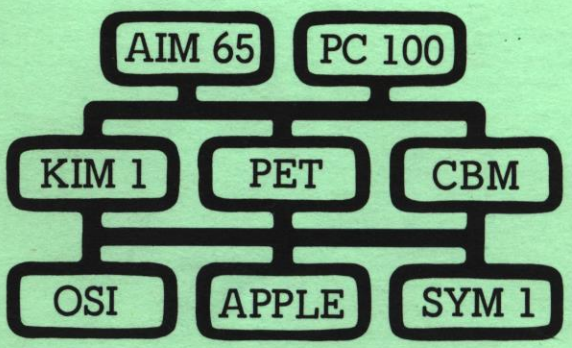
65xx MICRO MAG

COMPUTING · SOFTWARE · HOBBY

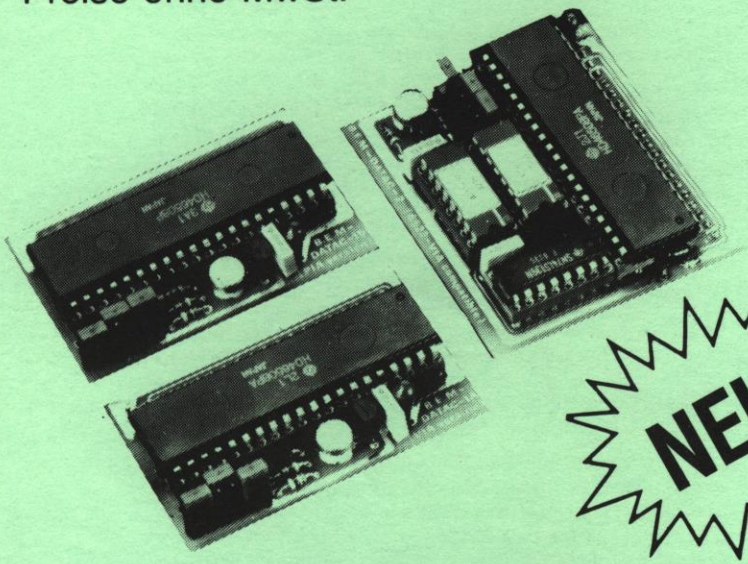
DM 9,50 Nr. 35 Februar 1984



Inhaltsverzeichnis

B.E.M.-DATAC-1A/1B und 2
Einsteckmodule für die Datenerfassung

B.E.M-DATAC-1A/1B . . . . . **DM 165.–***
B.E.M-DATAC-2 . . . . . **DM 280.–***
* Preise ohne MwSt.



NEU!

Alle Typen verfügen über 16 Analog-Eingänge mit einem 10-Bit-A/D-Umsetzer. Typ 2 besitzt außerdem zwei 8-Bit-D/A-Umsetzer. Die Modelle 1A und 2 sind VIA-kompatibel, Modell 1B läßt sich an eine PIA anschließen.

Die Einheiten werden direkt in eine PIA-Fassung (6520/6820, 6521/6821) bzw. VIA-Fassung (6522) eingesteckt.

BRUTECH ELECTRONICS

Postbus 58, 3645 ZK. Vinkeveen-Holland
Tel. 0 29 72/39 65, Telex 1 8 576 BEMIN-NL

Benutzung des erweiterten Befehlssatzes: R65C02

Programmierbeispiele für die CMOS-CPU R65C02 und ähnliche
und Anregungen auch für die MC68705xx-Einchipper

Einleitung

Die CPU 6800 von Motorola war in mancher Weise die 'Mutter' für Mikroprozessoren wie 6502 von MOS Technology/Commodore und Rockwell wie auch von Einchippern bei Motorola vom Typ 68705xx und der ebenso sehr aktuellen CMOS-CPU's von Rockwell (R65C02), GTE (G65 SCxx) mit diversen Abkömmlingen. Für die Familie 65xx darf man im Moment feststellen, daß der Befehlssatz zwar um 40 Befehle (R65C02) erweitert wurde, daß er aber aufwärtskompatibel blieb. Hier kann man also auch alte Programme mit den neuen CPU's abfahren.

Innerhalb der Familie 65xx gibt es inzwischen zahlreiche Typen mit einem gegen 6502 erweiterten Befehlssatz. Den umfangreichsten hat derzeit wohl die CPU R65C02 von Rockwell. Die weiteren Ausführungen decken daher ihre besonderen Fähigkeiten ab. Es ist darauf hinzuweisen, daß Rockwell und Motorola mit Sonderausführungen der CPU ähnlichen 'Philosophien' nachhängen, daß nämlich in der Zero Page (Speicherseite 0) die Adressen der Sonderregister liegen (Ports, Datenrichtungsregister, Timer) und daß für diese besonders elegante Befehle bereitgestellt werden sollten, nämlich die des Setzens und Löschens von einzelnen Bits in den Registern (oder Flag-Bytes) der Zero Page und solche der Programmverzweigung in Abhängigkeit von dort gesetzten/gelöschten Bits. Bei Rockwell heißen die beiden Befehlsgruppen RMBx (Reset/lösche Memory Bit x) und SMBx (Set Memory Bit x) sowie BBRx/BBSx (Branch on Bit x Reset/Set). Motorola hat die gleichwertigen mnemonischen Befehle BSETx/BCLRx (setze/lösche Bit x) bzw. für die Verzweigungen BRSETx/BRCLRx (Verweige, wenn Bit gesetzt/gelöscht - Branch on Bit Set/on Bit Clear).

Die vorgenannten Verzweigungsbefehle sind 3 Byte lang. Das erste enthält den Opcode, das zweite die Adresse in der Zero Page (Port- oder Flag-Adresse), das dritte die Verzweigungsweite/den Offset. Die Berechnung des Offsets erfolgt nach den bekannten Regeln für Verzweigungsbefehle: Das auf den Gesamtbefehl folgende Byte hat den Offset 0. Möglich sind Offsets von +127 und -128 (dezimal).

Unterschiede sind im Zeitbedarf zu bemerken. Bei Rockwell benötigt der Befehl Branch on Bit Set/Clear bei Nichtverzweigung 5 Zyklen, bei Verzweigung in gleicher Speicherseite 6 und bei Verzweigung in benachbarte Speicherseite 7 Zyklen. Bei Motorola werden immer 10 Taktzyklen angegeben, was bei der Reaktionsgeschwindigkeit des Systems ggfs. zu berücksichtigen ist.

Nachfolgend nun verschiedene Anregungen für die Ausschöpfung des neuen Befehlssatzes (im übertragenen Sinne auch für Motorola). Die Programm-Listen wurden vom hier entwickelten Assembler für die CMOS-CPU R65C02 erstellt, der zum allgemeinen Gebrauch in der bisherigen 6502-Assemblersprache geschrieben wurde.

Beispiel 1: RAM-Bytes auf 0 setzen

Wenn wir bisher eine Zelle zu 0 setzen wollten, so brauchten wir bisher zwei Befehle, künftig nur noch einen, so daß wir Zeit und Überlegungen sparen:

```
*******************************************************************************
BEISPIEL 1 - SETZE BYTE AUF NULL
0000              0001          *=$2000           ;STARTADRESSE
2000              0002 ADRESS   =$3000
2000  A9 00       0003          LDA #0            ;ALTE METHODE IN 2 BEFEHLEN
2002  8D 00 30    0004          STA ADRESS        ;ZIELADRESSE ZU 0
2005              0005
2005  9C 00 30    0006          STZ ADRESS        ;NEUE METHODE MIT 1 BEFEHL
2008              0007
```

Beispiel 2: Einen Bereich auf 0 setzen

Der Befehl STZ (Store Zero) läßt sich auch mit X indizieren, so daß sich die nachfolgende Gegenüberstellung der bisherigen und der möglichen kürzeren Programmierung ergibt:

```
*******************************************************************************
BEISPIEL 2 - BEREICH ZU 0 SETZEN
2008              0009 FELD   =$3300            ;ADRESSE ZIELFELD
2008              0010 BREITE =15               ;BENOETIGTE BREITE+1
2008  A2 0F       0011        LDX #BREITE       ;ALTE METHODE
200A  A9 00       0012        LDA #0            ;INITIALISIERUNG DER SCHLEIFE
200C  9D 00 33    0013 LOOP   STA FELD,X
200F  CA          0014        DEX
2010  10 FA       0015        BPL LOOP
2012              0016 ;NEUE METHODE MIT STZ
2012  A2 0F       0017        LDX #BREITE
2014  9E 00 33    0018 LOOP1  STZ FELD,X
2017  CA          0019        DEX
2018  10 FA       0020        BPL LOOP1
201A              0021
*******************************************************************************
BEISPIEL 3 - BIT-BEFEHL INDIZIERT
201A              0023 LAENGE =$5
201A  A2 FF       0024        LDX #$FF          ;INITIALIISERUNG MIT -1
201C  E8          0025 L2     INX               ;WEGEN INX
201D  3C 00 33    0026        BIT FELD,X
2020  10 FA       0027        BPL L2            ;BIT 7 NICHT GESETZT
2022  86 05       0028        STX LAENGE        ;LAENGE ABSPEICHERN
2024              0029
```

Beispiel 3: BIT-Befehle, indiziert

Der BIT-Befehl setzt bekanntlich in Abhängigkeit vom Operanden die Status-Flags N (negativ) und V (Bit 6, Overflow). Außerdem führt er ein logisches AND mit dem im Akkumulator enthaltenen Bitmuster aus, so daß man hinterher nicht nur mit BPL/BMI, BVC/BVS, sondern auch mit BEQ und BNE verzweigen kann. Dieser Befehl war bisher nur in der Zero Page oder absolut (extended) adressierbar. Inzwischen kann er auch mit dem Indexregister X indiziert werden, so daß sich z.B. für die Längenerkennung einer Textkette folgende Möglichkeiten ergeben, wenn im letzten Byte Bit 7 als Ende-Zeichen gesetzt ist.

Beispiel 4: BIT-Befehl immediate

Auch dieser Befehl ist neu. Im Gegensatz zu den noch zu besprechenden Befehlen TRB und TSB zerstört der BIT-Befehl keine Speicherinhalte oder den Akku. Wir können also den Akku jederzeit prüfen, ob er eines oder mehrere Bits gesetzt hat, z.B.:

```
*******************************************************************************
BEISPIEL 4 - BIT IMMEDIATE
2024              0031 PORT   =$5               ;PORT- ODER FLAGBYTE
2024              0032 MASKE  =%00100000        ;BEDINGUNGSMASKE
2024  A5 05       0033        LDA PORT          ;HOLEN EINES INHALTES
2026  89 20       0034        BIT #MASKE        ;PRUEFUNG
2028  D0 04       0035        BNE *+6           ;VERZWEIGUNG NACH BEDARF
202A              0036
*******************************************************************************
BEISPIEL 5 - BRANCH ALWAYS
202A  B8          0038        CLV               ;LOESCHE V-FLAG DEFINIERT
202B  50 EF       0039        BVC L2            ;VERZWEIGUNG ALWAS
202D              0040
202D              0041 ;BENUTZUNG STATTDESSEN VON BRA
202D  80 ED       0042        BRA L2
202F              0043
```

Beispiel 5: Branch always

Im Sinne verschieblicher Programme war es oft üblich, eine Programmierung von Sprungbefehlen möglichst zu vermeiden, indem man Verzweigungen benutzte. Mit den heute zur Verfügung stehenden Programmierhilfen und größeren Speichern ist das nicht mehr unbedingt nötig. Früher hat man dann häufig die Befehlsfolgen CLV - BVC Ziel benutzt oder die Abfolge BEQ - BNE Ziel. Heute kann man schreiben: BRA Ziel. Das hat den Vorteil besserer Dokumentation, der Gewinn ist allerdings gering.

