311            F00C    BEQ BACK        ;1) FERTIG
0313            48      PHA             ;2) NOWLN DATE
0314            297F    AND #%01111111    ;2) KILL BIT 7
0316            20BCE9  JSR OUTALL      ;GIB ASCII AUS
0319            68      PLA             ;2) HOLE DATE WIEDER
031A            10EE    BPL UN1         ;BIT 7 NICHT '1'
031C            202403  JSR INCREM      ;2) BEREITE SPRUNGADRESSE
031F BACK                               ;ZURUECK ZUM RUFENDEN PROGRAMM
031F            A4DE    LDY SAVEY       ;0) HOLE (Y) ZURUECK
0321            6CDF00  JMP (NOWLN)     ;UND SPRING
0324            ---------------------
0324
0324 INCREM
0324            ;3) SUBROUTINE ZUR 16-BIT ADRESSERHOEHUNG
0324            E6DF    INC NOWLN       ;LO-BYTE +1
0326            D002    BNE NOPAGE
0328            E6E0    INC NOWLN+1     ;HI-BYTE +1
032A NOPAGE     60      RTS             ;3)
032B
032B            ---------------------
032B            TEST PROGRAMM
032B
032B                    *=TEST
0400            200003  JSR TEXT        ;TEXT ROUTINE (NOP-END)
0403            0D      .BYT $0D,'1. TEXT',$0D   ;VOR UND DANACH 'CRLF'
0404            312E
040B            0D
040C            EA      NOP
040D            200003  JSR TEXT +(BIT-7-END)
0410            322E    .BYT '2. TEXT MIT BIT-7-EN',$C4
0424            C4
0425            60      RTS             ;PROGRAMMENDE
0426
0426            ---------------------
0426            AIM EQUATES
0426
0426 OUTALL             =$E9BC
0426                    .END
```

Der Probelauf des Programmes TEST ergibt:

```
1. TEXT
2. TEXT MIT BIT-7-END
```

Das Programm UNITEX mit NOP-End in seiner Minimalform:

```
0000 TEXT           =$300           ;AUFRUFADRESSE
0000                ----------------------
0000          UNITEX
0000
0000                *=$DE
00DE NOWLN          *=*+2           ;INDIR. INDIZ. POINTER
00E0 OUTALL         =$E9BC
00E0
00E0                *=TEXT
0300          A000  LDY #0
0302          68    PLA             ;ADRESSE DES AUFRUFERS
0303          85DE  STA NOWLN
0305          68    PLA             ;DITO HI-BYTE
0306          85DF  STA NOWLN+1
0308 UN1
0308          E6DE  INC NOWLN       ;ERHOEHE UM 1 (WIE 'RTS')
030A          D002  BNE NOPAGE
030C          E6DF  INC NOWLN+1     ;FALLS NOETIG HI-BYTE
030E NOPAGE
030E          B1DE  LDA (NOWLN),Y   ;HOLE ASCII DATE
0310          C9EA  CMP #$EA        ;OP-CODE 'NOP'
0312          F005  BEQ BACK        ;JA DANN FERTIG
0314          20BCE9 JSR OUTALL     ;GIB ASCII AUS
0317          10EF  BPL UN1         ;IMMER
0319 BACK                           ;ZURUECK ZUM RUFENDEN PROGRAMM
0319          6CDE00 JMP (NOWLN)    ;NUN SPRING AUF 'NOP'
031C                .END
031C           ERRORS= 000
```

Ω

Horst Brettin, 1000 Berlin 44

ROM-Test für CBM 3001

Das Testprogramm ist für alle diejenigen gedacht, die ihrem CBM nicht trauen (es könnten ja einige Bit verbogen sein) oder die ihn in Verdacht haben, nach längerer Betriebszeit zu spinnen (Wärmefehler im ROM). Für den Fall, daß der Rechner schon beim Einschalten abstürzt, nützt dieses Programm gar nichts (man muß es ja laden und starten können).

Das Programm besteht aus einer einzigen BASIC-Zeile: 10 SYS 1037 und dem Maschinenprogramm von hex 040D bis 04D5. Beide Teile passen genau hintereinander, so daß sie als ein Programm geladen und mit RUN gestartet werden können.

Für jedes ROM wird ein Prüfpolynom berechnet (das ist sicherer als eine einfache Prüfsumme) und mit einer Tabelle verglichen (CKTABL und CKTABH). Adresse (oberes Halbbyte) und Länge (unteres Halbbyte) aller zu prüfenden ROMs sind in einer weiteren Tabelle (ROMTAB) untergebracht, die Null kennzeichnet das Ende.

Für die Berechnung eigener ROMs ist das NOP bei 0450 durch ein BRK zu ersetzen, im Monitor sind dann die Prüfbits aus $11 und $12 zu lesen (in BASIC geht das nicht, daher das BRK). - Die

Prüfpolynom-Routine ist aus der Pseudo-Zufallszahlen-Routine aus Heft 8 dieser Zeitschrift, Seite 35 entwickelt worden. Man beachte auch, das zum Herausschieben eines Bits kein Zähler benutzt wird, sondern einfach geprüft wird, ob der Akku leer ist. Das ginge auch beim Schieben einer RAM-Zelle. Damit dies mit Sicherheit erst nach 8 Bit der Fall ist, wird eine Endmarke gesetzt, sie befindet sich zum Schluß im Carry.

Das Programm kann durch Drücken der STOP-Taste beendet werden. Die Reaktion kann unter Umständen ein paar Sekunden auf sich warten lassen, da die Stop-Taste nur nach vollständiger Prüfung eines ROMs abgefragt wird. Sonst läuft es endlos, so daß auch Fehler, die erst nach Stunden auftreten, gefunden werden können. Als Hinweis, daß das Programm noch läuft, wird die Anzeige 'ROMTEST LAEUFT' nach jedem geprüften ROM invertiert. Solange kein Fehler auftritt, erscheint der Text 'ROM OK'. Tritt ein Fehler auf, so verschwindet das 'OK' und die Adresse des ROMs wird angezeigt. Dazu wird keine Ausgaberoutine des Betriebssystems verwendet, damit diese Ausgabe auf jeden Fall erfolgt. Diese Anzeige bleibt auch dann stehen, wenn das ROM bei allen weiteren Prüfungen in Ordnung ist!

```
                *       ---- ROMTEST ---

                CEKL    EQU $11          (C) 1981 BY
                CEKH    EQU $12
                VEKL    EQU $21         HORST BRETTIN
                VEKH    EQU $22
                XR      EQU $AD
                TV2     EQU $8050
                TV4     EQU $80A0
                STRING  EQU 51740
                PIAK    EQU $E812


0400  A99A    START   LDA#TEXT          ;L-BYTE
0402  A004            LDY#TEXT/256      ;H-BYTE
0404  201CCA          JSR STRING        ;TEXT AUSGEBEN
0407  78              SEI
0408  A000            LDY#0
040A  8421            STY VEKL
040C  A2FF    NEW     LDX#$FF
040E  E8      NEXT    INX
040F  86AD            STX XR
0411  BDC604          LDA ROMTAB,X
0414  F0F6            BEQ NEW           ;TABELLENENDE ?
0416  AA              TAX               ;ACCU RETTEN
0417  29F0            AND#$F0           ;MASKE F. ADRESSE
0419  8522            STA VEKH
041B  8A              TXA               ;ACCU WIEDERHERSTELLEN
041C  290F            AND#$0F           ;MASKE F. PAGE-ANZAHL
041E  AA              TAX
041F  206F04          JSR INV
0422  8411            STY CEKL          ;PRUEFSUMME
0424  8412            STY CEKH          ;LOESCHEN
0426  B121    LOOP    LDA (VEKL),Y
0428  208704          JSR CEKBYT
042B  C8              INY
042C  D0F8            BNE LOOP
042E  E622            INC VEKH
0430  CA              DEX
0431  10F3            BPL LOOP
```

```
0440  A6AD          LDX XR              ;XR WIEDERHERSTELLEN
0442  A511          LDA CEKL
0444  DDCC04        CMP CKTABL,X
0447  D007          BNE ERROR
0449  A512          LDA CEKH
044B  DDD104        CMP CKTABH,X
044E  F0CB          BEQ NEXT

0450  EA      ERROR NOP                 ;HIER BRK EINSETZEN
0451  BDC604        LDA ROMTAB,X        ;ROMADR. HOLEN
0454  4A            LSR                 ;NACH
0455  4A            LSR                 ;UNTEN
0456  4A            LSR                 ;SCHIEBEN
0457  4A            LSR
0458  C90A          CMP#$0A             ;IN BILDCODE
045A  09B0          ORA#$B0             ;WANDELN
045C  9002          BCC ZIFF
045E  E939          SBC#$39
0460  9DA680  ZIFF  STA TV4+6,X

0463  A920          LDA#' '             ;'OK' UEBERSCHREIBEN
0465  8DA480        STA TV4+4
0468  A93A          LDA#':
046A  8DA580        STA TV4+5
046D  D0AC          BNE NEXT            ;IMMER !

046F  A00E    INV   LDY#14
0471  B94F80  INVLOOP LDA TV2-1,Y
0474  4980          EOR#$80
0476  994F80        STA TV2-1,Y
0479  88            DEY
047A  D0F5          BNE INVLOOP
047C  AD12E8        LDA PIAK            ;STOP-TASTE ?
047F  2910          AND#$10
0481  D003          BNE NOSTOP
0483  68            PLA                 ;ADRESSE VOM STAPEL
0484  68            PLA
0485  58            CLI
0486  60      NOSTOP RTS

              ************************************
              *        PRUEFPOLYNOM              *
              ************************************

0487  38      CEKBYT SEC                 ;ENDEMARKE
0488  2A             ROL                 ;1.BIT IN C
0489  48      CKLOOP PHA                 ;AC RETTEN
048A  A511           LDA CEKL
048C  2A             ROL
048D  2612           ROL CEKH
048F  B002           BCS CS
0491  492D           EOR#$2D
0493  8511    CS     STA CEKL
0495  68             PLA                ;AC WIEDERHERSTELLEN
0496  0A             ASL                ;AC LEER ?
0497  D0F0           BNE CKLOOP         ;NEIN => NOCHMAL
0499  60             RTS
```

```
049A  934752
049D  414649
04A0  4B2D43
04A3  424D20
04A6  524F4D
04A9  544553
04AC  54      TEXT    ASC "GRAFIK-CBM ROMTEST"
04AD  0D0D            DFB CR,CR
04AF  524F4D
04B2  544553
04B5  54204C
04B8  414555
04BB  4654            ASC "ROMTEST LAEUFT"
04BD  0D0D            DFB CR,CR
04BF  524F4D
04C2  204F4B          ASC "ROM OK"
04C5  00              DFB 0
04C6  B7CFDF
04C9  E7FF00  ROMTAB  DFB $B7,$CF,$DF,$E7,$FF,0
04CC  7FDD77

04CF  E720    CKTABL  DFB $7F,$DD,$77,$E7,$20
04D1  969379
04D4  C105    CKTABH  DFB $96,$93,$79,$C1,$05
```

Ω

Uhr und Kalender

Bei zahlreichen Prozessen möchte man die Uhrzeit und das Datum loggen, zu dem ein Ereignis eingetreten ist. Man wird dann im allgemeinen dazu einen Kontrollstreifen mit den Daten bedrucken. Man denke nicht nur an Meß- und Steueranordnungen, sondern auch an Zugangskontrollen für Tore und besondere Räume. Zwar gibt es LSI-Bausteine mit einer Echtzeituhr. Man wird sie wählen, wenn das System nach einem Stromausfall wieder selbstanlaufend ist, in dem der power on reset also in das Anwenderprogramm hineinführt, ohne daß ein Bediener eingreifen muß. In vielen anderen Fällen reicht eine Software-Uhr mit Kalender, die man mit Hilfe eines Timers im Interrupt betreibt. Dazu das folgende Programm als eine mögliche Lösung. Benutzt wird eine VIA, hier die Anwender-VIA 6522 im AIM 65 in den Adressen ab hex A000. Die weiteren Referenzen zum Betriebsprogramm sind gering. OUTDIS besorgt die Ausgabe auf das rote LED-Display, CURPO2 ist der Zeiger in die nächste zu beschreibende Display-Zelle, GETKEY holt ein Zeichen von der Tastatur, NUMA gibt 1 Hexbyte als 2 ASCII aus und PACK verdichtet 2 ASCII-Zahlen zu einem Hexbyte. Diese Dinge sind auch für einen anderen Rechner schnell umcodiert.

Das Programm benutzt Zähler für die Jiffies (1/20 Sekunde), für die Sekunden, Minuten, Stunden, für das Tagesdatum und für den Monat in der Zeropage. Diese Zähler können auch an passendere Stelle ins RAM gelegt werden. Es ist dafür gesorgt, daß Uhr und Kalender richtig fortschalten, ausser an einem 29, Februar.

Der Programmbeginn ab INIT0 ist ein Muß. Es werden der IRQ-Vektor eingesetzt und der Timer T1 gestartet. Der interaktive Teil mit den Routinen ab EINGAB kann entfallen, wenn man Uhr und Kalender von Hand in den Speicherzellen initialisiert. Das Interruptprogramm ist wieder ein Muß, es schaltet die Zeiteinheiten fort. Den Aufruf JSR OUTDIS und JSR EIN5 nach dem Label UPD1 ziemlich am Schluß wird man sich in allgemeinen wohl ersparen, er dient nur zur Demonstration, daß das Programm läuft. Im praktischen Einsatz wird man wohl die Zeit den Zählerstellen entnehmen und per OUTPRI auf den Drucker schreiben, zusammen mit den anderen benötigten Meldungen.

Die Quarze der Rechner haben Streuungen, ihre Ganggenauigkeit ist auch von den Umgebungseinflüssen abhängig. Man kann daher an den bezeichneten Stellen einen anderen Wert als GRANU benutzen, um eine Feinkorrektur der Uhr vorzunehmen, z.B. bei Label UPDATH oder bei UPDATD. Die benötigten Werte hängen auch davon ab, was man in der Interruptroutine evtl. zusätzlich ablaufen läßt.