Beispiel 6: Retten der Register Akku, X und Y

Beim Eintritt in Unterprogramme und Interruptroutinen hat es bisher einige umständliche Programmabfolgen gegeben, um die Registerinhalte zu retten, weil direkte Verbindungswege von X oder Y zum Stack nicht bestanden. Die nachfolgende Gegenüberstellung zeigt, daß man künftig mit weniger Befehlen zum Ziel kommt. Bei 65xx ist allerdings weiter zu beweinen, daß es keinen Anwender-Stackpointer gibt.

```
*******************************************************************************
BEISPIEL 6 - RETTEN DER REGISTER AKKU, X UND Y
202F  48        0045        PHA             ;ALTE METHODE: AKKU RETTEN
2030  8A        0046        TXA             ;DANN X
2031  48        0047        PHA
2032  98        0048        TYA             ;TRANSFER Y
2033  48        0049        PHA             ;ENDE DES RETTENS
2034            0050
2034  68        0051        PLA             ;REGISTER ZURUECKHOLEN
2035  A8        0052        TAY             ;ZUERST Y
2036  68        0053        PLA             ;DANN X
2037  AA        0054        TAX             ;TRANSFER
2038  68        0055        PLA             ;ZULETZT AKKU, ALTE METHODE
2039            0056
2039            0057 ;NEUE METHODE
2039  48        0058 RETTE  PHA
203A  DA        0059        PHX
203B  5A        0060        PHY             ;FERTIG
203C            0061
203C  7A        0062 RETURN PLY             ;REGISTER ZURUECK
203D  FA        0063        PLX
203E  68        0064        PLA             ;FERTIG
203F            0065
*******************************************************************************
BEISPIEL 7 - AUSTAUSCH VON X UND Y-REGISTER
203F  DA        0067        PHX
2040  5A        0068        PHY
2041  FA        0069        PLX             ;UMGEKEHRT ZURUCKHOLEN
2042  7A        0070        PLY
2043            0071
*******************************************************************************
```

Beispiel 8: Entscheidungen mit JMP (Tabelle),X

Im Sinne einer klaren Programmierung enthielt diese Zeitschrift die Empfehlung, das Prinzip 'ON GOTO' zu verwenden, z.B. in Nr. 29 (Parser und Entscheider). Die nachfolgende Gegenüberstellung bei Entscheidungsprozessen zeigt die künftigen Vereinfachungen für die Anwendung dieses Prinzips auf.

```
*******************************************************************************
BEISPIEL 8 - ENTSCHEIDUNGEN MIT JMP (TABELLE),X
2043  41 42     0073 TABEL  .BYT 'ABCDEFG'   ;Z.B. ERRLAUBTE TASTENDRUECKE
204A            0074 JUMP   =$A47D           ;SPRUNGVERTEILER
204A            0075        *=*
204A  A2 06     0076        LDX #6           ;LAENGE TABEL -1
204C  DD 43 20  0077 LX     CMP TABEL,X      ;PUEFZEICHEN SCHON IM AKKU
204F  F0 04     0078        BEQ ONGOTO       ;UEBEREINSTIMMUNG
2051  CA        0079        DEX
2052  10 F8     0080        BPL LX           ;WIEDERHOLUNG DER ABFRAGE
2054  EA        0081        NOP              ;AKTIVITAET WENN NICHTS GEFUNDEN
2055            0082
```

```
2055  8A          0083 ONGOTO TXA              ;MULTIPLIKATION VON X MIT 2
2056  0A          0084        ASL A
2057  AA          0085        TAX              ;ZURUECK, X=X*2
2058  BD 68 20    0086        LDA TAB2,X       ;LOW BYTE AUS TABELLE HOLEN
205B  8D 7E A4    0087        STA JUMP+1
205E  E8          0088        INX
205F  BD 68 20    0089        LDA TAB2,X       ;HIGH BYTE
2062  8D 7F A4    0090        STA JUMP+2       ;ZURECHTLEGEN DES VEKTORS
2065  6C 7E A4    0091        JMP (JUMP+1)     ;PROGRAMMVERZWEIGUNG
2068              0092
2068              0093 ;TAB2 ENTHAELT 7 SYMBOLISCHE ADRESSEN
2068  00 00       0094 TAB2   .WOR 0,1,2,3,4,5,6
206A  01 00       0094
206C  02 00       0094
206E  03 00       0094
2070  04 00       0094
2072  05 00       0094
2074  06 00       0094
2076              0095
*******************************************************************************
STATTDESSEN: MIT X INDIZIERTER INDIREKTER SPRUNG
2076              0097 ;AB LABEL ONGOTO IST WIE FOLGT ZU PROGRAMMIEREN:
2076  8A          0098 ONGOT2 TXA
2077  0A          0099        ASL A
2078  AA          0100        TAX              ;X=X*2 WIE OBEN
2079  7C 68 20    0101        JMP (TAB2),X     ;FERTIG, SPRUNGVERTEILER
207C              0102
*******************************************************************************
BEISPIEL 10 - INKREMENTIERUNG/DREKREMENTIERUNG DES AKK
207C  18          0104        CLC              ;ERHOEHUNG UM 1, ALTE METHODE
207D  69 01       0105        ADC #1
207F              0106
207F  1A          0107        INC A            ;NEUE METHODE
2080  3A          0108        DEC A            ;DITO ERNIEDRIGUNG
2081              0109
```

Beispiel 9: Arithmetische und logische Operationen indirekt, nicht indiziert

Eine gewisse Stärke der 6502 und ihrer Abkömmlinge liegt in der indirekten Adressierung Modulo Y, also bei Befehlen wie ADC (Pointer),Y. Im Normalfall hat man Y zu 0 oder einen anderen Wert initialisiert. Die neue indirekte Adressierungsart arbeitet ohne das 'Modulo Y'. Der Pointer in der Zeropage enthält die effektive anzusprechende Adresse, also z.B. den Zeiger auf das erste Byte eines Datensatzes oder auf eine andere Kontrollinformation. Der unmittelbare Gewinn von Befehlen wie ADC (Pointer) liegt nicht auf der Hand.

Beispiel 10: Inkrementierung/Dekrementierung des Akku

Sehr oft ist der Inhalt des Akku im Programmverlauf um 1 zu erhöhen oder zu erniedrigen. Das wurde entweder mit einer arithmetischen Operation gemacht oder auf dem Umweg über ein freies Indexregister. Die nachfolgende Gegenüberstellung zeigt die Vereinfachungen mit INC A oder DEC A.

Beispiel 11: Testen und Umschalten von Bits

Es gibt zwei neue Befehle, die dem BIT-Befehl verwandt sind, sie verändern jedoch die angesprochene Speicherzelle (Adressierung Zero Page oder absolut):

TRB Test and Reset Memory Bits with Accumulator
TSB Test and Set Memory Bits with Accumulator

Es erfolgt ein Test Akku gegen Speicherzelle (Memory) für die im Akku zu 1 gesetzten Bit-Positionen. Man kann also prüfen, ob vor der Operation ein (oder mehrere, logisch oder) Bit an entsprechender Stelle gesetzt waren. Es wird davon abhängig das Z-Flag beeinflußt, so daß man anschließend mit BEQ oder BNE verzweigen kann.

Diese Befehle bewirken zugleich ein Umschalten der angesprochenen Bitpositionen im Memory auf definiert 0 oder definiert 1. Ersteres bedeutet ein Maskieren wie mit AND. Der zweite Befehl ähnelt dem logischen ORA. Wie die Beispiele zeigen, kürzen die Befehle TRB und TSB besonders

Interfaceoperationen ab. Im Gegensatz zur nachfolgenden Gruppe sind sie nicht nur in der Zero Page wirksam, sondern extended auch im ganzen Speicher.

Beispiel 12: Setzen/Löschen einzelner Bits in der Zero Page

Die Befehle der Gruppe RMBx und SMBx (bei Motorola BCLRx und BSETx) wurden schon in der Einleitung erwähnt: Reset bzw. Set Memory Bit x. Sie sind besonders für Interfaceregister und Flags gedacht, die in der Zero Page zu adressieren sind wie beim R6501Q oder R6511 bzw. Motorola 68705. Es werden jeweils Einzelbits gelöscht oder zu '1' gesetzt:

```
*******************************************************************************
BEISPIEL 11 - TESTEN UND UMSCHALTEN VON BITS
2081            0111 FLAG   =PORT
2081  A9 20     0112        LDA #MASKE EIN BIT SOLL BEHANDELT WERDEN
2083  14 05     0113        TRB FLAG          ;BIT ZURUECKSETZEN
2085  D0 04     0114        BNE *+6           ;VERZWEIGE, WENN VORHER GESETZT
2087  A9 1B     0115        LDA #%00011011
2089  04 05     0116        TSB PORT          ;SETZEN DES BIT ODER VON BITS
208B  D0 04     0117        BNE *+6           ;VERWZ. SOBALD 1 BIT HI WAR
208D            0118
*******************************************************************************
BEISPIEL 12 - SETZEN/LOESCHEN VON EINZELBITS
208D  C7 05     0120        SMB4 PORT         ;SETZE BIT 4 IM PORT
208F  37 05     0121        RMB3 PORT         ;LOESCHE BIT 3 IM PORT
2091            0122
```

Beispiel 13: Bit-abhängige Verzweigungen

Auch die Befehle Branch on Bit Set oder Clear werden auf Flags und Interfaces in der Zero Page angewandt (s.o.), um Programmverzweigungen in Abhängigkeit von einzelnen Bits vorzunehmen. Sie erlauben eine straffe Programmierung besonders beim Warten auf Ereignisse an Interfaces. Es soll z.B. auf logisch '1' an Pin PA5 gewartet werden:

```
*******************************************************************************
BEISPIEL 13 - BRANCH ON BIT SET/CLEAR
2091  5F 05 FD  0124 WAIT   BBR5 PORT WAIT    ;WARTE SOLANGE BIT 5=LOW
2094            0125
2094            0126        .END
 ERRORS=0000
```

Bei Motorola würde es entsprechend lauten: WAIT BCLR5 PORTA WAIT

Nicht nur die größere Eleganz fällt hier auf, sondern auch eine erweiterte Syntax in der Assemblerprogrammierung: Bisher hatten die Befehle entweder keinen (inherent) oder einen Operanden. Hier finden wir erstmals 2 Operanden PORTA (die Interfaceadresse) und WAIT als Verzweigungsziel.