```
0000            ;--------------------
0000            ;KALENDER-UHR
0000                                    ;V03.10.81-1
0000            ;--------------------
0000            ;SYMBOLERKLAERUNG
0000            ;ADRESSEN DES &ONITOR-PROGRAMMES
0000            ;VERKEHRSZEICHEN & KONSTANTE
0000 NUMA               =$EA46          ;AUSGABE 1 HEX = 2 ASCCI
0000 IFR                =$A00D          ;INTERRUPT FLAG REGISTER
0000 T1EN               =$40            ;OPERATOR FUER ACR
0000 INTEN              =$80            ;OPERATOR FUER IFR
0000 GRANU              =$EC            ;ZAHL JIFFIES/SEKUNDE
0000 T1LL               =$A006          ;TIMER LATCH LOW
0000 T1CH               =$A005          ;COUNTER HIGH, TIMER 1
0000 ACR                =$A00B          ;STEUERREGISTER
0000 IER                =$A00E          ;INTERRUPT ENABLE REGISTER
0000 IRQV4              =$A400          ;IRQ-VEKTOR IM RAM
0000 PSLS               =$E7DC          ;LESEN MIT RUBOUT
0000 OUTDIS             =$EF05          ;AUSGABE DISPLAY
0000 PORTB              =$A800          ;VIA
0000 OUTFLG             =$A413          ;MONITOR-RAM
0000 CURPO2             =$A415          ;ZEIGER FUER DISPLAY-BUFFER
0000 CR                 =$0D            ;CARRIAGE RETURN
0000 GETKEY             =$EC40          ;HOLE 1 ZEICHEN VON TASTATUR
0000 PHXY               =$EB9E          ;RETTE X, Y
0000 PLXY               =$EBAC          ;RESTORE X, Y
0000 STIY               =$A429          ;MONRAM
0000 PACK=$EA84
0000                                    ;2 ASCII = 1 HEXBYTE
0000            ;--------------------
0000            ;ZEROPAGE-BELEGUNG
0000            ;EINRICHTUNG VON ZAEHLERN
0000                    *=0
0000 SUBSEC             *=*+1
0001 SEK                *=*+1
0002 MIN                *=*+1
0003 HOURS              *=*+1
0004 DATE               *=*+1
0005 MONTH              *=*+1
0006            ;--------------------
0006
0006                    *=$10C          ;BELEGUNG DER F1-TASTE
010C            4C0070 JMP INITO
010F            ;--------------------
010F            EINTRITT IN DAS VORDERGRUNDPROGRAMM
010F
010F            ;MIT INITIALISIERUNG
010F                    *=$7000         ;START VORDERGRUNDPROGRAMM
7000 INITO
7000            A9E1    LDA #<EMPF      ;IRQ-VEKTOR SETZEN
7002            8D00A4 STA IRQV4
7005            A970    LDA #>EMPF
7007            8D01A4 STA IRQV4+1
```

```
700A           202970 JSR EINGAB      ;DATUM UND UHRZEIT ABFRAGEN
700D
700D           A940   LDA #T1EN       ;TIMER 1 FREE RUNNING EINSTELLEN
700F           8D0BA0 STA ACR
7012           A94E   LDA #$4E        ;HIER GGFS. FEINEINSTELLUNG
7014           8D06A0 STA T1LL        ;INS TIMER LATCH LOW
7017           A9C0   LDA #T1EN+INTEN
7019           8D0EA0 STA IER         ;ENABLE INTERRUPT VON TIMER 1
701C           A9C3   LDA #$C3        ;START TIMER WITH 50 MS-CLOCK
701E           8D05A0 STA T1CH        ;IN DEN COUNTER HIGH
7021           A9EC   LDA #GRANU      ;VORGABE DER JIFFIES
7023           8500   STA SUBSEC      ;20 JIFFIES = 1 SEKUNDE
7025           58     CLI             ;ENABLE INTERRUPT DURCH DEN STATU
7026           4C56E9 JMP $E956       ;RUECKSPRUNG MONITOR
7029           ----------------------
7029           VORGABE FUER CLOCK     ;INTERAKTIVE ABFRAGE
7029 EINGAB
7029           A203   LDX #3
702B           A000   LDY #0
702D EIN1
702D           A90D   LDA #CR         ;DISPLAY LEEREN
702F           2005EF JSR OUTDIS
7032 EIN2
7032           B9A970 LDA M1,Y        ;PROMPTE MONAT, TAG USW.
7035           2005EF JSR OUTDIS      ;TEXT AUSGEBEN
7038           C8     INY
7039           0A     ASL A
703A           90F6   BCC EIN2        ;NOCH KEIN ENDE
703C           209EEB JSR PHXY
703F           208570 JSR EIN3        ;EINGABE VOM BEDIENER HOLEN
7042           208570 JSR EIN3
7045           0D29A4 ORA STIY
7048           20ACEB JSR PLXY
704B           9502   STA MIN,X
704D           A920   LDA #'
704F           2005EF JSR OUTDIS
7052           CA     DEX
7053           10DD   BPL EIN2
7055           A90D   LDA #CR
7057           2005EF JSR OUTDIS
705A           A200   LDX #0          ;SEKUNDEN MIT 0 VORLADEN
705C           8601   STX SEK
705E           A203   LDX #MONTH-MIN
7060 EIN41
7060           B502   LDA MIN,X       ;EINGABEN FILTRIEREN
7062           DDA570 CMP LIMIT,X     ;GEGEN HOECHSTWERTE
7065           B0C2   BCS EINGAB      ;GGFS. WIEDERHOLEN
7067           CA     DEX
7068           10F6   BPL EIN41
706A EIN4
706A           B9A970 LDA M1,Y
706D           2005EF JSR OUTDIS      ;OKAY ABFORDERN
7070           C8     INY
7071           0A     ASL A
7072           90F6   BCC EIN4
7074           A204   LDX #4
7076           209770 JSR EIN5
```

```
7079 EIN6                              ;WIRD ZEITVORGABE GENEHMIGT?
7079          2040EC JSR GETKEY
707C          C94E   CMP #'N ;NEIN?
707E          F0A9   BEQ EINGAB
7080          C94A   CMP #'J
7082          D0F5   BNE EIN6          ;WENN NICHT JA
7084          60     RTS               ;VORGABE GENEHMIGT
7085          ---------------------
7085
7085 EIN3                              ;SUBROUTINE
7085          2040EC JSR GETKEY        ;HOLE 1 ZEICHEN
7088          C930   CMP #'0 ;NUMERISCHES FILTER
708A          90F9   BCC EIN3          ;STEUERZEICHEN
708C          C93A   CMP #': ;>9
708E          B0F5   BCS EIN3
7090          2005EF JSR OUTDIS
7093          2084EA JSR PACK
7096          60     RTS
7097 EIN5
7097          B501   LDA SEK,X         ;VORGABEN NOCHMAL ANZEIGEN
7099          2046EA JSR NUMA
709C          A92D   LDA #'-
709E          2005EF JSR OUTDIS
70A1          CA     DEX
70A2          10F3   BPL EIN5
70A4          60     RTS
70A5
70A5          ----------------------
70A5          KONSTANTENBEREICH **********************
70A5          ----------------------
              CLOCKLIMITS              ;FUER FILTERUNG DER ZEITVORGABEN
70A5 LIMIT
70A5          60     .BYT $60,$24,$32,$13   ;MINUTEN, STUNDEN USW.
70A6          24
70A7          32
70A8          13
70A9          ---------------------
70A9          MESSAGES
70A9 M1
70A9          4D4F   .BYT 'MONAT=?',$A0
70B0          A0
70B1 M2
70B1          4441   .BYT 'DATUM=?',$A0
70B8          A0
70B9 M3
70B9          5354   .BYT 'STUNDE=?',$A0
70C1          A0
70C2          4D49   .BYT 'MINUTE=?',$A0
70CA          A0
70CB M5
70CB          5354   .BYT 'START OK?',$A0
70D4          A0
70D5          ---------------------
70D5          MONATSLAENGE ;IN ASCII-WERTEN GETARNT
70D5 MDAY
70D5          3229   .BYT '2)2121221212'
70E1
```

```
                ---------------------
70E1            INTERRUPTPROGRAMM
70E1 EMPF
70E1            48      PHA             ;SAVE AKKU
70E2            8A      TXA             ;SAVE X
70E3            48      PHA SAVE
70E4            AD04A0  LDA T1CH-1      ;RESET IFR
70E7            E600    INC SUBSEC      ;ZAEHLE JIFFIES HOCH BIS 00
70E9            D061    BNE IOUTT       ;KEINE VOLLE SEKUNDE
70EB            F8      SED             ;DEZIMALMODUS
70EC            18      CLC
70ED            A901    LDA #1
70EF            6501    ADC SEK         ;SEKUNDEN +1
70F1            8501    STA SEK
70F3            C960    CMP #$60        ;MINUTE VOLL?
70F5            905A    BCC UPDATM      ;NEIN
70F7            A900    LDA #0          ;60 SEK = 0 SEK
70F9            8501    STA SEK
70FB            A900    LDA #0
70FD            6502    ADC MIN         ;+1 AUS CARRY ZU MINUTEN
70FF            8502    STA MIN
7101            C960    CMP #$60        ;STUNDE VOLL?
7103            9050    BCC UPDATH      ;NEIN
7105            A900    LDA #0          ;60 MIN. = 0 MIN.
7107            8502    STA MIN
7109            6503    ADC HOURS       ;+1 AUS CARRY ZU STUNDEN
710B            8503    STA HOURS
710D            C924    CMP #$24        ;MITTERNACHT = DATUMSWECHSEL
710F            902D    BCC UPDATD      ;NEIN
7111            A900    LDA #0          ;STUNDEN WIEDER AB 0
7113            8503    STA HOURS
7115            6504    ADC DATE        ;+1 AUS CARRY ZUM DATUM
7117            8504    STA DATE
7119            A505    LDA MONTH
711B            C910    CMP #$10        ;BCD-CODE MUSS GGFS. BINAER WERDE
711D            9006    BCC CKDATE      ;<10, KEINE UMCODIERUNG
711F            290F    AND #$0F        ;MASKIERE HALBBYTE
7121            D8      CLD             ;BINAER
7122            6909    ADC #$9         ;+9 + 1 VOM CARRY =10
7124            F8      SED             ;DEZIMAL
7125 CKDATE     AA      TAX             ;ADRESSIERUNG DER MONATSLAENGE
7126            A504    LDA DATE        ;WELCHES DATUM IST ERREICHT?
7128            DDD470  CMP MDAY-1,X    ;VERGL. MONATSLAENGE
712B            9011    BCC UPDATD      ;KEIN MONATSENDE
712D            A901    LDA #1          ;DATUM BEGINNT WIEDER BEI 1
712F            8504    STA DATE
7131            18      CLC
7132            6505    ADC MONTH       ;+1 ZUM MONAT
7134            8505    STA MONTH
7136            C913    CMP #$13        ;JAHRESWECHSEL?
7138            9004    BCC UPDATD      ;NEIN
713A            A901    LDA #1          ;WIEDERBEGINN MONAT 1
713C            8505    STA MONTH
713E            ---------------------
713E
713E UPDATD                             ;DIE FOLGENDEN DIREKTOPERANDEN
713E            A9EC    LDA #GRANU      ;ZUR FEINKORREKTUR DER UHR
7140 UPD1
7140            8500    STA SUBSEC      ;VORGABE DER JIFFIES
```

```
7142            A90D    LDA #CR         ;ZU TESTZWECKEN: ZWISCHENANZEIGE
7144            2005EF  JSR OUTDIS      ;LEERE DISPLAY
7147            A204    LDX #4
7149            209770  JSR EIN5        ;ANZEIGE MON. TAG. STD. MIN. SEK.
714C            --------------------
714C
714C IOUTT                              ;DER INTERRUPT WIRD VERLASSEN
714C            68      PLA             ;WIEDERHERSTELLUNG VON A UND X
714D            AA      TAX
714E            D8      CLD
714F            68      PLA
7150            40      RTI             ;KEHRE ZUM VORDERGRUNDPROGR. ZUR.
7151 UPDATM
7151            A9EC    LDA #GRANU+0    ;GGFS. FEINEINSTELLUNG DER JIFFIES
7153            D0EB    BNE UPD1
7155 UPDATH
7155            A9EC    LDA #GRANU+0    ;FEINEINSTELLUNG NOETIG?
0@571           D0E7    BNE UPD1        ;BRANCH ALWAYS   ;END OF INTERRUPT
7159                    .END
7159             ERRORS= 000
```

R.L.

Software-Besprechung

Das EXBASIC II ist eine leistungsfähige Expansion für alle Commodore-Rechner mit BASIC 3 oder BASIC 4. Es umfaßt zahlreiche Hilfsfunktionen für das Editieren und Erproben von BASIC-Programmen, eine große Anzahl zusätzlicher BASIC-Befehle und mathematischer Funktionen sowie besondere Grafik- und Bildschirmbefehle. Geliefert wird es in 2 EPROMs (zus. 8KB) für die Adreßbereiche hex 9000-AFFF zusammen mit einem 96-seiten Handbuch, das für die Funktionen und Befehle die notwendigen erklärenden Beispiele enthält. Preis DM 392,- (unverb.). Bezug im Fachhandel oder von INTERFACE AGE, Dahlienstr. 4, 8011 Vaterstetten. Autoren der Firmware sind Andreas Dripke, Wiesbaden, und Michael Krause.

Es fällt etwas schwer, aus der Menge der Eigenschaften die wichtigsten voranzustellen. Sie seien so gekennzeichnet: Es gibt den INSTRING-Befehl für die Feststellung, ob ein Suchstring in einer Variablen enthalten ist, wir haben das PRINT USING für die formatierte Zahlenausgabe, ferner die Befehle FRACTION (Dezimalbruch) und ROUND (auf Stellenzahl). EXBASIC II läßt strukturierte Statements mit IF .. THEN .. ELSE zu, erlaubt das Umsetzen des DATA-Pointers mit RESTORE (Zeilennummer) und ON .. RESTORE. Eintretende Fehler können mit ON ERROR GO TO abgefangen werden. Mit RESUME ist danach eine Weiterverarbeitung möglich.

Für rekursive Programme ist eine Vervielfachung der Unterprogrammebenen möglich, mit DISPOSE werden FOR-NEXT-Schleifen verlassen. INPUTLINE und INPUTFORM erweitern das sonst bei Sonderzeichen immer abbruchverdächtige INPUT-Statement auf bis zu 79 Zeichen. Viele oft nützliche weitere Funktionen stehen zur Verfügung, das Swappen von zwei Variablen, Bestimmung des Speicherplatzes einer Variablen, Listen der augenblicklichen Variableninhalte, speziell auch für Arrays, mathematische Funktionen wie MAX, MIN, ODD, HEX und DEC zur Umwandlung.

Die Editierung von Programmtexten wird durch FIND, AUTO, RENUMBER, DELETE, FAST usw. unterstützt. Files können durch MERGE zusammengebunden werden. Kassettenoperationen laufen mit 5-facher Geschwindigkeit. Der TRACE erlaubt Probeläufe, langsan, schnell, mit Stopp.

Die grafischen Fähigkeiten betreffen das gezielte Setzen und Löschen von Bildpunkten, horizontalen und vertikalen PLOT, die Definition von Bildschirmfenstern mit Textrollen nur in diesen. Das Interface zur Maschine hat doppeltes PEEK und POKE, CALL mit mehreren Parametern und DEF USR (Adreßvektor). Der EXMON für die Serie 8001 eröffnet mnemonische Befehlseingabe und Disassemblierung, auch auf den Printer.

Das EXBASIC II in 8K Festwertspeicher wertet das von Haus aus schon sehr leistungsfähige BASIC der Commodore-Rechner weiter auf, und zwar mit Befehlen, deren bisheriges Fehlen schon oft als schmerzlich empfunden wurde, wenn man sich um eine zügige und konzentrierte Programmierung bemühte. Es kann mit sog. Softmodulen erweitert werden, so daß im Laufe der Zeit wohl nur noch wenige Wünsche offenbleiben. EXBASIC II dürfte sich schnell als ein beliebtes und geeignetes Werkzeug für den BASIC-Programmierer erweisen.

SM-Kit/M für maschinensprachliche Programmierung. In Heft 20 wurde bereits das SM-Kit/B für den BASIC-Programmierer vorgestellt. Das SM-Kit/M arbeitet mit diesem zusammen und wird als 4K EPROM mit 60-seitigem Handbuch geliefert (ca. DM 200,-). Man kann es in erster Linie als eine Analyse- und Lernhilfe für CBM-Betreiber ansprechen. Sicherlich, es enthält auch einen Mini-Assembler, seine Stärken liegen aber auf der Disassembler- und Trace-Seite. Man kann sowohl für den Assembler, wie auch für den Disassembler verschiedene Ein- und Ausgabearten schalten: Mnemonische Ein- und Ausgabe, E/A von Bytes, Adreßwörtern und Strings, Zahlen in Hex und in Dezimal. Durch 1-2 Tastendrücke schaltet man zwischen den Funktionen um. Der Disassembler schaltet auf Byte-Ausgabe, wenn er auf einen unzulässigen Code trifft. Interessant sind die FIND-Befehle: Bereiche lassen sich gezielt nach Maschinencodes und Operanden absuchen, ferner nach Befehlsgruppen, wie ladende/vergleichende oder speichernde/ändernde Befehle, und zwar in jeweils drei Adressierungsarten: absolut, indiziert und indirekt indiziert.

Speicherblöcke können mit dem .t-Befehl verschoben werden. Die Befehle .i und .d erlauben die Einfügung bzw. das Löschen von Befehlen in erklärten Programmbereichen, wobei der Programmbereich nachgeführt und solche Adressen umgerechnet werden, die für den Bereich speicherplatzabhängig sind. - Maschinenprogramme werden durch den Trace einzelschrittweise abgearbeitet, und zwar unter Anzeige der jeweiligen Registerinhalte, des Status in einem übersichtlichen Anzeigefeld, der laufenden Adresse und des disassemblierten Befehles. Es ist bei einem solchen Trace auch die Beobachtung besonderer interessierender Speicherbereiche möglich, was wird z.B. dort abgelegt, welche Veränderungen treten an einer Adresse im Interface ein? Auch lassen sich Breakpoints in Programmen setzen, ab denen der Trace beginnt.
Die lettzgenannten Fähigkeiten werden nicht nur beim Testen eigener Programme durch den Anfänger oder den Profi von Interesse sein, sondern besonders auch bei der Analyse fremder Programme, aus denen man ja oft sehr viel lernen kann. Für letztere Zwecke dürfte die bereits erwähnte gezielte Absuche nach ladenden und speichernden Befehlen und ihrer Adressierungsarten die größte Bereicherung gegenüber früheren Hilfsmitteln sein. - Anfragen an die SM Softwareverbund Microcomputer GmbH, Scherbaumstr. 29, 8 München 83.

Bedarfsgerecht selektierbare BASIC-Erweiterungen, die mit dem BASIC-Programm zusammen geladen werden, sind die sog. REM-ROU-Routinen, die vom Softwareverbund Microcomputer entwikkelt wurden. Hier ist ein flexibler Weg für die Programmentwicklung und -verbreitung aufgezeigt. Die Übertragbarkeit von Programmen für CBM-Computer ist nicht mehr davon abhängig, daß der Zielrechner ein Expansions-ROM hat. Die REM-Routinen machen davon Gebrauch, daß BASIC die Interpretation einer Befehlszeile abbricht, sobald ein REMARK angetroffen wird. Im Raum bis hin zum nächsten Statement, der normalerweise für Bemerkungen benutzt wird, verbirgt REM-ROU schnell ausführende Maschinenprogramme, die nach ihrer Aktivierung mit SYS 1040 die Zusatzbefehle wie jede andere BASIC-Expansion zur Verfügung stellen. Für REM-Rou sind bisher zwei Programmpakete mit umfangreicher Dokumentation erschienen. Dazu gehören die Routinen selbst auf Diskette, zusammen mit einem Merge-Programm und Anwendungsbeipielen, je Paket ca. DM 320,-. Das erste enthält die Befehle INSTRING (suchen), MIDSTRING (Überschreiben innerhalb eines Strings), GETSTRING (bis 255 beliebige Zeichen einlesen), GK (Groß-/Kleinumwandlung von Buchstaben). Das zweite Paket enthält CLR (Entfernen von Arrays aus der Variablentabelle mit der Möglichkeit der Neudimensionierung, FRE (garbage collect nur bei Bedarf, schneller als FR (0)), COPY (Stringfelder vertauschen, duplizieren, tranporteren, löschen), BIT Abfrage und Veränderung eines beliebigen Bits in einem String), PAC (komprimierte Ausgabe von Gleitkommazahlen nebst entpacken). - Anfragen an SM Softwareverbund Microcomputer GmbH, Scherbaumstr. 29, 8 München 83.

R.L.

Dr. A. Schnell, 5100 Aachen

Einfache Sprachausgabe mit Kleinrechner

Die Firma VOTRAX bietet einen CMOS-Baustein an, der Sprache durch Kombination einzelner Phoneme erzeugen kann. Da natürliche Sprache aus einer begrenzten Anzahl solcher Phoneme (Lautelemente) besteht, kann mit dieser Methode ein nahezu unbegrenztes Vokabular sogar in verschiedenen Sprachen erzeugt werden. Im Vergleich zu anderen spracherzeugenden Systemen ist die notwendige Datenrate sehr niedrig, da lediglich der 8-Bit-Code für das nächste zu sprechende Phonem in richtigen Moment auf die Dateneingangsleitungen des Chips gelegt werden muß. Dies hat im Mittel alle 100 ms zu erfolgen. Dadurch ist das Interfaceproblem des SC-01-Chips an einem Mikrocomputer recht einfach zu lösen, ledigleich ein 8 Bit Outputport mit Handshake-Kontrolle muß vorhanden sein.

Abb. 1 zeigt die Verbindungen zu dem Userport des CBM/PET-Rechner. Dieser erfüllt gerade obige Voraussetzungen. Die Spannungsversorgung des Chips erfolgt separat, da der Versorgungsspannungsbereich zwischen 7 und 14 V liegt. Mit einer 9 V Batterie, die nur mit ca. 10 mA belastet wird, ist Kompatibilität zu den TTL-Pegeln des Userports noch gegeben. Das Timing des SC-01 wird durch ein separates RC-Glied (Anschlüsse 15, 16) gesteuert und kann mit R auf beste Sprachqualität justiert werden. Die anallogen Sprachsignale werden an den Anschlüssen 20, 21, 22 ausgegeben und können über ein einfaches Tiefpaßfilter direkt in den Eingang eines üblichen Audioverstärkers gegeben werden. Wie man sieht, ist die Hardware sehr einfach aufzubauen, allerdings müssen die Phonemdaten richtig zum Userport ausgegeben werden. Dies hat durch Software zu geschehen.

Software

Listing 1 zeigt das Maschinenprogramm, welches notwendig ist, um einen String aus dem BASIC-Programmtext zu übernehmen und welches die Stringinformation mit geeigneter Handshake-Kontrolle zu Userport gibt. Mit SYS 826, D$ wird diese Routine aufgerufen. Dabei enthält D$ die Phonemdaten, die in einem separaten BASIC-Programm erzeugt werden. Die Arbeitsweise des Maschinenprogrammes ist wie folgt: Mit dem ersten JSR CDF8 (JSR CE11 für älteres Betriebssystem) wird auf ein Komma als Trennzeichen geprüft. JSR CCF9 (CCB8) wertet den folgenden Variablenausdruck aus, in unserem Beispiel die Stringvariable D$. Das Ergebnis dieser Auswertung zeigt die Abb. 2. Die RAM-Adressen 61,62 (B3,B4) in der Zeropage enthalten die Adresse des Stringdescriptors von D$. Dieser Stringdescriptor besteht aus drei Byte. Das erste enthält die Länge des Strings (deshalb Stringlänge begrenzt auf maximal 255!), während die nächsten zwei Byte des Stringdesscriptors die Startadresse des Strings beinhalten. JSR CC90 (CCA9) gibt einen TYPE MISMATCH ERROR, wenn die ausgewertete Variable kein String gewesen ist.

Die nächsten Statements initialisieren den Userport. Die Startadresse des Strings D$ wird dann in 4D, 4E (9F, A0) und die Länge des Strings in 4F (A1) mit Hilfe der indirekt indizierten Zeropageadressierung umgespeichert. Nach diesen Vorbereitungen erfolgt die Ausgabe der einzelnen Stringzeichen ab Adresse 0362, Das erste Stringzeichen wird auf die Outputleitungen des Userports gelegt und mit einem Puls an CB2 in den SC-01-Chip geschrieben. BIT E84D wartet auf einen Übergang von Low zu High am CA1-Impulseingang der VIA , was 'data taken' und eine Anforderung für neue Phonemdaten bedeutet. Nachdem alle Stringdaten zum Userport gesendet worden sind, erfolgt ein Rücksprung in das aufrufende BASIC-Programm.

Der eigentliche Phonemdatenstring wird mit dem BASIC-Programm nach Listing 2 erzeugt. Die Sprachprogrammierung erfolgt durch Verwendung der Phonemabkürzungen, die von der Herstellerfirma VOTRAX verwendet werden. Diese Abkürzungen sind in den Zeilen 20-40 aufgelistet und werden in Zeile 50 in das Stringfeld A$ (I,1) eingelesen.Das Feld A$ (i,0) wird in Zeile 10 mit den jeweils dazugehörenden Daten gefüllt, die zum Chip übergeben werden müssen. Dies ist der Initialisierungsteil des Programms. Die Phonemdaten, die die zu sprechenden Worte beschreiben, sind in den DATA-Zeilen ab Zeile 100 abgespeichert. Diese Phoneme werden in Zeile 60 in die Variable

C$ eingelesen und in Zeile 70 mit der Liste der möglichen Phoneme verglichen. Wenn Übereinstimmung festgestellt wird, wird der entsprechende Phonemcode an den String D$ abgehängt. Wenn keine Übereinstimmung gefunden wird, geht die Kontrolle an das Maschinenprogramm mit SYS 826,D$, und der Phonemdatenstring wird wiederholt vom SC-01 gesprochen.

Im Beispiel sind die englischen Zahlen one, two ... eight in den Datazeilen 100 und 110 codiert, PA1 ergibt eine Pause, um Worte zu trennen.

Mögliche Erweiterungen

Das Maschinenprogramm kann mit vorprogrammierten Phonemdatencodes in einem EPROM abgespeichert werden und im PET/CBM in einen der freien ROM-Sockel gesteckt werden. Ein BASIC-Programm kann dann diese vorprogrammierten Worte benutzen, und zwar durch Ausführung eines entsprechenden SYS-Befehles. Viele Anwendungen sind denkbar, wie z.B. eine sprechende Uhr (unter Verwendung der internen Uhr des Rechners) oder Sprachausgabe von Rechenergebnissen. Ebenso sollte es möglich sein, ein BASIC-Programm akustisch zu 'listen'. Hierzu müssen die einzelnen BASIC-Keyworte in Phonemcode abgelegt sein.

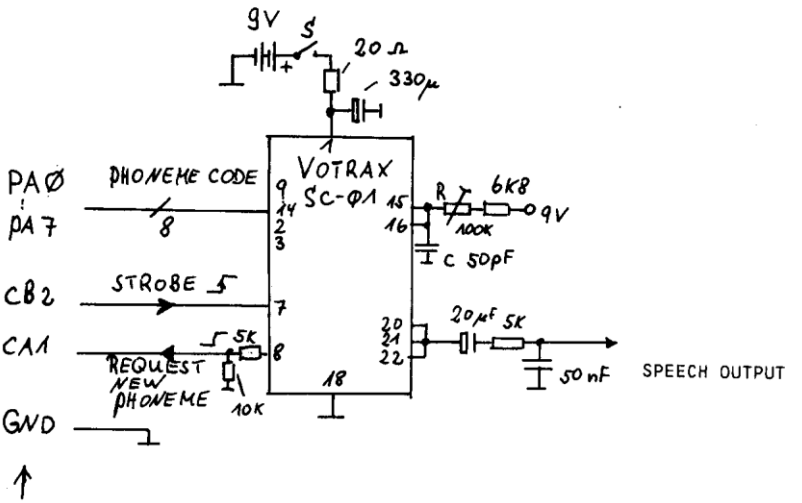
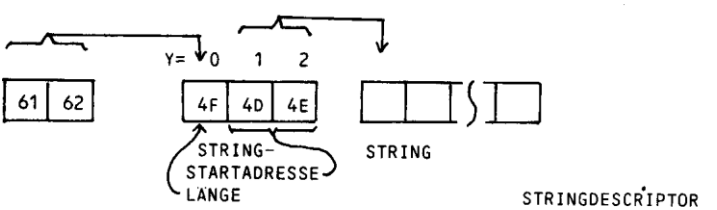


ABB 1: BESCHALTUNG



LISTING 1