Zusammenfassung

Die neuen Befehle erlauben auf weite Strecken eine straffere und klarere Programmierung insbesondere beim Retten der Maschinenregister (PHX, PLX, PHY, PLY), beim Setzen/Löschen von Bits (Befehle BIT, TRB, TSB, SMBx RMBx) sowie bei Entscheidungen mit den Befehlen BBSx, BBRx und JMP (Tabelle),Y.

R.L.

Martin Albrecht, 4350 Recklinghausen
Dipl.-Ing. Rüdiger Wollenberg, 4600 Dortmund 50

DISKMON - AIM mit Floppy Disk VC-1541

Inhalt:

1. Einleitung

Das vorliegende Programm DISKMON 1.0 ist eine erweiterte Version des in /1/ und /2/ vorgestellten Monitorprogramms REMON für den AIM 65. Aus Platzgründen finden sich deshalb einige der dort aufgeführten Routinen an anderen Speicherstellen wieder.

Über den neugeschaffenen PRIOUT-Vektor ist jedes beliebige Videointerface ansteuerbar. DISKMON 1.0 läßt sich aber auch auf dem Grundgerät sinnvoll einsetzen. Im Gegensatz zu seinen Vorgängerversionen wird Kleinschrift auf dem LED-Display in Großbuchstaben ausgegeben. Beim Umschalten der Schriftart wird der eingeschaltete Zustand durch einen Prompt angezeigt.

Als wesentliche Erweiterung des AIM steht mit DISKMON 1.0 der serielle IEC-Bus der Commodore-Rechner VC-20 bzw. C-64 zur Verfügung. Damit können alle für diesen Bus konzipierten Geräte angeschlossen werden. Das gilt besonders für das preiswerte Floppydisk-Laufwerk VC-1541, das in die Reihe der Standard Input/Output-Geräte aufgenommen wurde. Die erforderliche Interfaceschaltung kostet nur ca. DM 5,- und findet sicher in jedem Gehäuse Platz (Anm. d. Hrsg.: 7406 sind am Markt verknappt!). Damit steht endlich ein preiswerter Massenspeicher für den AIM zur Verfügung.

Um die umfangreichen Routinen zur IEC-Bus-Ansteuerung unterzubringen, mußte auf die Kim-, TTY- und Thermodruckerroutinen verzichtet werden. Sie können, falls erforderlich, im RAM nachkodiert und über UIN, UOUT bzw. PRIOUT angesprochen werden.

2. Monitoränderungen

2.1 Verbesserungen

2.1-1 Korrektur des Line Assemblers: Der mnemonische Einzeilen-Assembler unter I-Kommando übernimmt jetzt LDX ABS,Y korrekt.

2.1-2 Das Unterprogramm PACK zum Packen von ASCII-Zeichen nimmt kein '@' mehr an. Darüberhinaus werden auch Kleinbuchstaben verarbeitet.

2.1-3 Die Eingabe über Lochstreifen mit IN=L ist nicht mehr möglich. Stattdessen wird mit der Routine MREAD ab der akturellen Editorzeile aus dem Speicher gelesen. Das ist insbesondere für BASIC interessant. Ein Programm kann im Editor entwickelt und über LOAD IN=L direkt eingelesen werden. Vor der ersten BASIC-Zeile ist dabei eine Leerzeile und am Schluß des Textes ein CTRL Z einzufügen.

2.1-4 Die Tape-Loadroutine wurde korrigiert. Bei $E321 steht jetzt LDX #2 statt bisher 5, so daß Input-Files immer korrekt beendet werden. Nach einem Tape-Dump wird der maskierbare Interrupt wieder freigegeben und die Motorsteuerung wird nach einem Ladevorgang auf Stop geschaltet.

2.1-5 Die Motorsteuerung für die Kassettenrecorder wird nur noch über die Taste <1> bedient. Nach ihrer Betätigung wird mit 'T=' nach Gerät 1 oder 2 gefragt. Die Taste <2> dient jetzt der Eingabe von Floppykommandos (siehe 3.3).

2.1-6 Die Editor-Prompts in spitzen Klammern werden unterdrückt.

2.1-7 Der neue RAM-Test begrenzt den Editorbereich auf hex 9FFF. Dadurch wird verhindert, daß ungewollt Registerinhalte des User-VIA in der Seite $A0 verändert werden.

2.1-8 Die Routine REPLACE des Editors wurde durch die bereits in Micro Mag Nr. 32, S. 44 gezeigte wesentlich schnellere Routine ersetzt.

2.1-9 Im Unterprogramm NAMO wurde bei hex E8ED BEQ NAMO4 durch BCS NAMO4 ersetzt, um NAMO2 mit beliebigem Inhalt im Y-Register aufrufen zu können. Diese Änderung steht im Zusammenhang mit einem floppyunterstützten Assembler, der in einer der nächsten Ausgaben veröffentlicht wird.

2.2 Erweiterungen

2.2-1 Erweiterung der Reset-Routine: Nach jedem Reset wird der IEC-Bus initialisiert. Durch Drücken der Taste 'C' während der Reset-Phase wird ein Monitor-Kaltstart erzeugt: Alle Variablen von hex A402 bis A414 und die neuen Monitorvariablen für den IEC-Bus und der PRIOUT-Vektor werden gesetzt. Die Tastatur wird auf Großschreibung geschaltet und das Printerflag in A411 steht auf 'Aus'. - Der Rechner läßt sich damit auch ohne Ausschalten aus dem 'Weltraum' zurückholen.

2.2-2 Für den Monitor und den Editor stehen die Vektoren MOLINK und EDLINK für Erweiterungen zur Verfügung. Nach Power On sind MOLINK (hex A40A/A40B) und EDLINK (A40C/0D) auf die Routine QM (in E7D4) initialisiert. Das Beispiel zeigt für MOLINK, wie die Vektoren umgesetzt werden müssen und wie eine Befehlserweiterung aussieht:

```
INIT        LDA #<ERWEITERUNG   ;Umsetzen des MOLINK
            STA MOLINK          ;Low-Byte
            LDA #>ERWEITERUNG
            STA MOLINK+1        ;High-Byte
            RTS

ERWEITERUNG CMP #'-'            ;Gültiger Befehl ?
            BNE FEHLER          ;wenn nicht, dann FEHLER
            ...                 ;'-' Kommando

            RTS                 ;Rückkehr zum Monitor

FEHLER      JMP QM              ;'?', zur. zum Monitor
```

2.2-3 Die Tastenrepeatfunktion wurde nach /2/ implementiert. Die dazugehörige Routine HELP7 befindet sich jetzt bei $EFB2. Sie konnte leicht gekürzt werden.

2.2-4 Die Umschaltung von Groß- auf Kleinschreibung wurde ebenfalls nach /2/ implementiert. Mit CTRL A wird zwischen reiner Großschreibung und gemischter Klein- Großschreibung umgeschaltet. Um auch ohne Video den akturellen Status feststellen zu können, erscheint beim Umschalten ein Prompt: ON (=Klein/Groß) bzw. OFF (=Groß). So kann man sich jederzeit über den Status informieren. Um die eingebaute TOGL-Funktion verwenden zu können, dient jetzt Bit 7 von $A408 als Flag für die Umschaltung. Auf die Funktion Display ON/OFF wurde verzichtet.

Die Kleinschreibung ist an den Monitor und den Editor angebunden. Hier können alle Befehle auch in Kleinschrift erfolgen. Die aus /1/ bekannte Routine FILT1 wandelt jetzt auch Kleinbuchstaben mit gesetzten Bit 7 zu Großbuchstaben. - Auf dem LED-Display erscheinen alle Kleinbuchstaben als Großbuchstaben. Damit ist die Kleinschreibung auch auf dem Grundsystem einsetzbar. FILT1 filtert auch Kleinbuchstaben für PACK (siehe 2.1-2) und für KEPR, so daß auf die Prompts 'IN=' und 'OUT=' in Kleinschrift geantwortet werden kann.

2.2-5 Deutscher Zeichensatz und deutsche Tastatur: Durch den Wegfall des Thermodruckers entfällt auch die Funktion der Taste PRINT. Über sie wird jetzt das ö/Ö eingegeben. Zusätzlich wurde die deutsche Tastaturbelegung eingeführt: Y und Z sind vertauscht, Ü (=F2), Ö (=PRINT) und Ä (=F1) finden sich an den genormten Plätzen. Die genaue Belegung ist unter 3.G im Listing zu finden. - Die Tastenkappen der Tastatur müssen dafür ausgetauscht werden. Sie lassen sich leicht mit einer Häkelnadel der Größe 3-3,5 nach oben abziehen.

2.2-6 Bei allen Funktionen des Betriebsprogrammes, die eine 2 Byte Hex-Eingabe verlangen, wie G, A, X, Y, P, S und insbesondere '/' ist die direkte Eingabe von ASCII-Zeichen möglich, indem zuerst ein '-Zeichen und dann das gewünschte ASCII-Zeichen eingegeben werden. Dafür ist in RD2 (bei Adresse EA5D) die Möglichkeit fortgefallen, statt ' ' (Space) auch '.' einzugeben. Die Position von OUTD7 in OUTDIS mußte verändert werden, um ein definiertes Carry Bit (C=0) zu übergeben.

2.2-7 Drucker- und Video-Anschluß über PRIOUT: Der PRIOUT-Vektor bei A478 arbeitet analog zum DILINK-Vektor für das Display. Alle Zeichen, die über OUTDP ausgegeben werden, stehen über PRIOUT zur Verfügung. Die Taste LF gibt hier ein $0A, die Taste DEL ein $7F ab. Die Dekodierung dieser Zeichen bleibt dem Anwender überlassen. CURPOS (A416) und das Printerflag (in A411) stehen für eigene Anwendungen zur Verfügung. - Vorsicht! PRIOUT darf ebenso wie DILINK nicht von Hand geändert werden!