```
5 DIMA$(63,1)
10 FORI=0TO63:A$(I,0)=CHR$(I):NEXT
20 DATAEH3,EH2,EH1,PAO,DT,A2,A1,ZH,AH2
30 DATA OO1,OO,L,K,J,H,G,F,D,B,A,AY
35 DATA Y1,UH3,AH,P,O,I,U,Y,T,R,E,W
```

```
 40 DATA AE,AE1,AW2,UH2,UH1,UH,O2,O1,IU,U1,THV
 45 DATA TH,ER,EH,E1,AW,PA1,STOP
 50 FORI=0TO63:READB$:A$(I,1)=B$:NEXT
 52 REM****************************
 60 READ C$
 70 FORI=1TO63: IF C$=A$(I,1)THEN90
 80 NEXT
 85 SYS826,D$: GOTO85
 90 D$=D$+A$(I,0)
 92 GOTO60
 95 REM****************************
 96 REM PHONEME DATA
 100 DATA W,AW1,N,PA1,T,U1,U1,PA1,TH,R,E,PA1
 105 DATA F,O1,R,PA1,F,AH1,Y,F,PA1,S,I,K,S,PA1
 110 DATA S,EH1,V,EH2,N,PA1,A1,AY,Y,H,T,PA1
 200 DATA STOP,*
 11085 SYS826,D$: GOTO85
READY
```

LISTING 2

```
0000            *=$033A
033A  20F8CD    JSR $CDF8       ;PRUEFT AUF KOMMA
033D  209FCC    JSR $CC9F       ;WERTET VARIABLE AUS
0340  2090CC    JSR $CC90       ;PRUEFT AUF TYPE MISMATCH ERROR
0343
0343  A9FF      LDA #$FF        ;USERPORT AUF OUTPUT
0345  8D43E8    STA $E843
0348
0348  AD4CE8    LDA $E84C
034B  291F      AND #$1F
034D  09C1      ORA #$C1        ;CB2 LOW, CA1 ACTIVE TRANSITION
034F  8D4CE8    STA $E84C               LOW TO HIGH
0352
0352  A002      LDY #$02
0354  B161      LDA ($61),Y
0356  854E      STA $4E
0358  88        DEY
0359  B161      LDA ($61),Y
035B  854D      STA $4D
035D  88        DEY             ;ERMITTELT STRINGADRESSE
035E  B161      LDA ($61),Y     ;STRINGLAENGE
0360  854F      STA $4F
0362
0362  B14D      LDA ($4D),Y     ;GIBT STRINGZEICHEN AUS
0364  8D41E8    STA $E841
0367
0367  AD4CE8    LDA $E84C
036A  09E0      ORA #$E0        ;CB2 HIGH
036C  8D4CE8    STA $E84C
036F
036F  AD4CE8    LDA $E84C
0372  291F      AND #$1F
0374  09C1      ORA #$C1        ;CB2 LOW
0376  8D4CE8    STA $E84C
0379
0379  A902      LDA #$02
037B  2C4DE8    BIT $E84D       ;CA1 HIGH?
037E  F0FB      BEQ $037B
```

```
0380        C8      INY             ;NAECHSTES ZEICHEN
0381        C44F    CPY $4F         ;LETZTES ZEICHEN?
0383        DODD    BNE $0362
0385        60      RTS
```

Ω

Peter W. Arps, 2000 Hamburg 73

Einkommensteuerberechnung 1981

Mit den folgenden Befehlen, die statt der bisherigen einzusetzen sind, wird das Einkommensteuerprogramm aus Heft 19 dieser Zeitschrift den Gesetzesänderungen für 1981 angepaßt. Die neue Programmversion kann ebenfalls vom Autor auf Cassette bezogen werden. Hierzu sind DM 20,- auf das Postscheckkonto Hamburg 3719 49-208 einzuzahlen.

```
 130 DEF  FND(D)=INT(D/54)*54
 150 REM DELETE
 460 IF LF(0)>0 THEN X=LF(0)-2000: LF(0)=-(X>0)*X
 480 Y=0: Z=2000: FOR J=0 TO 1: Y=Y+Z: IF LF(J)<0 THEN 510
 740 IF X>52800 THEN X=52800
 850 DEF  FNY(Y)=-(Y*(Y=<52800)+52800*(Y>52800))
1160 IF Y<240 THEN Y=270
1170 IF SK=3 THEN IF Y<540 THEN Y=540
1850 IF SK=2 AND KD=0 THEN Y=864
1870 Z=Z+Y: X(16)=Y
2070 Y=ZE-Z: GOSUB 2210: X=FNM(SS/ZX*100): Z=INT(X/100*Z)
2220 ZX=FND(ZE)
2230 IF ZX=<4212 THEN SS=0: RETURN
2240 IF ZX=<18000 THEN SS=ZX*.22-926: GOTO 2330
2250 IF ZX>=130000 THEN 2320
2260 IF ZX>59999 THEN 2300
2270 X=(ZX-18000)/1E4
2280 SS=FNM((FNM((FNM((FNM(3.05*X)-73.76)*X)+695)*X)+2200)*X)+3034:
GOTO 2330
2300 X=(ZX-60000)/1E4
2310 SS=FNM((FNM((FNM((FNM(.09*X)-5.45)*X)+88.13)*X)+5040)*X)+20018:
     GOTO 2330
2320 SS=.56*ZX-14837
2740 INPUT "2.1.1917 geboren (J=Ja/N=nein)  n←←←←";X$: IF X$="N"
     THEN 2770
2780 PRINT "↓↓Wurde er vor dem 2.1.1932 geboren
3340 REM DELETE
```

Hinweis des Herausgebers: Aus reprografischen Gründen waren in Heft 19 nicht alle Zeilen komplett bis zum rechten Rand ausgedruckt oder lesbar. Diese Zeilen werden hier wiederholt;

```
890 PRINT"ARBEITGEBERANTEIL ZUR RV ";RIGHT$("    "
    +STR$(INT((Y+Z)*.09)),5);

110 DIM AL(1),VB(1),WK(1),FB(5),KV(3),VV(1),LR(1),
    LF(1),GW(1),AE(3),GB(1),X(18)

910 PRINT"EINNAHMEN AUS BEAMTENTAETIGKEIT U.AE.":
    GOSUB2170:X=X1+INT(X*.09)

1120 PRINT" WECKE";:GOSUB2160:I=I+X:IFX>(GE/10)THENX=INT(GE/10+.9)
```

```
1230 Y=AL(I)+FNS(I)-FB(I):IFFB(I)>YOR(Y=0ANDFB(I)<0)
     THENX=INT(FB(I+4)*.05+.9)
1490 PRINT"EINKUENFTE EINSCHL.BAFOEG DES KINDES";:
     GOSUB2170:Z=X-2400
1910 X1=0:IF(SK<3ANDZE>24E3)OR(SK=3ANDZE>48E3)
     OR(AL(0)+AL(1)=0)THEN1990
```

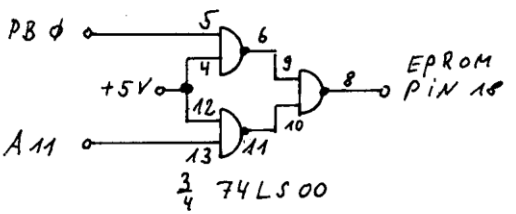
Ω

Feedback

Heft 20 enthielt Schaltung und Programm **'EPROM-Programmiereinheit für AIM 65'** aus der Feder von Herrn Peter Rix. Der Artikel fand großes Interesse und löste auch einige Rückfragen aus. Der Autor sandte daher folgende Präzisierungen:

1. Port A des Anwender-VIA wird durch die Parallelbeschaltung von zwei 6-Bit-Speichern und den Datenbus des EPROMs belastet. Schließt man jetzt noch die EPROM-Brennplatine über eine längere Kabelverbindung an den Anwenderstecker der AIM-Platine an, ***so sollten unbedingt CMOS-Speicher 40174 verwendet werden!*** Durch die höhere Portbelastung mit TTL-Speichern 74LS174 könnten die Datenpegel beim Brennen sonst bereits zu stark zusammenbrechen und Programmierfehler verursachen.

2. Das Oder-Gatter vor Pin 18 des EPROMs ist ein völlig unkritisches Element der Schaltung. Jede Gatterschaltung mit der entsprechenden Funktion kann dafür eingesetzt werden. Insbesondere eignet sich natürlich der Baustein 74LS32 (4 Oder-Gatter). Nachfolgend folgt die Beschaltung des (ausschließlich aus Kostengründen) gewählten Bausteins 74LS00 (4 NAND-Gatter):

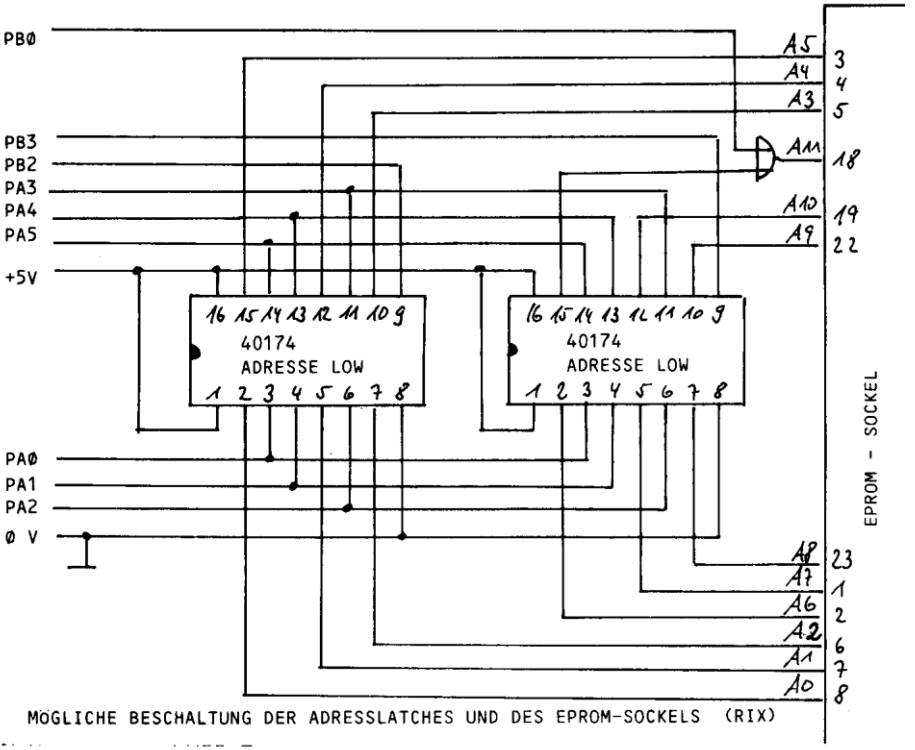


3. Die einzelnen Flip-Flops der beiden 6-Bit-Speicher sind untereinander völlig gleichberechtigt und dürfen je nach Verdrahtungserfordernissen belegt werden. Zu beachten ist jedoch, daß keine Adreßeingänge am EPROM falsch zugeordnet werden. Auf der nächsten Seite ist eine mögliche Beschaltung der Bausteine skizziert.

Ω

Die nachfolgende ***Variante der Beschaltung*** wurde von Herrn Karl-Anton Dichtel in Freiburg entwickelt. Bausteinmäßig hat sie folgende Änderungen: a) Clear des 74LS174 auf +5 Volt. b) Ein 74LS02 statt 2x 74LS00. c)zusätzlich 1/2 74LS00. d) C von 47 nF auf 10 nF, 50 V verändert.

Herr Dichtel hat sich ferner bemüht, die Bedienung sicherer zu machen. Der Schalter soll nicht unbeabsichtigt auf 'Burn' stehen. Die +25 Volt werden vielmehr kurz vor dem 50 ms-Impuls ein- und sofort danach ausgeschaltet. Der Promsockel läßt sich zum Wechsel völlig stromlos machen. - Diese Änderungen wirken sich auf das Programm aus. Nachfolgend das Listing der Veränderungen. In 0D00 und 0DAA werden die Unterprogrammaufrufe geändert. Änderungen auch im Bereich 0E5E bis 0E67. Vor die Texttabelle treten zwei Unterprogramme:

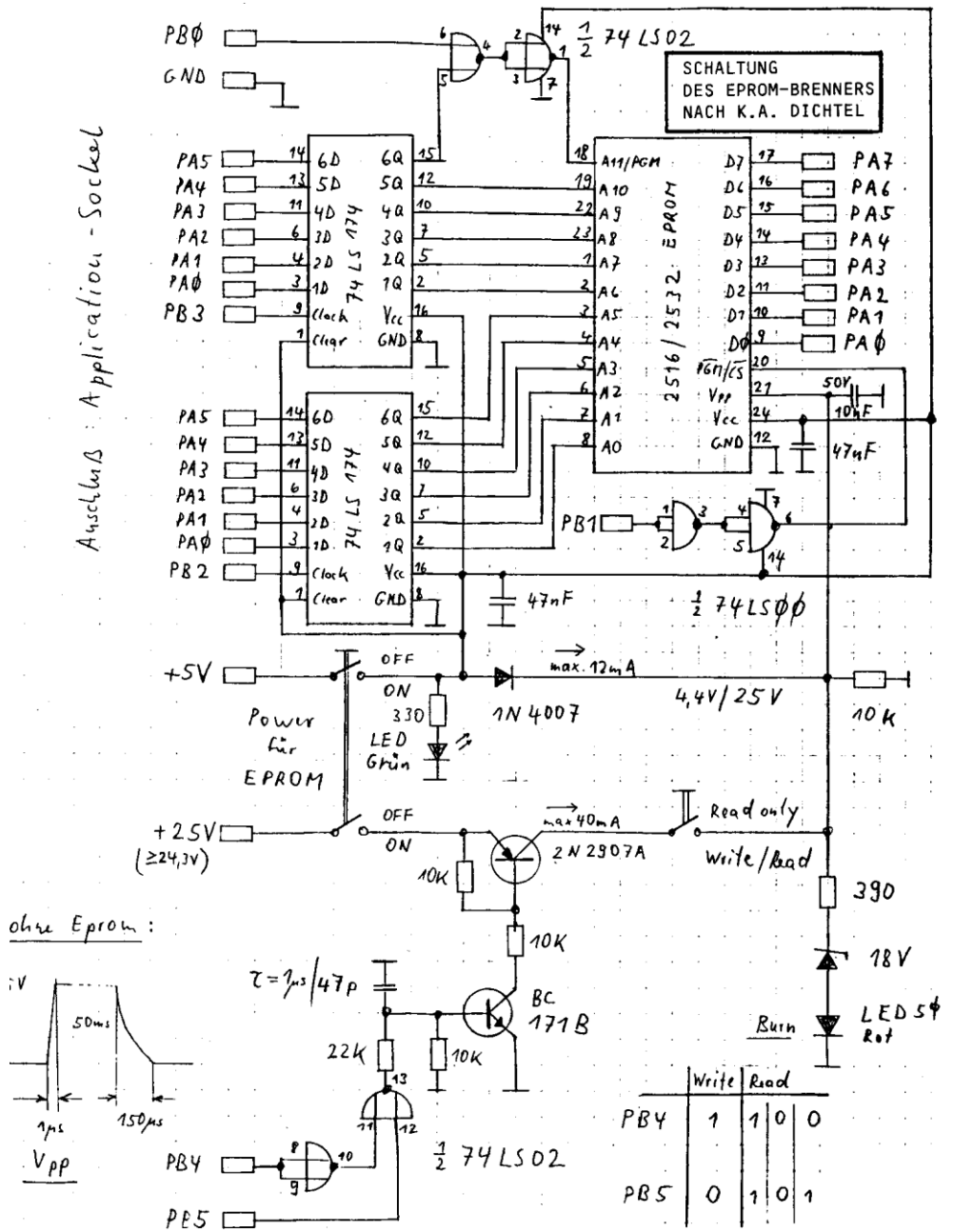


MÖGLICHE BESCHALTUNG DER ADRESSLATCHES UND DES EPROM-SOCKELS (RIX)

```
0D00                    .OPT LIS
0D00 START    204B0F    JSR VORBER
0DAA                    .OPT LIS
0DAA MENUE    204B0F    JSR VORBER
0E5E                    .OPT LIS
0E5E          A912      LDA #$12        ;25 VOLT EINSCHALTEN,
0E60          8D00A0    STA UDRB          KEIN PROGRAMMIMPULS
0E63          20400F    JSR EINSCH
0E66          EA        NOP
0E67          EA        NOP
0F32                    .OPT LIS
0F32 TEXT     B9590F    LDA TXT1,Y
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          EINSCHUB NEUER UNTERPROGRAMME
0F40 EINSCH
0F40          A913      LDA #$13        ;+25 VOLT EIN, PROGRAMMIMPULS EIN
0F42          A64C      LDX TYP
0F44          E008      CPX #$08
```

PROGRAMMÄNDERUNGEN
NACH K.-A. DICHTEL

FORTSETZUNG AUF DER 2. FOLGESEITE


SCHALTUNG
DES EPROM-BRENNERS
NACH K.A. DICHTEL
Anschluß Application - Sockel
PBØ
GND
1/2 74 LS02
PA5
PA4
PA3
PA2
PA1
PAØ
PB3
PB2
PB1
74 LS 174
2516 / 2532 EPROM
A11/PGM
A10
A9
A8
A7
A6
A5
A4
A3
A2
A1
AØ
D7
D6
D5
D4
D3
D2
D1
DØ
PGM/CS
Vpp
Vcc
GND
47nF
50V
10nF
1/2 74LSØØ
+5V
OFF
ON
max. 12mA
4,4V/25V
10K
330
LED Grün
1N 4007
Power für EPROM
+25V
(≥24,3V)
max 40mA
2N 2907A
Read only
Write/Read
390
18V
LED 5Ø Rot
Burn
ohne Eprom :
50ms
1µs
150µs
Vpp
τ = 1µs / 47p
22K
BC 171 B
1/2 74LS02
PB4
PB5
Write
Read
PB4 1 1 0 0
PB5 0 1 0 1

```
0F46            F002    BEQ OUT
0F48            A910    LDA #$10
0F4A OUT        60      RTS
0F4B            --------------------
0F4B
0F4B VORBER     A9FF    LDA #$FF
0F4D            8D03A0  STA UDDRA      ;PORT A AUF AUSGANG
0F50            A900    LDA #$00
0F52            8D01A0  STA UDRA       ;AUSGANGSSIGNAL LOW
0F55            20F0E9  JSR CRLF
0F58            60      RTS
0F59
0F59 TXT1       4D45    .BYT 'MEMORY',$A0
0F5F            A0
0F60            ;REST DER TEXTTABELLE
```

PROGRAMMÄNDERUNGEN NACH DICHTEL

Ω

Heft 20 enthielt auf den Seiten 10-12 den Beitrag **'Cross Reference List für BASIC-Variable (CBM)** von Herrn B. Dohmann. Hierzu folgender Verbesserungsvorschlag von Herrn Klaus Meyer, Berlin 19. DATA-Statements werden ausgeblendet, READ-Statements werden in den Zeilen 500-502 einbezogen. Vielen Dank für den Vorschlag!

```
400 IFA$<>CHR$(143)ANDA$<>CHR$(131)THEN500:REM & DATA
500 A=ASC(A$):IFA<128THEN505
501 REM  FOR,      INPUT,    INPUT#,  READ,    LET,      GET,      THEN
502 IFA<>129ANDA<>132ANDA<>133ANDA<>135ANDA<>136ANDA<>161ANDA<>167
    THENG=1
```

Ω

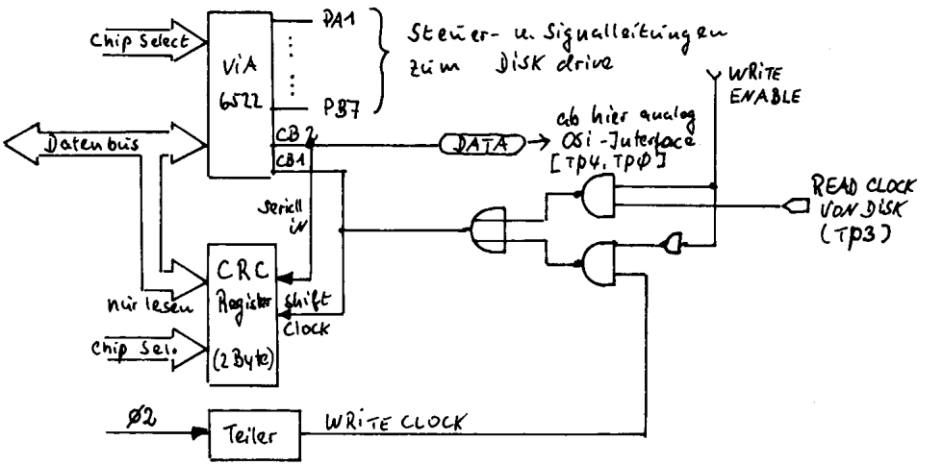
Tape-Dupe for AIM. In Heft 18 wurde ein Programm zum Kopieren von Programmen und Textfiles für den AIM 65 beschrieben. Es enthält einen Befehl zuviel, der durch 3 NOPs zu ersetzten ist. An der Speicherstelle 0247 entfällt JSR PLXY.

Ω

AIM 65 mit 'fremder Systemsoftware - Betrieb einer OSI-Floppy mit dem 610-Board. Heft 17 brachte auf den Seiten 17 ff. einen entsprechen Artikel mit Hard- und Software von Herrn Dr. A. Schnell. Dazu schreibt Herr Frank H. Koether aus Fullerton, Kalifornien:

Ich experimentiere selber mit dem vorgeschlagenen Interface und benutze die erwähnte Software. Nach dem Studium der Funktionsweise der Schaltung, der tatsächlich von OSI genutzten Signale und einem umfangreichen Studium der Fachliteratur, bin ich zu der Erkenntnis gekommen, daß man sich mit diesem Typ des Interfaces u.U. in eine Sackgasse begibt: a) Hardware- und Software-Dokumentation von OSI sind für meine Begriffe völlig unzureichend. b) Dieses Interface arbeitet mit 5" und mit 8" Floppy-Einheiten. Es ist möglich, doppelseitige Einheiten zu steuern. Jedoch ist nur eine Schreibdichte (single density-FM) verfügbar. c) Ohne Hardware-Modifizierung kann man nur 2 Einheiten kontrollieren, obwohl der Bauteileaufwand mehr zuläßt. d) Es wird ein Aufzeichnungsschema benutzt, wie es für 8"-Disketten üblich ist: 77 Tracks (sind mit DOS 3.2 kontrollierbar) aber 11 Sektoren ('a 128 Bytes ?). 13 sind tatsächlich möglich, statt 26 (bedingt durch die Clock Rate). e) Das Bitmuster der Aufzeichnung ist nicht gerade platzsparend, bedingt durch den Einsatz des ACIA 6850: 1 Startbit, 8 Datenbits, 1 Paritybit, 1 Stopbit. Dieses Format ist inkompatibel zu den meisten handelsüblichen Formaten. Zusätzlich fordert OSI, wie im Artikel dargestellt, daß in der Floppy-Disk-Einheit ein Datenseparator vorhanden ist. f) Dieses Interface bietet keinen implizierten Cyclic Redundancy Check (CRC), der gerade in der Betriebsart FM sehr leicht zu integrieren ist, ohne daß umfangreiche Softwareänderungen vorgenommen werden müssen. OSI hat dafür den Parity-Check für jedes Byte.

Sinnvoller ist es vielleicht doch, ein universell kompatibles Interface auf der Basis eines (!) VIA 6522 zu entwickeln. Damit lassen sich Disketten im üblichen single density (FM) lesen und beschreiben. Durch einen Vorsatz oder Software können dann double density-Betriebsarten (MFM, M2FM) benutzt werden. Ein 16-Bit Cyclic Redundancy Check läßt sich dort auch leicht integieren (s. Zeichnung). - In diesem Zusammenhang ist es außerdem interessant herauszufinden, in wie weit das Tapeformat des AIM 65 nach Entfernen aller Zeitverzögerungen ein Floppy-Disk-Interface unterstützen kann. (Vom Hrsg. leicht gekürzt. Die Anschrift des Lesers: 2018 Calle Alegria, Fullerton, CA 92633, USA).



PRINZIPIELLER AUFBAU EINES FLOPPY-DISK-CONTROLLERS, BETRIEBSART FM

Genutzte Signale: Headload, Drive Select 0, Drive Select 1, Step, Direction, Write Enable, Data Out (Write, ACIA) Data In (Read, ACIA), Read Clock, Index, Write Protect, Track 00. - Zusätzlich auf dem 610-Board noch "genutzt' bzw. erzeugt: Low Current, Reset, Erase Enable. Ω

Ein- und Ausgabe am AIM 65 (4)

4. Anwenderbestimmte Ein- und Ausgabe (USER)

In den Abschnitten 1.4 und 1.5 wiesen wir bereits darauf hin, daß für IN=U ein indirekter Sprung JMP ($0108) erfolgt und für OUT=U ein JMP ($010A) An diesen Stellen muß zuvor also der A dreßvektor der Anwenderprogramme hinterlegt sein. Diese müssen jeweils mit RTS enden.

Der indirekte Sprung erfolgt sowohl für die initialisierenden Kopfverteiler WHEREI und WHEREO, wie auch für die Tranportverteiler INALL und OUTALL mit dem gleichen Adreßvektor. Es liegt also in der Sorgfalt des Programmierers, zwischen Initialisierung (das Carry-Flag ist gesetzt) und Datentransport (Carry Clear) zu unterscheiden und mit BCC zu verzweigen, wenn die Durchlaufroutine in seinem Programm hinter der Initialisierungsroutine steht. Er wird dabei weiter bedenken, daß beim Herkommen von OUTALL das auszugebende Zeichen noch auf dem Stack liegt. Für die Datenausgabe ergibt sich damit folgendes grundsätzliches Ablaufschema:

```
                *=$10A          ;HINTERLEGUNG ADRESSVEKTOR
        0C01    .WORD UOUT
        ------------------

UOUT    9001    BCC WRITE       ;VERZWEIGE ZUM DATENTRANSPORT
        ;INITIALISIERE GERAET, BEFEHLSFOLGE
        60      RTS             ;RUECKKEHR NACH INITIALISIERUNG
        ------------------

WRITE   68      PLA             ;ZEICHEN VOM STACK ZURUECKHOLEN
        ;BEFEHLE DER ZEICHENABLAGE
        60      RTS             ;RUECKKEHR NACH JEDEM EINZELNEN ZEICHEN
```

Für die Verwirklichung dieser Grundsequenz hat es in dieser Zeitschrift bereits viele Programmbeispiele gegeben, in letzter Zeit z.B. 'Assembler Object Deplacer' in Heft 18 oder 'AIM 65 am IEEE 488-Bus' (Heft 19), so daß hier keine Wiederholung notwendig ist.

Im Normalfall gibt es also eine Rückkehr mit RTS nach der Initialisierung. Das muß aber nicht immer so sein. Im Artikel 'AIM 65 mit Floppy Disk CBM 3040' in diesem Heft folgt auf die Initialisierung bei USOINI lückenlos der Dump. Bei einer solchen Handhabung beachte man jedoch, daß der Stack noch Rückkehradressen der aufrufenden Programme enthält. Man sollte daher entweder den Stack durch verschiedene PLAs berichtigen oder in den Monitor zurückspringen.

Mancher AIM-Betreiber wird sich schon gewundert haben, warum nach OUT=U keine weitere Anzeige mehr erfolgt, obwohl eine Anwenderausgabe initialisiert ist. Die Begründung ist ergreifend einfach: Das OUTFLG in Adresse A413 ist auf 'U' gestellt. Damit sind die Systemeinheiten Display, Drucker und ggfs. Video automatisch abgeschaltet. Wer also die Anwenderausgabe in einer eigenen Befehlsebene aufsplitten will, muß für die Zeit der interaktiven Befehlsabfrage vorübergehend das OUTFLG wieder auf 'CR' (hex 0D) stellen. Das kann z.B. mit dem Unterprogrammaufruf JSR OUTLOW (in Adresse E901) oder auch JSR LL (E8FE) geschehen. Die letztere Routine stellt Ein- und Ausgabe auf die Systemeinheiten zurück. - Gleichsinnig ist bei der anwenderbestimmten Einabe das INFLG auf 'U' verändert, die Tastatur ist also abgeschaltet. Auch hier wird man ein 'CR' in das INFLG (Adresse A412) schreiben, wenn man von der Tastatur noch etwas regieren will.

Für die Konstruktion einer eigenen Befehlsebene in der Ausgabe folgt nun der wesentliche Ausschnitt aus einem größeren Programm. Mit JSR LL wird dort auf die Systemeinheiten zurückgeschaltet. Die eigentliche interaktive Routine ist COMMIO, sie erzeugt per JSR MESOUT einen Prompt, eine Anzeige für den Benutzer und erwartet danach per JSR KEYS eine Tastatureingabe. Die Eingaben werden in einem STRING gesammelt und jeweils durch ein Komma abgegrenzt und abgeschlossen.