3. Serieller IEC-Bus und Floppy-Disk VC-1541

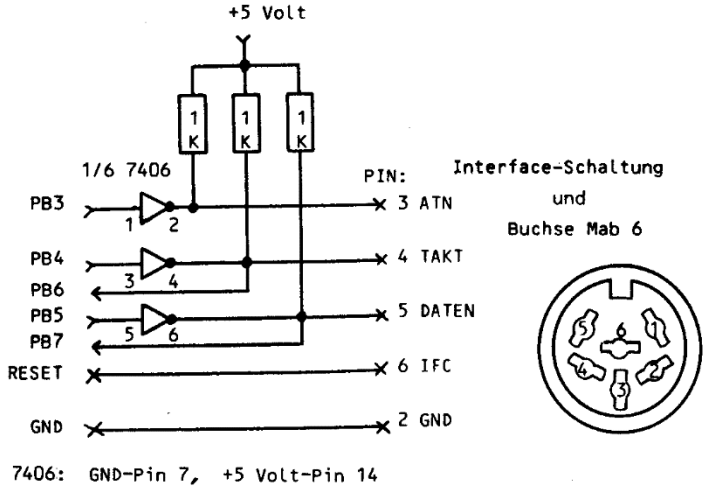
3.1 Busstruktur

Der serielle IEC-Bus ist eine abgemagerte Version des Standard IEEE-488 Bus /5,6/. Es können bis zu 15 Geräte angeschlossen und je Gerät 15 Kanäle angesprochen werden. Die Datenübertragung erfolgt seriell über eine Daten- und eine Taktleitung. Als Steuerleitungen dienen ATN und IFC. ATN (Attention) kennzeichnet die übertragene Information als Daten- oder Steuerinformation. IFC (Interface Clear) dient als Resetleitung für das Gesamtsystem.

3.2 IEC-Bus Interface

Der serielle IEC-Bus besteht aus 5 Leitungen. Die Anschlüsse IFC (=Reset) und GND (Ground=Masse) können direkt verbunden werden. Die Daten-, Takt- und ATN-Leitungen sind mit Open-Collector Invertern vom Typ SN7406N und 1k Ohm Pull-Up Widerständen versehen. Das aktive Gerät zieht damit bei Bedarf die Leitungen auf 0-Potential. Auf der Rechnerseite sind neben GND und Reset 5 Portleitungen erforderlich. Es wurden die Anschlüsse PB3 bis PB7 des User-VIA gewählt. Auf der Floppyseite erfolgt der Anschluß über eine DIN-Buchse des Typs Mab 6.

Die genaue Beschaltung geht aus dem Bild hervor. Die Interfaceschaltung wird am besten auf einer kleinen Lochrasterplatte aufgebaut und in den Leitungsweg gehängt.



3.3 Bedienung der Floppy

DISKMON 1.0 ist auf die Benutzung eines einzelnen Floppy-Laufwerkes VC-1541 abgestellt. Die Gerätenummer der Floppy ist ab Werk auf '8' eingestellt. - Das Programm unterstützt einen Input- und einen Output-Kanal. Nach dem Reset sind in INDEV Gerät 8 und Kanal 0 als Input, in OUTDEV Gerät 8 und Kanal 1 als Output gesetzt. Auf diesen Kanälen werden alle Files als PRG-Files abgelegt, siehe dazu /4/.

Das AIM 65 Input/Output-Format wurde aus Gründen der Kompatibilität mit älteren Programmen beibehalten. Die beiden 80-Zeichen-Puffer der Tape-Routine werden auch von der Floppy benutzt.

Die Bedienung der Floppy erfolgt in der gewohnten Weise: Auf die Prompts 'IN=' bzw. 'OUT=' ist mit 'F' zu antworten. Der Filename wird mit 'F=' angefordert. Er darf bis zu 16 Zeichen lang sein und auch Leerzeichen enthalten. Werden andere Floppykanäle als 0 oder 1 benutzt, dann ist zusätzlich der File-Typ (PRG, SEQ, USR, REL) und die Richtung (READ, WRITE) anzugeben. Näheres dazu in /4/.

Über die Taste (2) können Floppykommandos gegeben werden. Auf den Prompt 'DECICE' darf mit RETURN geantwortet werden, wenn das letzte aktive Gerät angesprochen werden soll. Sonst ist die Gerätenummer zweistellig anzugeben. Nach dem Prompt 'CMD:' wird der Befehlsstring eingegeben. Er wird von der Floppy ausgeführt. Währenddessen ist der Rechner für andere Aufgaben frei. Gibt man nach 'CMD:' nur RETURN ein, dann wird lediglich die Floppymeldung gelesen. Informationen über die Floppybefehle finden sich in /4/.

Fehlerzustände auf dem IEC-Bus werden mit MISSING, W-TIMEOUT und R-TIMEOUT gemeldet. Die Floppy-Fehlermeldung kann mit der Tastenfolge (2) RETURN RETURN abgerufen wer werden. Das Directory kann mit den vorhandenen Programmen nicht gelesen werden. Eine auf den AIM abgestimmte Routine steht zur Verfügung.

3.4 Verhalten der Sprachen

Die Floppydisk-Routinen sind an die Funktionen (L), (D) und (3) des Monitors und 'R' und 'L' des Editors angebunden. Alle Files werden immer ordnungsgemäß geschlossen. Weil nicht alle Sprachen die Standard-Unterprogramme des Monitors benutzen, treten folgende Besonderheiten auf:

3.4-1 Assembler: Es können keine Files direkt von der Floppy assembliert werden, da der Assembler das Input-Gerät 'F' nicht unterstützt. Object-Files werden mit OBJ-OUT=F richtig auf der Floppy abgelegt. Mit ASSMOS 1.0 steht ein passender Assembler mit komfortabler Floppyunterstützung zur Verfügung.

3.4-2 BASIC: BASIC bedient nur die Ein/Ausgabegeräte 'L', T' und 'U'. Die Floppy wird nicht unterstützt. Programme werden am besten im Editor entwickelt und mit LOAD IN=L eingelesen.

3.4-3 FORTH: FORTH beendet das Einlesen des Quelltextes hinter FINIS. Alle Lesefiles werden von DISMON normalerweise automatisch geschlossen. Nur wenn FINIS und das folgende 'CR' nicht im letzten 80-Zeichen Block stehen, bleibt das Lesefile offen. Es wird dann durch Drücken von ESC geschlossen.

3.4-4 PASCAL: Beim Einlesen von Quelltext gilt die Regel wie bei FORTH. Beim Listen aus dem PASCAL-Editor auf die Floppy wird das Schreibfile nicht geschlossen. Ein Druck auf ESC besorgt dies nachträglich.

3.4-5 PL/65 beendet das Einlesen von Quelltext hinter EXIT. Es gelten dieselben Regeln wie bei FORTH. Die PL/65 Schreibfiles werden immer ordnungsgemäß geschlossen.

Literatur:

/1/ R. Wollenberg, A. Redder: REMON - Redigierter Monitor. 65xx MICRO MAG Nr. 28, Dezember 1982
/2/ AIM Spezial (14). 65xx MICRO MAG Nr. 31, Juni 1983
/3/ R. Wollenberg, A. Redder: Schnelles Replace für AIM 65. 65xx MICRO MAG Nr. 32, August 1983
/4/ VIC-1541 Single Drive Floppy Disk User's Manual. Commodore Business Machines Inc., Dezember 1982
/5/ AIM 65 mit Floppy Disk CBM 4040. 65xx MICRO MAG Nr. 21, Oktober 1981
/6/ Der Datenverkehr Rechner und CBM-Floppy 4040. 65xx MICRO MAG Nr. 21, Oktober 1981
/7/ Englisch, Szczepanowski: Das große Floppy Buch. Data Becker, 1983

4.Speicherbelegung
Übersicht über die benutzten Speicherbereiche. Die freien Bytes können für zusätzliche Routinen verwendet werden. Im EPROM sind sie zu $FF programmiert.

von	bis	frei	ab	AIM-65	DISKMON 1.0
E11B	E13D	2	E13C	KB/TTY Bit Rate Measurement	IEC-Initialisierung/Kaltstart
E3A4	E3FC	8	E3F5	LOAD Kim-Format	IEC-Routinen
E425	E43A	-	-	GETID Kim-Format	Ergänzung zu RBYTE
E587	E5D3	-	-	Dump Kim-Format	IEC-Routinen
E80B	E836	11	E82C	Teil von PSLS	IEC-Routinen
E926	E93A	5	E936	TTY	FILTER
E9FA	EA12	-	-	Teil von CRLF	Neue Toggle-Tape Funktion
EA27	EA35	2	EA34	Teil von CRCK	IEC-Bus initialisieren
EBDB	ECOE	5	ECOA	Char. von TTY lesen	IEC-Routinen
ECOF	EC17	4	EC14	Delay für TTY	Initialisierungstabelle
EC23	EC37	1	EC37	DEHALF	Ergänzung zu OUTCK1 u. Editor-Error
EE40	EE66	-	-	Kim-Format	IEC-Routinen
EEA8	EEFB	-	-	TTY-Ausgabe	IEC-Routinen und Editor-RAM-Test
EF6D	EF74	1	EF74	Nops	Display Großschreibung
EFB2	EFFF	-	-	-	Help-Routinen
F000	F18A	-	-	Thermodruckerroutinen	IEC-Routinen
F255	F28A	-	-	Kim-Format	IEC-Routinen
F2BE	F2D6	-	-	Kim-Format	IEC-Routinen
F2D7	F420	1	F420	Druckmuster für Thermodrucker	IEC-Routinen
F6B6	F6CE	12	F6C3	Editor-RAM-Test	ECLR: CR für Editor
FA42	FA5B	26	FA42	Teil der alten REPLAC-Routine	-
FD4E	FD57	-	-	TTY	IEC-Routinen
FEB1	FEB6	6	FEB1	PATCH5	-
FEBC	FEE7	7	FEE1	Editor Kommandos lesen	Messages
FEEF	FEF7	-	-	Prompt-Routine	IEC-Routinen
FFC5	FFF9	-	-	-	IEC-Routinen

65xx MICRO MAG

```
0000          ;AIM DISKMON VERSION 1.0
0000          ;Betriebssystem für AIM-65 mit Floppy-Disk VC-1541
0000          ;(c) 1983 : M. Albrecht, A. Redder, R.Wollenberg
0000          ;Monitor-Unterprogramme und Variablen hier nicht gelistet
0000                .OPT LIS
0000          ;Neue Monitorvariablen (nach Gruppen geordnet)
0000          ;bei Kaltstart (C) bzw. Warmstart (W) auf den angegebenen
0000          ;Wert initialisiert.
0000 IECFLG         =$A474          ;C W $20   Flag für IEC-Routinen
0000          ;Bit7=1: Zeichen gespeichert, Bit6: Sendeflag,
0000          ;Bit5=1: Meldung des Kommandokanals anzeigen
0000          ;Bit0=1: Input File offen, Bit1=1: Output File offen
0000 LASTDV         =$A475          ;C   $08   letztes aktives IEC-Gerät
0000          ;Alle Floppyfiles werden als PRG-Files angelegt.
0000          ;Bei anderen Kanälen als 0 und 1 muß beim Filenamen der
0000          ;Filetyp explizit angegeben werden:
0000          ;Filename,Typ(PRG/SEQ/USR/REL),Richtung(READ/WRITE)
0000 INDEV          =$A476          ;C   $80   Input-Gerät (8) und Kanal (0)
0000 OUTDEV         =$A477          ;C   $81   Output-Gerät (8) und Kanal (1)
0000 PRIOUT         =$A478          ;C   $E119 Vektor auf Benutzer-Druckroutine
0000 FEHLER         =$A47A          ;Fehlernummer von Kanal 15
0000 SPUR           =$A47B          ;Spurnummer
0000 SEKTOR         =$A47C          ;Sektornummer
0000 RESBYT         =$A47D          ;Resultat beim lesen
0000 BITCNT         =$A47E          ;Bitzähler
0000 STATUS         =$A417          ;IEC Busstatus (Timeout, EOF)
0000 CBUF           =$A418          ;Buffer für ein Zeichen
0000 MOLINK         =$A40A          ;C   $E7D4 Vektor für Monitorerweiterung
0000 EDLINK         =$A40C          ;C   $E7D4 Vektor für Editorerweiterung
0000 LAE            =$A7            ;Länge des Floppy-Filenamens
0000 TXT            =$A8            ;Ort des Floppy-Filenamens
0000
0000          ;User-Via Adressen
0000 PORTB          =$A000          ;Data Register B
0000 DDRB           =$A002          ;Data Direction Register B
0000 T2LL           =$A008          ;Timer 2 Latch Low
0000 T2CH           =$A009          ;Timer 2 Counter High
0000 ACR            =$A00B          ;Auxiliary Control Register
0000 IFR            =$A00D          ;Interrupt Flag Register
0000 IER            =$A00E          ;Interrupt Enable Register
0000
0000          ;Timerkonstanten bezogen auf 1MHz Takt
0000 CONST1         =4              ;für 1024 Microsec. (T2LL=0)
0000 CONST2         =2              ;für 512 Microsec. (T2LL=0)
0000
0000          ;sonstige Konstanten
0000 CR             =13             ;Carriage Return
0000 TMSGM1         =$4D            ;Konstante für KEP ' F='
0000
0000          ;-------------------------------------------------------
0000          ;                  Teil 1: Verbesserungen
0000          ;-------------------------------------------------------
0000                .OPT LIS
0000          ;------------------ 1.A ----------------------------------
0000          ;Korrektur des Line-Assemblers:
0000          ;LDX Abs.,Y wird korrekt übernommen
0000                *=$FB1D
FB1D          02    .BYT 2            ;statt bisher 0
FB1E          ;------------------ 1.B ----------------------------------
FB1E          ;Korrektur von PACK:
FB1E          ;Die alte Version ließ '§' ($40) als Buchstaben zu
FB1E                *=$EA90
EA90          C941  CMP #'A'          ;jetzt Test auf 'A' ($41)
EA92          ;------------------ 1.C ----------------------------------
EA92          ;IN=L liest mit MREAD aus dem Editor (für Basic)
EA92                *=$E9B7
E9B7          4CD0FA JMP MREAD        ;statt bisher JMP GETTTY
E9BA          ;------------------ 1.D ----------------------------------
E9BA          ;Korrektur der LOAD-Routine:
E9BA                *=$E321
E321 LOAD4    A202  LDX #2            ;statt bisher 5 nur 2 Bytes lesen
E323          ;Interrupt nach Tape-Dump wieder zulassen
E323                *=$E523
E523          58    CLI               ;statt bisher CLC
E524          ;Kassettenlaufwerke am Ende eines Ladevorgangs stoppen
E524                *=$E529
E529          EA    NOP               ;5 mal NOP, hier nicht gelistet !
E52E                .OPT LIS
E52E          ;Neue Toggle-Tape-Funktion nur noch über Taste <1>
```

```
E52E            ;<1>T= 1 oder 2 für Tape 1 oder 2 eingeben
E52E            ;Taste <2> wurde mit Floppykommando belegt
E52E                    *=$E215
E215            FAE9    .WOR NEWTOG,FCMD  ;Neue Einsprünge
E217            A8EE
E219                    *=$E9FA
E9FA NEWTOG     A050    LDY #$50
E9FC            2070E9  JSR KEPR          ;'T=' ausgeben
E9FF            C931    CMP #'1'          ;"1" ?
EA01            D003    BNE NTOG1
EA03            4CBDE6  JMP TOGTA1        ;Toggle Tape 1
EA06 NTOG1      C932    CMP #'2'          ;"2" ?
EA08            D003    BNE NTOG2
EA0A            4CCBE6  JMP TOGTA2        ;Toggle Tape 2
EA0D NTOG2      20D4E7  JSR QM            ;Fehler, "?" ausgeben
EA10            4CFAE9  JMP NEWTOG        ;nochmal
EA13            ;------------------ 1.E ---------------------------------
EA13            ;unterdrücke Editor Prompts
EA13                    *=$FA8A
FA8A            203CE9  JSR READ          ;keine =< > Prompts mehr
FA8D            ;------------------ 1.F ---------------------------------
FA8D            ;Neuer Editor-Ram-Test ersetzt den alten Test bei $F6B6,
FA8D            ;der bei Ram bis $A000 in die Seite $A0 testet und den
FA8D            ;User-Via beeinflußt
FA8D                    *=$EEDC
EEDC EDI        AD1DA4  LDA ADDR+1
EEDF            C9A0    CMP #$A0          ;teste auf Seite $A0
EEE1            B017    BCS EDI2C         ;erreicht
EEE3            A000    LDY #0            ;teste ob schreibfähig (Ram)
EEE5            20B7FE  JSR PATCH6        ;hole Byte (ADDR)
EEE8            48      PHA               ;rette es
EEE9            A9AA    LDA #$AA          ;Schreibversuch mit $AA
EEEB            2078EB  JSR SADDR
EEEE            D009    BNE EDI2B         ;schreibt nicht
EEF0            68      PLA
EEF1            2078EB  JSR SADDR         ;Byte wiederherstellen
EEF4            EE1DA4  INC ADDR+1        ;nächste Seite
EEF7            18      CLC               ;C=0 wenn Ram
EEF8            60      RTS
EEF9 EDI2B      68      PLA
EEFA EDI2C      38      SEC               ;C=1 wenn Rom
EEFB            60      RTS
EEFC            ;Anbindung an den Editor
EEFC                    *=$F673
F673 EDI3       20DCEE  JSR EDI
F676                    *=$F68D
F68D            20DCEE  JSR EDI
F690            ;------------------ 1.G ---------------------------------
F690            ;Schnelle Replace Routine für den Editor nach
F690            ;Redder/Wollenberg, Micro-Mag Nr.32, S.44
F690                    *=$F93F
F93F            ;hier nicht gelistet
FA42                    .OPT LIS
FA42            ;------------------ 1.H ---------------------------------
FA42            ;Korrektur von PLNE: PLNE gibt die aktuelle Zeile nur noch
FA42            ;über OUTALL aus und schreibt nicht mehr in DIBUFF.
FA42                    *=$F734
F734            EA      NOP               ;STA DIBUFF,Y entfällt
F735            EA      NOP
F736            EA      NOP
F737            ;------------------ 1.I ---------------------------------
F737            ;Änderung in NAMO2 für Assembler
F737                    *=$E8ED
E8ED            B006    BCS NAMO4         ;statt bisher BEQ NAMO4
E8EF
E8EF            ;-----------------------------------------------------------
E8EF            ;  Teil 2: Änderungen, weil bestimmte Routinen wegfallen
E8EF            ;-----------------------------------------------------------
E8EF                    .OPT LIS
E8EF            ;------------------ 2.A ---------------------------------
E8EF            ;Änderungen bei Entfernen der Kim-Routinen
E8EF                    *=$E46D
E46D            EA      NOP               ;11 mal NOP, hier nicht gelistet !
E478                    .OPT LIS
E478                    *=$EE3B
EE3B            A9C7    LDA #$C7          ;nur Tape
EE3D            EA      NOP
```

```
EE3E                    *=$EE8E
EE8E        A9C7        LDA #$C7        ;nur Tape
EE90        EA          NOP
EE91                    *=$EE94
EE94        C9C7        CMP #$C7        ;nur Tape
EE96        EA          NOP
EE97                    *=$EE9A
EE9A        C9C7        CMP #$C7        ;nur Tape
EE9C        EA          NOP
EE9D                    *=$F21D
F21D        EA          NOP             ;JSR SETSPD entfällt
F21E        EA          NOP
F21F        EA          NOP
F220                    *=$F252
F252        4C90F2 JMP OUTTA1           ;Test auf TSPEED entfällt
F255        ;---------------- 2.B ----------------------------------
F255        ;Änderung bei Entfernen der TTY-Routinen
F255        ;TTY-Test anpassen
F255                    *=$E844
E844        EA          NOP             ;kein BIT DRB mehr
E845        EA          NOP             ;damit immer <> 0
E846        EA          NOP
E847        ;Anpassung in OUTPUT
E847                    *=$E98B
E98B        EA          NOP             ;kein TTY
E98C        EA          NOP
E98D        EA          NOP
E98E        ;Start von CRCK ändert sich
E98E                    *=$EA24