```
1015            ;WE ARE COMING FROM WHEREO BY USER
1015            ;THE OBLIGATE 'BCC' IS NOT SHOWN HERE
1015 USOINI                             ;INTITIALIZE OUTPUT
1015
1015            209EEB JSR PHXY         ;SAVE REGISTERS
1018            20FEE8 JSR LL           ;RETURN OUTPUT TO NORMAL
101B USO1
101B            202613 JSR COMDIN       ;OUTPUT PROMPS & INPUT COMMANDS
101E            ;COMMANDS MAY BE INTERPRETED NOW
1187                    .OPT LIS
1187
1187            PROMPTS WHEN INPUTTING COMMANDS
1187 MES1       434F    .BYT 'COMMAND ',$BF
118F            BF
1190 MES2       4649    .BYT 'FILENAME ',$BF
1199            BF
119A MES3       4445    .BYT 'DEVICE ',$BF
11A1            BF
1326                    .OPT LIS
1326 COMDIN     A000    LDY #0          ;ADDRESSER TO PROMPTS
1328            A200    LDX #0          ;ADDRESSER TO COMMAND STRING
132A            203413 JSR COMMIO
132D            203413 JSR COMMIO
1330            203413 JSR COMMIO
1333            60      RTS
1334
1334            OUTPUT PROMPT + ECHO INPUT, ;
1334
1334 COMMIO     20F0E9 JSR CRLF         ;OUTPUT TO NEW LINE
```

```
1337            204213 JSR MESOUT
133A COMMI1
133A            204C13 JSR KEYS
133D            C92C   CMP #',  ;DELIMITER ?
133F            D0F9   BNE COMMI1
1341            60     RTS
1342            ---------------------
1342
1342 MESOUT
1342            B98711 LDA MES1,Y      ;GENERATE A PROMPT
1345            C8     INY
1346            207AE9 JSR OUTPUT
1349            10F7   BPL MESOUT
134B            60     RTS
134C            ---------------------
134C            INPUT KEY & ECHO
134C KEYS       205FE9 JSR RDRUB       ;READ WITH RUBOUT
134F            9D1F01 STA STRING,X    ;CHAR. TO STRING
1352            E8     INX
1353            60     RTS
```

Die Auffächerung der anwenderbestimmten Befehlsebene in der Initialisierungsphase ist für die Eingabe mit IN=U analog zu vollziehen. Man vergesse jedoch nicht, das OUTFLG bzw. im zweiten Fall das INFLG nach der Initialisierung auf 'U' zurückzusetzen.

R.L.

Klaus Flesch, 7814 Breisach

Monitor-Erweiterung AIM 65

Jeder, der sich etwas mehr mit dem AIM 65 beschäftigt, wird den Vorteil der Funktionstasten erkannt haben. In der Praxis werden sie von jedem Programm belegt. Damit sind sie schnell voll belegt, z.B. durch Debugger-Hilfen, Video-Monitor oder auch die DAIM-Floppy. Gleichzeitig wird z.B. für die Videoausgabe oder einen Drucker versucht, das Ausgabeformat auf eine neue Breite einzustellen. Dies führt aber dazu, daß andere Monitorkommandos nicht mehr ausgeführt werden können. Abhilfe ist dadurch zu schaffen, daß man in der Monitor-Befehlsdekodier-Routine (ab hex E182) eine Umleitung einbaut. Das ist über das Display-Link in A406/07 möglich. Der Monitor läßt nämlich zuerst das Kommando ausdrucken und druckt dann das '>'-Zeichen (carrots). Wenn man nun in der Routine, die das DILINK initialisiert, auf dieses Zeichen prüft und weiter auf dem Stack feststellt, daß der Monitor einen Befehl empfangen hat, weil die Rücksprungadresse E192 ist, dann kehrt man nicht nach E193, der Folgeadresse zurück, sondern unterwirft das Zeichen der Befehlstaste, welches sich zunächst noch auf dem Stack befindet, einer eigenen Dekodierroutine.

Im nachfolgend abgedruckten Programm wird eine vollständige Decodierung aller denkbaren Befehlstasten durchgeführt, ob sie nun bereits mit den schon bekannten Befehlen des Monitor besetzt sind oder nicht. Dazu wird aus dem Zeichenwert durch Multiplikation mit 2 ein Offset ermittelt, welcher in der Adreßworttabelle MONCOM auf die Startadresse des zu benutzenden Befehls weist. Im gewählten Beispiel wird der Memory Dump auf die Ausgabebreite von 16 Byte erweitert, statt wie üblich vier. Viele andere eigene Erweiterungen sind möglich und auch der Sinn dieses Programmes.

```
0000            ---------------------
0000            MONITORERWEITERUNG AIM 65
0000            ;BEZUEGE ZU MONITORROUTINEN
0000 ADDIN              =$EAAE
0000 BLANK              =$E83E
0000 WRITAZ             =$E2DB
```

```
0000 RD2            =$EA5D
0000 SADDR          =$EB78
0000 MEMERR         =$EB33
0000 CH31           =$E2C5
0000 NXTADD         =$E2CD
0000 MEM1           =$E2DF
0000 PHXY           =$EB9E
0000 PLXY           =$EBAC
0000 COMIN          =$E1A1
0000          ;BEZUEGE ZUM MONRAM AB $A400
0000 ADDR           =$A41C
0000 DILINK         =$A406
0000 JUMP           =$A47D
0000 COMM           =$A47E
0000
0000                *=$680
0680          AD9B06 LDA START       ;UMSETZEN DILINK VECTOR AUF START
0683          AC06A4 LDY DILINK
0686          8D06A4 STA DILINK
0689          8C9B06 STY START
068C          AD9C06 LDA START+1
068F          AC07A4 LDY DILINK+1
0692          8D07A4 STA DILINK+1
0695          8C9C06 STY START+1
0698          4CA1E1 JMP $E1A1       ;ZUM MONITOR, COMIN
069B          ;DAS UMSETZEN DER VEKTOREN LAESST EIN SCHON
069B          ;INITIALISIERTES VIDEO WEITER MITLAUFEN.
069B          ;BEI START MIT COLD RESET DIE ADRESSE VON
069B          ;START1 NACH START LADEN!
069B
069B          --------------------
069B
069B START    9D06    .WOR START1    ;INDIREKTER SPRUNGVEKTOR
069D
069D START1   48      PHA            ;RETTE REGISTER
069E          209EEB  JSR PHXY
06A1          BA      TSX            ;TRANSFER STACKPOINTER
06A2          BD0301  LDA $103,X     ;IST AUSZUGEBENDES ZEICHEN EIN '>'
06A5          C93E    CMP #'>      ;WENN JA, DANN UEBERPRUEFUNG AUF
06A7          D00E    BNE SUBEND     ;AUS $E190                ANSPRUNG
06A9          BD0501  LDA $105,X     ;WEITER AUF DEN STACK SCHAUEN
06AC          C9E1    CMP #$E1       ;WO KOMMT ER HER?
06AE          D007    BNE SUBEND
06B0          BD0401  LDA $104,X     ;NIEDRIGER ADRESSTEIL    RUFERS
06B3          C992    CMP #$92       ;STACKADRESSE IST IMMER PC+2 D.
06B5          F007    BEQ NEWMON     ;KAROTTE KAM AUS BEFEHLSABFRAGE
06B7 SUBEND   20ACEB  JSR PLXY
06BA          68      PLA            ;REGISTER WIEDERHERSTELLEN
06BB OUT      6C9B06  JMP (START)    ;NORMALE DILINK-AUSGABE
06BE
06BE NEWMON                          ;NEUE KOMPLETTE BEFEHLSDECODIERUNG
06BE          BD0601  LDA $106,X     ;HOLE AUSZUDRUCKENDES ZEICHEN
06C1          8D7EA4  STA COMM       ;NUN AUSZUFUEHRENDES COMMANDO
06C4          20ACEB  JSR PLXY
06C7          68      PLA
06C8          20BB06  JSR OUT        ;BEENDE ROUTINE AUSDRUCK
06CB          68      PLA            ;HOLE RETURNADRESSE UND
06CC          68      PLA            ;GERETTETES COMMANDO VOM STACK
```

```
06CD            68      PLA
06CE            AD7EA4  LDA COMM
06D1            C97F    CMP #$7F        ;DELETE ?
06D3            D002    BNE *+4         ;SKIP FUER DELETE
06D5            A95F    LDA #$5F        ;OFFSET FUER DELETE-TASTE
06D7            0A      ASL A           ;MULTIPLIZIERE MIT 2 UM AUS ZEICHEN
06D8            AA      TAX             ;EINE ADRESSE ZU BERECHNEN
06D9            BDE506  LDA MONCOM,X
06DC            8D7DA4  STA JUMP        ;ADRESSVEKTOR EINSETZEN
06DF            BDE606  LDA MONCOM+1,X
06E2            4CB8E1  JMP $E1B8       ;NUN BEFEHLSAUSFUEHRUNG D. REST DER
06E5            ;       ;MONITORROUTINE
06E5            ---------------------
06E5            TABELLE DER SPRUNGVEKTOREN ZU DEN BEFEHLSTASTEN
06E5            ;UNBESETZTE BEFEHLSWORTE ERHALTEN STELLTVERTRETEND 'QM'
06E5            ;DURCH ERSETZEN EINES QM (=FRAGEZEICHEN) DURCH DEN VEKTOR
06E5            ;EINER EIGENEN ROUTINE IST ES MOEGLICH, MIT DIESER TASTE
06E5            ;EIN EIGENES PROGRAMM ANZUWAEHLEN,
06E5            ;ES KOENNEN ABER AUCH MONITORROUTINEN GEAENDERT WERDEN.
06E5 MONCOM
06E5            A507    .WOR QM,QM,QM,QM,QM,QM,QM,QM    ;CONTROL $-G
06E7            A507
06E9            A507
06EB            A507
06ED            A507
06EF            A507
06F1            A507
06F3            A507
06F5            A507    .WOR QM,QM,QM,QM,QM,QM,QM,QM    ;CONTROL H-O
06F7            A507
06F9            A507
06FB            A507
06FD            A507
06FF            A507
0701            A507
0703            A507
0705            A507    .WOR QM,QM,QM,QM,QM,QM,QM,QM    ;CONTROL P-W
0707            A507
0709            A507
070B            A507
070D            A507
070F            A507
0711            A507
0713            A507
0715            A507    .WOR QM,QM,QM,QM,QM,QM,QM,QM    ;CONTROL X-^
0717            A507
0719            A507
071B            A507
071D            A507
071F            A507
0721            A507
0723            A507
0725            B507    .WOR NXT5,QM,QM,CLRBK,QM,QM,QM,QM    ;BLANK - '
0727            A507
0729            A507
072B            FEE6
072D            A507
072F            A507
```

```
0731    A507
0733    A507
0735    A507    .WOR QM,QM,CGPC,QM,QM,QM,QM,CHNGG    ;( - /
0737    A507
0739    D4E5
073B    A507
073D    A507
073F    A507
0741    A507
0743    C307
0745    A507    .WOR QM,TOGTA1,TOGTA2,VECKSM,BRKK,BASIEN,BASIRE,QM
0747    BDE6
0749    CBE6                                                  0 - 7
074B    94E6
074D    E5E6
074F    00B0
0751    03B0
0753    A507
0755    A507    .WOR QM,QM,QM,QM,QM,QM,QM,QM,SHOW    ;8 - ?
0757    A507
0759    A507
075B    A507
075D    A507
075F    A507
0761    A507
0763    A507
0765    4DE6
0767    A507    .WOR QM,CGA,BRKA,QM,DUMP,EDIT,QM,GO    ;@ - G
0769    EEE5
076B    1BE6
076D    A507
076F    3BE4
0771    39F6
0773    A507
0775    61E2
0777    65E6    .WOR SHIS,MNEENT,QM,KDISA,LOAD,MEM,ASSEM QM
0779    9EFB
077B    A507                                          ;H - O
077D    0AE7
077F    E6E2
0781    48E2
0783    00D0
0785    EAE5    .WOR CGPS,QM,REG,CGS,REENTR,UP,REGT,QM    ;P - W
0787    A507
0789    27E2
078B    FAE5
078D    CFF6
078F    EB07
0791    D9E6
0793    A507
0795    F2E5    .WOR CGX,CGY,TRACE,KEYF1,QM,KEYF2,KEYF3,QM
0797    F6E5
0799    DDE6                                          ;X - ^
079B    0C01
079D    A507
079F    0F01
07A1    1201
07A3    A507
```

```
)7A5
)7A5 QM       20D4E7 JSR $E7D4       ;FRAGEZEICHEN AUSGEBEN --> ERROR
)7A8          4CA1E1 JMP COMIN       ;BEHANDLE ALS FEHLEINGABE
)7AB          ;DEFINITION DER NORMALEN MONITOR BEFEHLE
)7AB          ;DAS LISTING IST HIER AUS PALTZGRUENDEN UNTERDRUECKT
)7AB          ;DIE ADRESSVEKTOREN SIND UNTER MONCOM AUSGEDRUCKT
)7AB                 .OPT LIS
)7AB          ---------------------
)7AB          BEISPIELE FUER 2 ANWENDERBESTIMMTE BEFEHLE
)7AB          20AEEA JSR ADDIN       ;16 STELLEN SPEICHERAUSGABE IN 1 ZEILE
)7AE          B04B   BCS SRTS
)7B0 MEIN     A210   LDX #16         ;SHOW 16 BYTES
)7B2          4CDFE2 JMP MEM1
)7B5
)7B5 NXT5     203EE8 JSR BLANK
)7B8          A010   LDY #16
)7BA          20CDE2 JSR NXTADD
)7BD          20DBE2 JSR WRITAZ
)7C0          4CB007 JMP MEIN
)7C3
)7C3 CHNGG    203EE8 JSR BLANK
)7C6          20DBE2 JSR WRITAZ
)7C9 CHNG1
)7C9          203EE8 JSR BLANK
)7CC          205DEA JSR RD2
)7CF          900A   BCC CH2
)7D1          C920   CMP #'
)7D3          D013   BNE CH3
)7D5          203EE8 JSR BLANK
)7D8          4CE307 JMP CH4
)7DB CH2      2078EB JSR SADDR
)7DE          F003   BEQ CH4
)7E0          4C33EB JMP MEMERR
)7E3 CH4      C8     INY
)7E4          C010   CPY #16
)7E6          D0E1   BNE CHNG1
)7E8 CH3      4CC5E2 JMP CH31
)7EB
)7EB UP       38     SEC             ;ZEIGE VORHERGEGANGENE 16 BYTE
)7EC          AD1CA4 LDA ADDR
)7EF          E920   SBC #32
)7F1          8D1CA4 STA ADDR
)7F4          B0BF   BCS NXT5
)7F6          CE1DA4 DEC ADDR+1
)7F9          90BA   BCC NXT5
)7FB
)7FB SRTS     60     RTS
)7FC                 .END                                        Ω
```

Kleinanzeigen

Für AIM 65: Matrixdrucker 80 Stellen (Dohmann), Metallgehäuse (Feltron), sowie Assembler-, BASIC- und PL/65-ROMs zu verkaufen. Rott, Wernhüterstraße 7, 8900 Augsburg 1, Tel. 0821/71 29 23 (nach 18 Uhr).

Kleinanzeigen kosten DM 10,- je 2 Zeilen. Betrag bitte bei Auftragserteilung überweisen.

AIM Spezial (10)

Der Assembler des AIM 65 wird etwa fünfmal schneller, wenn man in die Speicherstellen D107 bis D109 jeweils ein NOP (No Operation, Code EA) programmiert. An dieser Stelle steht der Unterprogrammaufruf JSR OUTDP1 (20 02 EF), der das zeitraubende und sichtbare zweimalige Durchlesen des Quelltextes bewirkt. Um diese Änderung zu erreichen, kann man den Assembler in ein EPROM 'umschießen' oder ihn mit dieser Änderung grundsätzlich von einem auf D000 ... decodierten RAM aus betreiben. Der Zeitgewinn ist wirklich enorm!

Ein weiterer Schönheitsfehler ist mühsamer abzuändern: Der Assembler läßt auch die Eingabe des Quelltextes mit IN=U zu, die anwenderbestimmte Eingabe. Mit ihr ist aber nach dem ersten Durchlesen des Quelltextes der Ofen aus, wie folgender Programmausschnitt zeigt:

```
DFCC FLN15    AD12A4 LDA INFLG       ;INPUT FILE = TAPE?
DFCF          C955   CMP #'U
DFD1          F007   BEQ FLN12       ;USER DEV SO CALL INIT ROUTINE
DFD3          C954   CMP #'T
DFD5          D011   BNE FLN6        ;NO DO NOT SEARCH FOR FILE
DFD7          4C2FE3 JMP LOADTA      ;FIND THE FILE
DFDA FLN12    6C0801 JMP (UIN)       ;TO USER INPUT
```

Die Abfrage CMP #'U setzt das Carry=1, wenn das INFLG 'U' enthält. Nun werden Inputroutinen so aufgebaut, daß man mit Carry=1 den Transportteil des Eingabeprogrammes ansteuert, mit dem Carry=0 jedoch den Initialisierungsteil (siehe auch 'Die Ein- und Ausgabe am AIM 65). Mit dem Carry, wie es hier vom Assembler am Schluß des ersten Durchlaufes abgeliefert wird, ist man also genau falsch bedient, man kann sein namentlich anzusprechendes externes File, das z.B. bei Diskette oder einem schnellen Magnetbandformat schon geschlossen war so nicht wiedereröffnen. Abhilfe: Man stricke das Assemblerprogramm etwas mühsamer um oder man verwerte das PASS-Flag in Zelle 0023. Im ersten Assemblerlauf ist es 00, im zweiten 02. Die Anforderung an den Eingabetreiber muß lauten: Wenn PASS=0, dann gehe mit Carry=1 immer an der Initialisierungsroutine vorbei. Wenn Carry=1 und PASS ungleich 0, dann gehe beim ersten Auftreten dieser Kombination noch einmal durch die Initialisierung des User-Files.

Lückenlos Decodierte Speicherbereiche sind prinzipiell wünschenswert. Nicht alle RAM-Karten jedoch sind in Abschnitten von 4K dekodierbar. So mag es sich ergeben, daß die RAM-Karte erst ab Adresse 2000 anschließt oder aber, daß sie wohl bei 0000 anfängt, dann muß man aber die vorhandenen RAMs von der AIM-Platine abservieren. Die bessere Idee lautet: Man lasse die RAMs dort wo sie sind, man gebe ihnen jedoch eine andere Decodierung, z.B. die des schon erwähnten Assemblers mit D000. Dazu trennt man die zwischen Chip Z16/2 und Z19/14 verlaufende Decodierbahn auf und führe auf Z19 Pin 14 stattdessen einen der von Chip Z27 abgehenden Chip Selects. Man hat dort die Wahl zwischen den Adreßblöcken 8 bis F. Mit einem Dip-Schalter kann verschiedene Decodes auf das RAM legen und somit vielleicht auch einmal Experimente mit den sonst nur in ROMs zugänglichen Sprachen machen.

FORTH: In dieser Programmiersprache wurden noch keine Programme vorgelegt. Zur Ermunterung der Leser hier ein kleiner Quelltext für die Initialisierung eines Video-Terminals mit 300 Baud Übertragungsrate.Der Adreßvektor der Durchlaufroutine DILINK wird bei ON nach A406/A407 geschrieben. Mit OFF kann man wieder auf das LED-Display zurückschalten. Die Durchlaufroutine DILINK ist in FORTH Assembler-Code formuliert. Es wird auf die übliche Art auf ein Carriage-Return geachtet. Wenn Zeilenende, dann wird zusätzlich ein Linefeed nach OUTTTY geschossen. Vor der Eingabe dieses Programmes ist FORTH auf HEX einzustellen.

```
CODE  DILINK ( ----)
   PHA,
   7F # AND,
   D # CMP,
```

```
    0= NOT IF,
    PLA,
    EEA8 JMP,
    THEN,
    A # LDA,
    EEA8 JSR,
    PLA,
    EEA8 JMP,
END-CODE
: ON ( N---N. TERMINAL AKTIVIEREN)
   HEX 12C BAUD ' DILINK A406 ! ;
: OFF ( N---N. TERMINAL AUS)
   HEX EF05 A406 ! ;
```

Für die mit FORTH mögliche knappe Formulierung hier das durchsichtige Beispiel der Umrechnung von Grad Fahrenheit auf Celsius:

```
: CELSIUS ( TF---  RECHNET FAHRENHEIT AUF CELSIUS)
  32 - 5 9 */ . ;
```

Programmlauf:

```
 AIM 65 FORTH V1.3
55 CELSIUS 12 OK
32 CELSIUS 0 OK
18 CELSIUS -7 OK
-458 CELSIUS -272 OK
```

R.L.

Editorial

Beim Durchblättern dieses Heftes wird der Leser feststellen, daß ihm die Artikel einige wesentliche Fortschritte zur Verfügung stellen. Da ist einerseits das seit langem erwartete PRINTUSING für CBM-Recher mit BASIC 3 und BASIC 4. Dieser neue Befehl läßt die formatierte Ausgabe von Zahlen auf die einfache und bequeme Art zu, wie man sie von größeren Computern her kennt.

Weiterhin ist der Datenverkehr auf dem IEEE 488-Bus transparenter geworden, wenigstens in der von CBM benutzten Version. Viele Leser und auch der Autor haben eine entsprechende Dokumentation lange Zeit sehr vermißt. Zwar hat es schon viele Veröffentlichungen zum Bus gegeben, man empfand sie oft eigentlich mehr als verwirrend,denn belehrend Auch das Buch 'PET and the IEEE 488 Bus' von Fisher und Jensen (bei Osborne McGraw-Hill) enthält noch viele Unklarheiten und verursachte dementsprechend Zeitverluste. Der Bustransfer ist dabei eigentlich einfach. Bei der zunehmenden Bedeutung des Instrumentenbusses (GPIB, General Pupose Interface Bus) sollte der Leser die bewußt knapp gehaltenen Ausführungen für seine messenden und steuernden Anwendungen näher studieren.

Im Gefolge der Busanalyse bestehen für den AIM weitere Möglichkeiten für den Anschluß einer Floppy Disk. Man wird schnell erkennen, daß die CBM-Floppy dabei einige sehr nützliche Leistungsmerkmale hat, 'sie versteht in ASCII übermittelte Befehle'.

An der Sprachen- und Utility-Front ist rege Bewegung. Wie nicht nur aus den Software-Besprechungen hervorgeht, stehen dem Programmierer heute Hilfen zur Verfügung, die den professionellen Einsatz unserer 'kleinen' Rechner weiter fördern. Diese Zeitschrift wird in Kürze eine weitere Bereicherung beisteuern: Bekanntlich ist die IBM 360/370 eine 'mainframe', ein Großrechner. Man wird demnächst den 6502 in ihrer Assemblersprache programmieren können, dabei entsteht sogar Hexcode, wie ihn die 360 benutzt. Leger könnte man sagen: Eine minframe in jedes Heim. Ernster jedoch: Eine Legion professioneller Programmierer hat den IBM-Assembler gelernt, der eine hervorragend dokumentierte und leistungsfähige Sprache ist. Für sie mag der 6502 dadurch attraktiv

werden, andererseits sollte man voraussehen, daß gute Problemlösungen in 360-Assembler im Laufe der Zeit vielleicht auch auf den 6502 übertragen werden können. Das gilt ganz sicher für das dezimale Rechnen mit beliebiger Stellenzahl. Das bisher benutzte BASIC hat da ja bekanntlich einige Schwächen. - Eines der nächsten Hefte wird weiterhin die Möglichkeiten des Multi-User-Betriebes aufzeigen.

SYSTEMS 81

Von Montag, den 19., bis Freitag, den 23. Oktober findet in München die SYSTEMS 81 statt. Der Herausgeber besucht die Messe am Montag und Dienstag und wird sich bei dieser Gelegenheit über Gespräche mit Lesern sehr freuen. Kontaktaufnahme über den Rockwell-Stand, Halle 19/501.

Ω

Aus der Branche

Philips MDCR-Laufwerk am AIM 65/PC 100. Das Ingenieurbüro Horst Minnich, Lainzer Str. 97/2 in A-1130 Wien, bietet jetzt ein 2K-Softwaremodul nebst Dokumentation und Beschaltungsvorschlag zum Betrieb der Mini-Digitalcassette an. Preis ÖS 930, also etwa DM 133. Das Softwaremodul kann vom Monitor, Editor und BASIC speichernd und ladend betrieben werden. Nach dem Speichern wird automatisch ein 'verify' durchlaufen. Es wird in Blöcken von 256 Bytes übertragen, dabei besteht ein Schutz gegen Überschreiben. Die BASIC-Extension nach Konz (FUNKSCHAU 2/81) kann ebenfalls betrieben werden. BASIC-Befehle werden als 'tokens' abgespeichert. Insgesamt ergibt sich eine höhere Geschwindigkeit für den AIM. Benutzerführung interaktiv. Anschluß des Interface an eine VIA, z.B. die Anwender-VIA. Das Programm wird als EPROM ab Adresse D000-DFFF geliefert, nach Wunsch auch für andere Adressenbereiche.

Ein Interface mit diesem Betriebsprogramm (DM 290,- + MwSt) und auch MDCR-Laufwerke können bei der Firma Horst Neudecker in Berlin bezogen werden.

IEEE 488-Interface für Olivettischreibmaschinen H+S8103. Es handelt sich um ein mit einem Microcomputer ausgestattetes Modul, das den internationalen IEEE/IEC-Bedingungen voll entspricht. Es ist über den fest an der Schreibmaschine ET xxx installierten IEEE-Bus-Stecker mit jedem (Commodore-) Computer verbindbar und kann von diesem anwenderfreundlich gesteuert und betrieben werden. Es hat einen normalen ASCII- und einen 'Commodore-Modus', bei dem die Eigenheiten des CBM-Alphabets berücksichtigt sind. Umgeschaltet wird per Programmbefehl. Zusammen z.B. mit einem WORPRO wird die Schreibmaschine zu einem Textautomaten. Dem Anwender stehen die vielen auf Tasten gelegten Funktionen der Olivetti auch zur Ansteuerung durch den Computer zur Verfügung. Per Software kann es auch auf eine andere Geräteadresse als '4' eingestellt werden. Das Interface hat einen professionellen Aufbau und kostet DM 1198,- + MwSt. Von der Lieferfirma Hard + Soft wird ebenfalls angeboten:

EXTRABASIC + 1 für Commodore-Rechner. Mit AUTO, DELETE, NUMBER, FIND, Variablen- und Array-Ausdruck, HELP, MERGE, TRACE, PAGE (Bildschirmseiten), Floppy-Kurzbefehlen, Softkey (eigene Befehlsketten) und universalem Stringsort per USR1 u.a.m..Das EXBASIC + 2 wird zusammen mit dem vorerwähnten betrieben und bringt folgende zusätzlichen Funktionen für Commodore-Rechner mit großem Bildschirm: Listing gespeicherter Programme oder Daten direkt von Floppy auf den Bildschirm, ohne Beeinflussung des gespeicherten Programmes, SPOOL-Betrieb, Programmverschlüsselung, zahlreiche Bildschirm- und Eingabemasken-Befehle, PRINT-USING. Bezug: Hard + Soft R.S. GmbH, Gagernstr. 4, 8580 Bayreuth, Tel. 0921- 68 877.

Speicherwerweiterung on board AIM 65/PC 100. Die Firma Neudecker in Berlin liefert jetzt eine sehr sinnvoll konstruierte Speichererweiterung um dynamische 32 KB RAM, die den Expansionsstecker vollkommen frei läßt. Der Anwender zieht die CPU vom Sockel und steckt stattdessen die Speicherplatine samt CPU auf diesen Platz. Diese Platine wurde am 5. September in Berlin anläßlich des Tages der offenen Tür bei Neudecker vorgestellt. Diese Veranstaltung führte viele Gäste aus Berlin, dem Bundesgebiet und dem Ausland zusammen und wurde für lange Fachgespräche auch mit dem Herausgeber genutzt.

Umrüstung 6502 auf 6809. Dem Wunsch vieler Systembetreiber, ihren 6502-Computer auch mit einem 6809 zu betreiben, kann jetzt durch verschiedene Umrüstsätze entprochen werden. SYNERTEC bietet über seine Distributoren, z.B. bitronic in München, jetzt das MOD-69 an, eine Aufsteckkarte mit CPU und Quarz, die statt der CPU 6502 in den Prozessor eingesetzt wird. Dazu gehören für SYM-1 Festwertspeicher und Handbuch für das entsprechende SUPERMON-Betriebsprogramm. Das Gleiche ist als MOD-68 für den 6802-Prozessor zu haben. - Die Firma Dohmann in Gütersloh bietet jetzt eine ähnliche Umrüstkarte an, zusätzlich ein Monitorprogramm für den AIM 65 in Festwertspeichern. Eine wohl noch etwas anders konzipierte Aufsteckkarte ist dem Vernehmen nach in Berlin in Vorbereitung.

Während damit die Betriebsprogramme für mindestens zwei Entwicklungssysteme zur Verfügung stehen, ist für Commodore-Rechner noch keine solche Firmware entdeckt worden. Entsprechende Hinweise werden daher gern entgegengenommen. Das Thema 6809 wird für Commodore-Rechner bekanntlich auch durch die angekündigte 'Mini-Mainframe' im Gehäuse der 8000er Serie aktuell. Es wird sicher Möglichkeiten geben, vorhandene CBM-Rechner in die neuen Möglichkeiten einzupassen, wenn man eine solche Umrüstplatine benutzt.

Im Sinne einer Vermehrung der Information für den Gebrauch des 6809 werden künftig auch Beiträge für solche 'Kuckucks-Computer' (Bezeichnungsvorschlag: 'cuckoo computer' oder einfach 'cuckoo')berücksichtigt. - Im vorliegenden Heft konnte aus Platzgründen kein Aufsatz für den 6809 mehr aufgenommen werden. In Heft 22 folgt zum Ausgleich mehr Information zum Thema Adressierungsarten, auch mit Beispielen.

Ω