EA24        4C36EA JMP $EA36
EA27                    *=$FD4B
FD4B        4C58FD JMP $FD58            ;kein TTY mehr
FD4E
FD4E                    *=$F7C8
F7C8        4CB6F6 JMP ECLR             ;nur CLR wenn INFLG ungleich $0D
F7CB                    *=$F6B6
F6B6 ECLR   AD12A4 LDA INFLG            ;Input Device ?
F6B9        C90D    CMP #CR             ;Tastatur ?
F6BB        D003    BNE NURLED          ;nein, nur CLR für LEDs
F6BD        4C24EA JMP CRCK             ;CR
F6C0 NURLED 4C44EB JMP CLR              ;nur LEDs
F6C3        ;---------------- 2.C ----------------------------------
F6C3        ;Änderung der Routine PROMPT
F6C3        ;JMP PATC11 entfällt
F6C3                    *=$E7C2
E7C2        F0F8    BEQ $E7BC           ;RTS bei 'T'
E7C4        C94C    CMP #'L'            ;'L' ?
E7C6        F0F4    BEQ $E7BC           ;RTS bei 'L'
E7C8
E7C8        ;-------------------------------------------------------
E7C8        ;                  Teil 3: Erweiterungen
E7C8        ;-------------------------------------------------------
E7C8                    .OPT LIS
E7C8        ;---------------- 3.A ----------------------------------
E7C8        ;Erweiterung der Reset-Routine:
E7C8        ;nach jedem Reset wird der IEC-Bus initialisiert und
E7C8        ;die Kaltstarttaste 'C' abgefragt.
E7C8                    *=$E0E3
E0E3        D04A    BNE CUINIT          ;Kaltstart: setze alle Var.
E0E5                    *=$E0EB
E0EB        D042    BNE CUINIT          ;Kaltstart
E0ED                    *=$E11B
E11B        ;Warmstart: wird bei jedem Reset durchlaufen
E11B WUINIT 2027EA JSR SETIEC           ;IEC-Bus setzen
E11E        A920    LDA #$20            ;IECFLG setzen
E120        8D74A4 STA IECFLG
E123        A9FE    LDA #%11111110  ;Tastatur "C" abfragen
E125        8D80A4 STA DRA2
E128        AD82A4 LDA DRB2
E12B        C9DF    CMP #%11011111  ;"C" ?
E12D        D015    BNE RS7             ;nur Warmstart
E12F        ;Kaltstart: wird bei geändertem NMI-Vektor oder bei
E12F        ;gedrückter Taste 'C' durchlaufen
E12F CUINIT A204    LDX #4              ;Kaltstart: alle Var. setzen
E131 NEXTIN BD0FEC LDA UTAB,X
E134        9D75A4 STA $A475,X
E137        CA      DEX
```

```
E138            10F7    BPL NEXTIN
E13A            30B5    BMI RS3A        ;weiter mit Originalroutine
E13C            ;IEC-Bus initialisieren
E13C                    *=$EA27
EA27 SETIEC     AD02A0  LDA DDRB
EA2A            293F    AND #%00111111  ;Daten Eing., Takt Eing.
EA2C            0938    ORA #%00111000  ;Daten Ausg., Takt Ausg., ATN Ausg.
EA2E            8D02A0  STA DDRB
EA31            4C9CE5  JMP U3
EA34            ;Kaltstartwerte von IECFLG bis PRIOUT
EA34                    *=$EC0F
EC0F UTAB       08      .BYT 8,$80,$81,$19,$E1
EC10            80
EC11            81
EC12            19
EC13            E1
EC14            ;Caps-Lock nach Kaltstart:
EC14                    *=$E75C
E75C            00      .BYT 0          ;Flag auf 0
E75D            ;Printerflag ( $A411 ) auf 0 = Printer Off initialisieren:
E75D                    *=$E765
E765            00      .BYT 0
E766            ;Neue Systemmeldung nach Reset:
E766                    *=$FF88
FF88            2020    .BYT '  AIM DISKMON 1.0'
FF99            ;------------------ 3.B ---------------------------------
FF99            ;Erweiterungsmöglichkeit der Monitor- und Editorbefehle
FF99            ;über MOLINK und EDLINK Vektor:
FF99            ;Dekodierroutinen für weitere Eintastenbefehle können
FF99            ;über die Vektoren angesprungen werden.
FF99            ;für den Monitor:
FF99                    *=$E19E
E19E            20EFEF  JSR MOERW       ;statt JSR QM
E1A1                    *=$EFEF
EFEF MOERW      6C0AA4  JMP (MOLINK)    ;zur Benutzerroutine
EFF2            ;für den Editor:
EFF2                    *=$FA9A
FA9A            20F2EF  JSR EDERW       ;statt JSR QM
FA9D                    *=$EFF2
EFF2 EDERW      6C0CA4  JMP (EDLINK)    ;zur Benutzerroutine
EFF5            ;Initialisierung von MOLINK und EDLINK auf die Routine QM:
EFF5                    *=$E75E
E75E            D4E7    .WOR QM,QM
E760            D4E7
E762            ;------------------ 3.C ---------------------------------
E762            ;Tastenrepeatfunktion
E762            ;nach Redder/Wollenberg, Micro Mag Nr.31, S.23 ff
E762                    *=$ECEF
ECEF ROONEK     8A      TXA             ;rette X
ECF0            48      PHA
ECF1            A280    LDX #$80        ;langes Repeat
ECF3            AD08A4  LDA $A408
ECF6            6A      ROR A
ECF7            9002    BCC *+4
ECF9            A20A    LDX #$0A        ;kurzes Repeat
ECFB            20B2EF  JSR HELP7
ECFE            68      PLA
ECFF            AA      TAX
ED00            ;dazu HELP7:
ED00                    *=$EFB2
EFB2 HELP7      AD82A4  LDA DRB2
EFB5            0D7FA4  ORA ROLLFL
EFB8            49FF    EOR #$FF
EFBA            D008    BNE *+10
EFBC            A9FE    LDA #$FE        ;andere Taste
EFBE            2D08A4  AND $A408
EFC1            4CCFEF  JMP HELP7B
EFC4 HELP7A     2036ED  JSR $ED36
EFC7            CA      DEX
EFC8            D0E8    BNE HELP7
EFCA            A901    LDA #1
EFCC            0D08A4  ORA $A408
EFCF HELP7B     8D08A4  STA $A408
EFD2            60      RTS
EFD3            ;------------------ 3.D ---------------------------------
EFD3            ;Groß- und Kleinschreibung
EFD3            ;nach Redder/Wollenberg, Micro Mag Nr.28, S.8 ff
```

```
EFD3          ;Die Umschaltung erfolgt mit CTRL A, die Prompts
EFD3          ;ON und OFF geben den Zustand (OFF = Caps-Lock) wieder
EFD3 KLEIN    48      PHA
EFD4          8A      TXA
EFD5          D00A    BNE GROS
EFD7          68      PLA
EFD8          48      PHA
EFD9          2940    AND #$40
EFDB          F004    BEQ GROS
EFDD          68      PLA
EFDE          0920    ORA #$20
EFE0          48      PHA
EFE1 GROS     68      PLA
EFE2          60      RTS
EFE3          ;dazu HELP3
EFE3 HELP3    2040EC JSR GETKEY
EFE6          2C08A4 BIT $A408
EFE9          1003    BPL *+5
EFEB          20D3EF JSR KLEIN
EFEE          60      RTS
EFEF          ;Umschaltung Caps-Lock auf Klein/Groß
EFEF          ;mit CTRL A ($1)
EFEF          ;$A408 Bit 7 dient zur Umschaltung Groß/Klein
EFEF                  *=$ECE1

ECE1          C901    CMP #1
ECE3          D006    BNE GETK14
ECE5          4C7BFE JMP TOGKG        ;Toggle Caps-Lock Bit
ECE8                  *=$FE7B
FE7B TOGKG    A208    LDX #$08
FE7D          20E7E6 JSR TOGL
FE80          4CD0EC JMP $ECD0
FE83          ;Anbindung an die Routine READ:
FE83                  *=$E94A
E94A READ1    20E3EF JSR HELP3
E94D          ;Umwandlung von Klein- in Großbuchstaben
E94D                  *=$E926
E926 FILTER   2096FE JSR RED1
E929 FILT1    48      PHA
E92A          297F    AND #%01111111
E92C          C961    CMP #$61
E92E          9004    BCC NOFI
E930          68      PLA
E931          29DF    AND #$DF
E933          60      RTS
E934 NOFI     68      PLA
E935          60      RTS
E936          ;Anbindung an den Monitor:
E936                  *=$E18A
E18A          2026E9 JSR FILTER
E18D          ;Anbindung an den Editor:
E18D                  *=$FA8F
FA8F CD02     2029E9 JSR FILT1
FA92 LABEL1   DDACFA CMP COMTBL,X
FA95          F009    BEQ CFND1
FA97          CA      DEX
FA98          10F8    BPL LABEL1
FA9A          20F2EF JSR EDERW        ;Zur Befehlserweiterung
FA9D          4C24EA JMP CRCK
FAA0 CFND1    201AFF JSR PATC15+3
FAA3          ;Anbindung an die Displayroutine. Das Display zeigt
FAA3          ;Großbuchstaben für Kleinbuchstaben:
FAA3                  *=$EF9C
EF9C          206DEF JSR NURGRO
EF9F                  *=$EF6D
EF6D NURGRO   2029E9 JSR FILT1
EF70          8D02AC STA $AC02
EF73          60      RTS
EF74          ;Anbindung an PACK:
EF74          ;Pack verarbeitet jetzt auch Kleinbuchstaben
EF74                  *=$EA84
EA84 PACK     4CF5EF JMP PHELP
EA87          EA      NOP
EA88                  *=$EFF5
EFF5 PHELP    2029E9 JSR FILT1
EFF8          C930    CMP #$30
EFFA          9003    BCC PH1
EFFC          4C88EA JMP $EA88
EFFF PH1      38      SEC
F000          60      RTS
```

```
F001          ;und Anbindung an KEPR
F001          ;Kleinbuchstaben werden angenommen
F001                *=$E970
E970          4C17F4 JMP KEPRH
E973                *=$F417
F417 KEPRH    20AFE7 JSR KEP
F41A          2073E9 JSR REDOUT
F41D          4C29E9 JMP FILT1
F420          ;------------------ 3.E ---------------------------------
F420          ;ASCII Eingabe bei allen 2-Byte Eingaben, d.h. bei
F420          ;<'>, <G>, <A>, <X>, <Y>, <P>, <S>-Kommandos
F420          ;mit 'ASCII-Zeichen eingeben
F420                *=$EA68
EA68          C927   CMP #''' ;teste auf "'"
EA6A                *=$EA6C
EA6C          4C73E9 JMP REDOUT       ;1 Zeichen lesen
EA6F          EA     NOP
EA70          ;Dazu Änderung in OUTDIS:
EA70                *=$EF75
EF75 OUTD7    20ACEB JSR PLXY
EF78          68     PLA
EF79          18     CLC              ;C=0 übergeben
EF7A          60     RTS
EF7B                *=$EF6A
EF6A          4C75EF JMP OUTD7        ;anpassen
EF6D                *=$FEAB
FEAB          4C75EF JMP OUTD7        ;anpassen
FEAE                *=$EF2D
EF2D          3046   BMI OUTD7
EF2F                *=$EF51

EF51          D022   BNE OUTD7
EF53                *=$EF58
EF58          B01B   BCS OUTD7
EF5A          ;------------------ 3.F ---------------------------------
EF5A          ;Druckeranbindung über PRIOUT Vektor
EF5A          ;Alle Zeichen die auf dem Display erscheinen werden auch
EF5A          ;über PRIOUT abgegeben. Zusätzlich sendet die Taste LF
EF5A          ;ein Linefeed ($0A), die Taste DEL ein Del ($7F). Die
EF5A          ;Dekodierung dieser Zeichen bleibt dem Benutzer überlassen
EF5A          ;Das Printerflag $A411 steht weiterhin zur Verfügung,
EF5A          ;CURPOS $A416 bleibt unbenutzt.
EF5A          ;über PRIOUT kann eine Videoroutine problemlos angesteuert
EF5A          ;werden.
EF5A          ;V O R S I C H T  !   PRIOUT nicht von Hand ändern !
EF5A          ;Änderung von OUTDP:
EF5A          ;statt JSR OUTPRI jetzt JSR TOPRI unter Benutzung
EF5A          ;des indirekten Sprungs JMP (PRIOUT) in WHERE0:
EF5A                *=$EEFC
EEFC OUTDP    2094E8 JSR TOPRI
EEFF          ;Dazu Änderungen in OUTALL:
EEFF                *=$E9DA
E9DA          2094E8 JSR TOPRI
E9DD                *=$EA3B
EA3B          2094E8 JSR TOPRI
EA3B          ;und in WHERO:
EA3B                *=$E894
E894 TOPRI    6C78A4 JMP (PRIOUT)     ;indirekter Sprung über PRIOUT-Vektor
E897          ;und Änderung in PSLS:
E897          ;Das Blank zum löschen eines Zeichens nach einem DEL wird
E897          ;nur noch direkt an das Display übergeben. Für PRIOUT
E897          ;wird ein $7F abgegeben (z.B. für Video).
E897                *=$E7F0
E7F0          2005EF JSR OUTDIS       ;Blank nur auf Display
E7F3                *=$E802
E802          A97F   LDA #$7F         ;Delete
E804          2094E8 JSR TOPRI        ;Del für Drucker
E807          20ACEB JSR PLXY
E80A          60     RTS
E80B          ;Änderung in CLR:
E80B                *=$EB49
EB49          EA     NOP
EB4A          EA     NOP
EB4B          EA     NOP
EB4C          ;PATCH5 entfällt:
EB4C                *=$FEB1
FEB1          ;im EPROM auf $FF gesetzt
```

```
FEB7          ;Taste LF sendet ein Linefeed über PRIOUT Vektor:
FEB7                *=$EC38
EC38          A90A  LDA #$0A          ;LF
EC3A          2094E8 JSR TOPRI
EC3D          EA    NOP
EC3E          EA    NOP
EC3F          EA    NOP
EC40          ;------------------ 3.G ---------------------------------
EC40          ;Deutscher Zeichensatz und deutsche Tastatur:
EC40          ;Durch den Wegfall des Thermodruckers ist die Funktion der
EC40          ;Taste PRINT überflüssig geworden. Über diese Taste wird
EC40          ;jetzt das ö eingegeben.
EC40          ;Tastaturbelegung: F1      $5B/$7B     Ä/ä
EC40          ;                  F2      $5D/$7D     Ü/ü
EC40          ;                  F3      $5E/$7E     ^/ß
EC40          ;                  PRINT   $5C/$7C     Ö/ö
EC40          ;Durch Abhebeln der Tastenkappen des Keyboards läßt sich
EC40          ;die Tastatur den deutschen Normen annähern:
EC40                *=$F443
F443          5A    .BYT $5A          ;alt: Y       neu: Z
F444                *=$F451
F451          59    .BYT $59          ;alt: Z       neu: Y
F452                *=$F42B
F42B          60    .BYT $60          ;alt: PRINT   neu: §
F42C                *=$F45F
F45F          2D    .BYT $2D          ;alt: F2      neu: -
F460                *=$F434
F434          5D    .BYT $5D          ;alt: -       neu: F2
F435                *=$F437
F437          5C    .BYT $5C          ;alt: ;       neu: PRINT
F438                *=$F42A
F42A          5B    .BYT $5B          ;alt: §       neu: F1
F42B                *=$F460
F460          3B    .BYT $3B          ;alt: F1      neu: ;
F461          ;------------------ 3.H ---------------------------------
F461          ;Floppy-Disk-Routinen
F461                *=$F001
F001          ;Talk senden, Geräteadresse im Accu
F001 SDTALK   8D75A4 STA LASTDV       ;Gerätenummer festhalten
F004          0940   ORA #$40         ;Talkeradresse Bit6=1
F006          D005   BNE CMDSD
F008          ;Listen senden, Geräteadresse im Accu
F008 SDLIST   8D75A4 STA LASTDV       ;Geräteadresse festhalten
F00B          0920   ORA #$20         ;Listeneradresse Bit5=1
F00D CMDSD    48     PHA
F00E          2C74A4 BIT IECFLG       ;MSB=1: noch 1 Zeichen gespeichert
F011          1014   BPL NOCHAR       ;nein, keins
F013          38     SEC
F014          A940   LDA #%01000000   ;Sendeflag setzen
F016          0D74A4 ORA IECFLG       ;in IECFLG
F019          8D74A4 STA IECFLG
F01C          204CF0 JSR SDIEC        ;auf IEC-Bus ausgeben
F01F          A93F   LDA #%00111111   ;Sendeflag und Zeichenflag auf 0
F021          2D74A4 AND IECFLG       ;in IECFLG
F024          8D74A4 STA IECFLG
F027 NOCHAR   68     PLA
F028          8D18A4 STA CBUF         ;Zeichen speichern
F02B          78     SEI
F02C          2067F2 JSR OUT1         ;sende '1'
F02F          C93F   CMP #$3F
F031          D003   BNE PUTATN
F033          2055F2 JSR TAAN         ;Takt ein
F036 PUTATN   AD00A0 LDA PORTB
F039          0908   ORA #$08         ;ATN setzen
F03B          8D00A0 STA PORTB
F03E SDMATN   78     SEI
F03F          205EF2 JSR TAAUS        ;Takt aus
F042          2067F2 JSR OUT1         ;sende '1'
F045 MS1      8A     TXA              ;eine Millisekunde warten
F046          A2B8   LDX #$B8
F048 MSL      CA     DEX
F049          D0FD   BNE MSL
F04B          AA     TAX
F04C
F04C          ;in CBUF gespeichertes Byte auf IEC-Bus ausgeben
F04C SDIEC    78     SEI
F04D          A900   LDA #0
F04F          8D17A4 STA STATUS       ;Status löschen
```

```
F052          2067F2 JSR OUT1         ;sende '1'
F055          20A4E3 JSR REACT        ;hole Port
F058          B05C   BCS SETSTN       ;Bit7=1, Gerät nicht da
F05A          2055F2 JSR TAAN         ;Takt ein
F05D          2C74A4 BIT IECFLG       ;noch Byte gespeichert ?
F060          500A   BVC NOSTO         ;nein
F062 I1       20A4E3 JSR REACT        ;hole Port
F065          90FB   BCC I1           ;warte auf Bit7=1
F067 I2       20A4E3 JSR REACT        ;hole Port
F06A          B0FB   BCS I2           ;warte bis Bit7=0
F06C NOSTO    20A4E3 JSR REACT        ;hole Port
F06F          90FB   BCC NOSTO        ;warte bis Bit7=1
F071          205EF2 JSR TAAUS        ;Takt aus
F074          A908   LDA #8           ;setze Bitzähler auf 8
F076          8D7EA4 STA BITCNT
F079 I3       AD00A0 LDA PORTB        ;hole Port B
F07C          CD00A0 CMP PORTB
F07F          D0F8   BNE I3           ;warte bis stabil
F081          0A     ASL A
F082          9036   BCC SETSTT       ;Bit7=0, setze Status 'time out'
F084          6E18A4 ROR CBUF         ;Zeichen: LSB in Carry
F087          B005   BCS I4           ;C=1
F089          2070F2 JSR OUT0         ;sende '0'
F08C          D003   BNE I5           ;immer
F08E I4       2067F2 JSR OUT1         ;sende '1'
F091 I5       2055F2 JSR TAAN         ;Takt ein
F094          AD00A0 LDA PORTB        ;sende '1'
F097          29DF   AND #$DF         ;Takt aus
F099          0910   ORA #$10
F09B          8D00A0 STA PORTB        ;in Port B
F09E          CE7EA4 DEC BITCNT       ;Zähler-1
F0A1          D0D6   BNE I3           ;noch nicht fertig
F0A3          A904   LDA #CONST1
F0A5          20D7F2 JSR TIMER        ;Timer starten
F0A8 I6       A920   LDA #%00100000   ;teste T2-Flag
F0AA          2C0DA0 BIT IFR
F0AD          D00B   BNE SETSTT       ;fertig
F0AF          20A4E3 JSR REACT        ;hole Port
F0B2          B0F4   BCS I6           ;Bit7=1, weiter testen
F0B4          58     CLI
F0B5          60     RTS
F0B6
F0B6          ;Status setzen
F0B6 SETSTN   A980   LDA #$80         ;device missing
F0B8          D002   BNE SST
F0BA SETSTT   A903   LDA #3           ;time out
F0BC SST      0D17A4 ORA STATUS       ;Status setzen
F0BF          8D17A4 STA STATUS
F0C2          58     CLI
F0C3          209CE5 JSR U3           ;ATN zurück, Takt ein und sende '1'
F0C6          4CEFF2 JMP BUSERR       ;Test auf Busfehler
F0C9
F0C9          ;Accu als Sekundäradresse nach Listen senden
F0C9 SNL      8D18A4 STA CBUF         ;Zeichen speichern
F0CC          203EF0 JSR SDMATN       ;mit ATN ausgeben
F0CF REATN    AD00A0 LDA PORTB        ;ATN zurücksetzen
F0D2          29F7   AND #$F7
F0D4          8D00A0 STA PORTB
F0D7          60     RTS
F0D8
F0D8          ;Accu als Sekundäradresse nach Talk senden
F0D8 SNT      8D18A4 STA CBUF         ;Zeichen speichern
F0DB          203EF0 JSR SDMATN       ;mit ATN ausgeben
F0DE          78     SEI
F0DF          2070F2 JSR OUT0         ;sende '0'
F0E2          20CFF0 JSR REATN        ;ATN zurücksetzen
F0E5          2055F2 JSR TAAN         ;Takt ein
F0E8 S1       20A4E3 JSR REACT        ;hole Port
F0EB          30FB   BMI S1           ;warte auf Bit6=0
F0ED          58     CLI
F0EE          60     RTS
F0EF
F0EF          ;ein Byte vom Accu auf IEC-Bus ausgeben
F0EF STAIEC   2C74A4 BIT IECFLG       ;Zeichen gespeichert ?
F0F2          3009   BMI B1           ;ja, sende es
F0F4          2E74A4 ROL IECFLG
F0F7          38     SEC              ;setze Flag, speichere neues Z.
```

```
F0F8          6E74A4 ROR IECFLG
F0FB          D005   BNE STBYT
F0FD B1       48     PHA
F0FE          204CF0 JSR SDIEC        ;sende gespeichertes Byte
F101          68     PLA
F102 STBYT    8D18A4 STA CBUF         ;speichere neues
F105          60     RTS
F106 BLK1
F106