```
10 REM PROGRAMM AB 826 (DEZ.) SPEICHERN
32 DATA78:          :REM SEI
33 DATAA9,45:       :REM LDA #$45
34 DATA85,90:       :REM STA $90
35 DATAA9,03:       :REM LDA #$03
36 DATA85,91:       :REM STA $91
37 DATA58:          :REM CLI
38 DATA60:          :REM RTS
39 DATAA5,97:       :REM LDA $97
40 DATAC9,FF:       :REM CMP #$FF
41 DATAD0,07:       :REM BNE $0352
42 DATAA9,1E:       :REM LDA #$1E
43 DATA8D,FF,03:    :REM STA $03FF
44 DATAD0,0F:       :REM BNE $0361
45 DATACE,FF,03:    :REM DEC $03FF
46 DATAD0,F9:       :REM BNE $0350
47 DATAA9,FF:       :REM LDA #$FF
48 DATA85,95:       :REM STA $97
49 DATAA9,03:       :REM LDA #$03
50 DATA85,A8:       :REM STA $A8
51 DATAD0,EC:       :REM BNE $034E
52 DATAAD,12,E8:    :REM LDA $E812
53 DATAC9,EF:       :REM CMP #$EF
54 DATAD0,18:       :REM BNE $0380
55 DATAA5,97:       :REM LDA $97
56 DATAC9,FF:       :REM CMP #$FF
57 DATAD0,12:       :REM BNE $0380
58 DATAA5,98        :REM LDA $98
59 DATAD0,0E:       :REM BNE $0380
60 DATAA9,1D:       :REM LDA #$1D
61 DATA20,D2,FF:    :REM JSR $FFD2
62 DATAA9,14:       :REM LDA #$14
63 DATA20,D2,FF:    :REM JSR $FFD2
64 DATAA9,00:       :REM LDA #$00
65 DATA85,AA        :REM STA $AA
66 DATA4C,EE,E6:    :REM JMP $E62E
```

Jürgen Rüd, 7800 Freiburg

ERASE rechts vom Cursor

Das Programm findet im zweiten Cassettenpuffer Platz und wird mit SYS 826 gestartet (CBM). Ab Zeile 32 wird der IRQ-Vektor auf das das ERASE-Programm gerichtet, das dann von Zeile 39-51 geht.

Wenn die RUN/Stop-Taste gedrückt ist, wird im REPEAT rechts vom Cursor gelöscht.

Man tippe das Programm entweder per Assembler ein oder wandle die Hexcodes größer 9 in einem BASIC-READ in entsprechende Dezimalzahlen und POKE sie ab 826 in den Speicher.

Ω

EXBASIC LEVEL II™

eingetragenes Warenzeichen von Andreas Dripke

Ein Produkt von

Michael Krause und Andreas Dripke

EXBASIC LEVEL II stellt ein stark erweitertes Basic fuer Commodore Computer der Serien 2001 (nur neue Roms!), 3001, 4001 und 8001 dar. Insgesamt stehen ueber 75 neue, aeusserst leistungsfaehige Funktionen zur Verfuegung. Die Implementierung erfolgt in zwei 4 k Eproms, die einfach in zwei freie Sockel eingesteckt werden. Zu EXBASIC LEVEL II wird eine ausfuehrliche deutsche 96 Seiten starke Anleitung mit Einbauanweisung und vielen Beispielen geliefert.

EXBASIC LEVEL II Befehlsliste:

Hilfsfunktionen:
FIND, AUTO, DEL, RENUM, TRACE (OFF), ON/OFF, DUMP, MATRIX, LETTER (OFF), FAST (OFF), STOP ON/OFF, MEM, HIMEM, ".", SPACE (OFF), GO, HELP, HELP*, BASIC, MERGE, MERGE*.

Graphikbefehle:
PRINT AT, HPLOT (320/640 x 25), VPLOT (200 x 40/80), SET, RESET, POINT.

Mathematische Funktionen:
MAX, MIN, FRAC, ROUND, ODD, RND, HEX$, DEC.

EXBASIC LEVEL II:
IF..THEN..ELSE, RESTORE Zeilennummer, ON..RESTORE Zeilennummer, REK, DISPOSE NEXT, DISPOSE RETURN, DISPOSE CLR, INPUTLINE, INPUTFORM, DEF USR=, DEF CALL=, CALL, DOKE, DEEK, VARPTR, SPACE, STRING$, INSTR, EVAL, EXEC, SWAP, SEC, BEEP / BEEP OFF, PRINT USING, ON ERROR GOTO, RESUME, RESUME NEXT, RESUME Zeilennummer, "'", HARDCOPY.

Zusaetzlich steht zur Verfuegung (je nach Serie):

2001/3001/4001: DOS 1.0 Support, Kassettenoperationen mit 5facher Geschwindigkeit, MOD.
8001: 8 Bildschirmsonderbefehle und komfortabler Assembler-/Disassembler integriert.

EXBASIC LEVEL II ist erweiterbar mit SOFTMODULEN. Standard-SOFTMODUL (wird mitgeliefert): SORT (sortiert eindim. Variablenfeld in Sekunden), CLEAR (loescht Variablenfeld), GOTO X, GOSUB X.

Sie bekommen EXBASIC LEVEL II in jedem guten Fachgeschaeft oder direkt bei INTERFACE AGE, Dalienstr. 4, 8011 Vaterstetten, Post Baldham. Der Preis: DM 392,-. 96-seitige Anleitung vorab: DM 25,- (wird spaeter angerechnet).

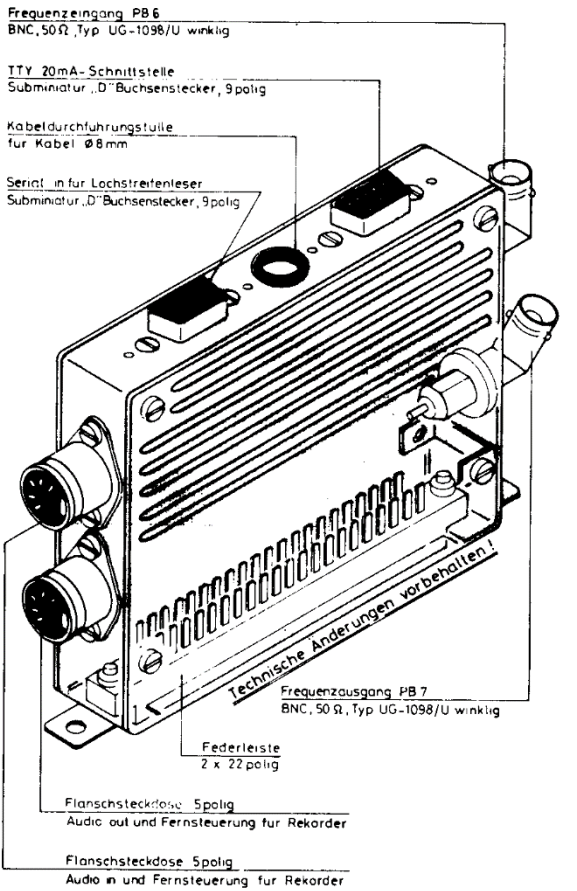
Universal-Steckverbinder

AIM 65 / Siemens PC 100

Bestückung nach Wunsch möglich

Lieferbar ab 45. KW. 81.

Preise auf Anfage



STECKER

INGENIEURBÜRO

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

Systemerweiterung für AIM 65 / PC 100

auf Europakarten

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

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

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

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

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

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

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

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

Neu: MDCR- Betriebsprogramm und -Interface,
Preis komplett DM 290,- + MWSt= 327,70 DM
Philips MDCR-Laufwerk DM 332,- + MWSt= 375,16 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,--	4116 200 ns = 8,--
4116 150 ns = 12,--	4164 200 ns = 80,--	2716 450 ns = 16,--
2532 450 ns = 38,--	6116 150 ns = 45,--	

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

DIPL.-ING. HORST NEUDECKER

INGENIEURBÜRO FÜR MESSTECHNIK

Mehringplatz 13
1000 Berlin 61

Tel.: 030 - 614 89 00
030 - 251 20 00

Video-Interface für AIM 65 / PC 100

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

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

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

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

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

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

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

Preise:	AIM65-Videointerface	615,- +MWSt=	694,35 DM
	Option 1	86,- +MWSt=	97,18 DM
	Option 2	140,- +MWSt=	158,20 DM
	NEC-Daten-Monitor 12 Zoll, grün (P31), 20 MHz, für Lichtgriffel geeignet	582,- +MWSt=	657,66 DM
	NEC-Daten-Monitor 12 Zoll, farbig (RGB), für 25 Zeilen zu je 80 Zeichen, 60 Pkte. hor.	DM 2478,- +MWSt=	2814,40 DM

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

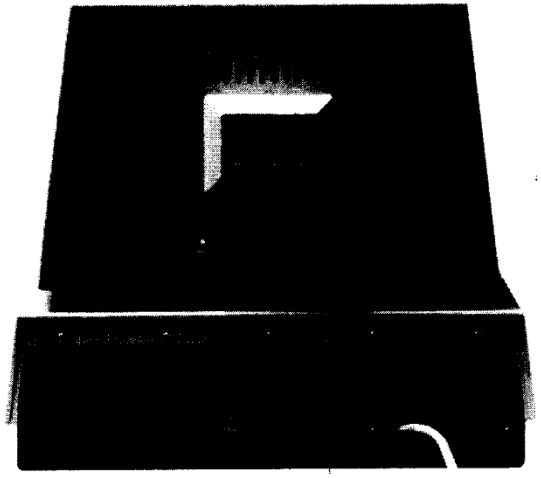
Preis für 32 KB	DM 692,- + MWSt =	DM 781,96
teilbestückt 16 KB	DM 426,- + MWSt =	DM 481,38

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

DIPL.-ING. HORST NEUDECKER

INGENIEURBÜRO FÜR MESSTECHNIK

Was hinter einem Rechner steckt . . .



. . . muß die Software beweisen!

Neu zur SYSTEMS:

IBIS – Integriertes Betriebsführungs- und Informations-System

DIOS – Datei-Informations- und Organisations-System

Bewährte Standardprogramme:

Lohn, Lager, Text, Adreßverwaltung, Faktura, Hausverwaltung, Versicherungspaket

Softwaretechnologie:

Betriebssystemerweiterungen, Interpretererweiterungen, Debug-Hilfen, Literatur, Hardwareergänzungen

-Software für Commodore: Produkte eines international anerkannten Know-hows.

Deutschland: Commodore Fachhändler
USA: AB-Computers, 252 Bethlehem-Pike, Colmar, PA 18915 Tel.: (215) 8 22 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

SYSTEMS 81 ®
Halle 1
St. 1101

Softwareverbund
Microcomputer GmbH
Scherbaumstr. 29, 8000 München 83

EDOTRONIK GmbH & Co. KG
St.-Veit-Str. 70 · D–8 München 80

GWK

GESELLSCHAFT FÜR TECHNISCHE ELEKTRONIK mbH.
Asternstr. 2, D-5120 Herzogenrath, Tel. (0 24 06) 6 23 94 · Telex: 8 32 109 gwk d

SYSTEMEXPANSION FÜR AIM 65/PC 100

- Floppy-Controller
- Video Interface
- A-D Converter
- D-A Converter
- Serielles und Parallel I/O
- Speichererweiterung RAM/EPROM
- Eprom-Programmer
- Prototyp Board
- Mother Board. Bus Buffer
- Power Supplies
- System Software

6809 COMPUTERSYSTEME AUF EUROPAKARTEN

- CPU-Karte
- Floppy-Controller
- Winchester-Controller
- Schneller A-D Converter
- D-A Converter
- Serielles und Parallel I/O
- Grafik Controller
- Ram Board 32K
- Eprom Board 16/32K
- Bus Board
- Multiuser, Multitasking bei geeignetem Betriebssystem

Wir stellen aus: SYSTEMS 81, Halle 19, Stand 505

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. - 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 7-20 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 59,--

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,--