F106                 *=$E587
E587          ;Untalk senden
E587 UNTALK   78     SEI
E588          205EF2 JSR TAAUS        ;Takt aus
E58B          AD00A0 LDA PORTB        ;ATN setzen
E58E          0908   ORA #8
E590          8D00A0 STA PORTB
E593          A95F   LDA #$5F         ;$5F=Untalk
E595          D002   BNE U0
E597          ;Unlisten senden
E597 UNLIS    A93F   LDA #$3F         ;$3F=Unlisten
E599 U0       200DF0 JSR CMDSD        ;auf IEC Bus ausgeben
E59C U3       20CFF0 JSR REATN        ;ATN zurücksetzen
E59F U2       8A     TXA              ;warte 40 Millisekunden
E5A0          A20A   LDX #$0A
E5A2 U1       CA     DEX
E5A3          D0FD   BNE U1
E5A5          AA     TAX
E5A6          2055F2 JSR TAAN         ;Takt ein
E5A9          4C67F2 JMP OUT1         ;sende '1'
E5AC BLK2
E5AC
E5AC                 *=BLK1
F106          ;ein Byte vom IEC-Bus in Accu holen
F106 LDAIEC   78     SEI
F107          A900   LDA #0
F109          8D17A4 STA STATUS       ;Status löschen
F10C          8D7EA4 STA BITCNT       ;Bitzähler auf 0
F10F          2055F2 JSR TAAN         ;Takt ein
F112 L1       20A4E3 JSR REACT        ;hole Port
F115          10FB   BPL L1           ;warte auf Bit 6=1
F117 L6       A902   LDA #CONST2
F119          20D7F2 JSR TIMER        ;Timer starten
F11C          2067F2 JSR OUT1         ;sende '1'
F11F L2       A920   LDA #%00100000   ;teste T2-Flag
F121          2C0DA0 BIT IFR
F124          D007   BNE L3           ;zu Ende
F126          20A4E3 JSR REACT        ;hole Port
F129          30F4   BMI L2           ;warte wenn Bit6=1
F12B          101D   BPL L4           ;weiter, Bit6=0
F12D L3       AD7EA4 LDA BITCNT       ;hole Bitzähler
F130          F005   BEQ L5           ;ist Null
F132          A902   LDA #2           ;setze Status
F134          4CBCF0 JMP SST

F137 L5       2070F2 JSR OUT0         ;sende '0'
F13A          2055F2 JSR TAAN         ;Takt ein
F13D          A940   LDA #$40         ;EOF
F13F          0D17A4 ORA STATUS       ;Status setzen
F142          8D17A4 STA STATUS
F145          EE7EA4 INC BITCNT       ;Bitzähler +1
F148          D0CD   BNE L6           ;nächster Zyklus
F14A L4       A908   LDA #8           ;setze Zähler für 8 Bits
F14C          8D7EA4 STA BITCNT
F14F L7       AD00A0 LDA PORTB        ;hole Port
F152          CD00A0 CMP PORTB
F155          D0F8   BNE L7           ;warte bis Signal stabil
F157          0A     ASL A
F158          10F5   BPL L7           ;warte auf Bit6=1
F15A          6E7DA4 ROR RESBYT       ;Resultat LSB einrotieren
F15D L8       AD00A0 LDA PORTB        ;hole Port
F160          CD00A0 CMP PORTB
F163          D0F8   BNE L8           ;warte bis Signal stabil
F165          0A     ASL A
F166          30F5   BMI L8           ;warte bis Bit6=0
F168          CE7EA4 DEC BITCNT       ;Zähler-1
F16B          D0E2   BNE L7           ;noch nicht zu Ende
F16D          2070F2 JSR OUT0         ;sende '0'
F170          2C17A4 BIT STATUS       ;hole Status
```

```
F173          5003    BVC GRES          ;kein EOF
F175          209FE5 JSR U2             ;ja Takt ein, sende '1'
F178 GRES     AD7DA4 LDA RESBYT         ;Resultat in Accu
F17B          58      CLI
F17C          60      RTS
F17D BLK3
F17D
F17D                  *=$F255
F255          ;Takt ein
F255 TAAN     AD00A0 LDA PORTB
F258          29EF    AND #%11101111
F25A          8D00A0 STA PORTB
F25D          60      RTS
F25E          ;Takt aus
F25E TAAUS    AD00A0 LDA PORTB
F261          0910    ORA #%00010000
F263          8D00A0 STA PORTB
F266          60      RTS
F267          ;Bit '1' ausgeben
F267 OUT1     AD00A0 LDA PORTB
F26A          29DF    AND #%11011111
F26C          8D00A0 STA PORTB
F26F          60      RTS
F270          ;Bit '0' ausgeben
F270 OUT0     AD00A0 LDA PORTB
F273          0920    ORA #%00100000
F275          8D00A0 STA PORTB
F278          60      RTS
F279 BLK4
F279
F279                  *=$E3A4
E3A4          ;warte auf stabiles Signal an den Portleitungen
E3A4          ;rotiere Input nach links
E3A4          ;Carry testet ext. Daten, N testet ext. Takt
E3A4 REACT    AD00A0 LDA PORTB
E3A7          CD00A0 CMP PORTB
E3AA          D0F8    BNE REACT
E3AC          0A      ASL A
E3AD          60      RTS
E3AE BLK5
E3AE
E3AE                  *=$F2D7
F2D7          ;Timer mit Accu als High-Byte starten
F2D7 TIMER    48      PHA               ;Accu retten
F2D8          AD0BA0 LDA ACR           ;setze T2 one shot
F2DB          29DF    AND #%11011111
F2DD          8D0BA0 STA ACR
F2E0          A97F    LDA #$7F
F2E2          8D0EA0 STA IER           ;keine Interrupts
F2E5          A900    LDA #0
F2E7          8D08A0 STA T2LL          ;setze T2 latch low
F2EA          68      PLA
F2EB          8D09A0 STA T2CH          ;starte Timer
F2EE          60      RTS
F2EF
F2EF          ;Busfehler: $42=hinter EOF führt zu normaler
F2EF          ;Beendigung, alle anderen Zustände werden mit der
F2EF          ;Gerätenummer ausgegeben; danach Rückkehr zum Monitor.
F2EF BUSERR   AD17A4 LDA STATUS        ;Status holen
F2F2          C942    CMP #$42          ;hinter EOF ?
F2F4          D001    BNE NOEOF         ;nein, Fehler
F2F6          60      RTS
F2F7 NOEOF    48      PHA
F2F8          20FEE8 JSR LL            ;In=Tastatur, OUT=Display
F2FB          2013EA JSR CRLOW         ;CRLF
F2FE          A000    LDY #DEVMSG-UTEXT ;' DEVICE '
F300          20D8E3 JSR KEPU          ;ausgeben
F303          AD75A4 LDA LASTDV        ;letztes Gerät
F306          C90A    CMP #10           ;>10 ?
F308          9003    BCC FNR           ;nein
F30A          18      CLC
F30B          6906    ADC #6            ;ja, +6 für Dezimalzahl
F30D FNR      2046EA JSR NUMA          ;ausgeben
F310          203EE8 JSR BLANK         ;Leerzeichen
F313          8D74A4 STA IECFLG        ;IECFLG wieder auf $20
F316          68      PLA
F317          1004    BPL TMOUTS        ;Status>0: nur timeout's
```

```
F319          A008    LDY #NODEV-UTEXT  ;' MISSING'
F31B          D009    BNE ANZEI         ;ausgeben
F31D TMOUTS   4A      LSR A
F31E          9004    BCC RTO           ;ist read timeout
F320          A00F    LDY #WTMOUT-UTEXT  ;' W-TIMEOUT'
F322          D002    BNE ANZEI
F324 RTO      A018    LDY #RTMOUT-UTEXT  ;' R-TIMEOUT'
F326 ANZEI    20D8E3 JSR KEPU           ;anzeigen
F329          4CA1E1 JMP COMIN          ;zurück zum Monitor
F32C
F32C          ;File auf IEC-Bus eröffnen
F32C          ;Gerätenummer im Accu, Kanal (=Sekundäradresse) in Y
F32C          ;bei Carry=0 wird ein Befehlsstring übertragen
F32C FOPEN    08      PHP               ;Flags retten (Carry)
F32D          2008F0 JSR SDLIST        ;Listen(Gerätenummer)
F330          98      TYA
F331          09F0    ORA #$F0
F333          20C9F0 JSR SNL           ;Sekundäradresse nach Listen
F336          28      PLP               ;Flags holen
F337          B003    BCS ENDOP         ;C=1, kein String
F339          20C5FF JSR TPUT          ;Befehlsstring ausgeben
F33C ENDOP    4C97E5 JMP UNLIS         ;Unlisten ausgeben
F33F BLK8
F33F
F33F                  *=BLK3
F17D          ;File auf IEC-Bus schließen
F17D          ;Gerätenummer im Accu, Sekundäradresse in Y
F17D FCLOSE   2008F0 JSR SDLIST        ;Listen(Gerätenummer)
F180          98      TYA
F181          29EF    AND #$EF
F183          09E0    ORA #$E0
F185          20C9F0 JSR SNL           ;Sekundäradresse nach Listen
F188          4C97E5 JMP UNLIS         ;Unlisten
F18B
F18B                  *=$FFF1
FFF1          ;Gerät zum Zuhören auffordern
FFF1          ;Gerätenummer im Accu, Sekundäradresse in Y
FFF1 DOLIST   2008F0 JSR SDLIST        ;Listen(Gerätenummer)
FFF4          98      TYA
FFF5          0960    ORA #$60
FFF7          4CC9F0 JMP SNL           ;Sekundäradresse nach Listen
FFFA
FFFA                  *=$FEEF
FEEF          ;Gerät zum Senden auffordern
FEEF          ;Gerätenummer im Accu, Sekundäradresse in Y
FEEF DOTALK   2001F0 JSR SDTALK        ;Talk(Gerätenummer)
FEF2          98      TYA
FEF3          0960    ORA #$60
FEF5          4CD8F0 JMP SNT           ;Sekundäradresse nach Talk
FEF8
FEF8                  *=BLK8
F33F          ;Kommandokanal (Kanal 15) des letzten aktiven Geräts
F33F          ;lesen, Fehlernummer, Spur- und Sektornummer
F33F          ;einlesen.
F33F          ;bei Textflag=1 wird die Gerätenachricht angezeigt.
F33F          ;Carry=1 bei Fehler >= $14
F33F MES15    AD75A4 LDA LASTDV        ;letztes Gerät
F342          A00F    LDY #15           ;Kanal 15
F344          38      SEC
F345          202CF3 JSR FOPEN         ;Kommandokanal öffnen
F348          AD75A4 LDA LASTDV        ;letztes Gerät
F34B          A00F    LDY #15           ;Kommandokanal
F34D          20EFFE JSR DOTALK        ;zum Senden auffordern
F350          209BF3 JSR NUMRD         ;Fehlernummer einlesen
F353          48      PHA
F354          8D7AA4 STA FEHLER        ;in Fehler
F357          2006F1 JSR LDAIEC        ;Komma überlesen
F35A          AD74A4 LDA IECFLG        ;Textflag testen
F35D          2920    AND #%00100000
F35F          F002    BEQ STOIMM       ;ist Null, kein Text
F361          A9FF    LDA #$FF
F363 STOIMM   85A7    STA LAE
F365          F006    BEQ BTEXT        ;Display lassen
F367          2013EA JSR CRLOW         ;CRLF
F36A          203EE8 JSR BLANK         ;Leerzeichen
F36D BTEXT    2006F1 JSR LDAIEC        ;Zeichen holen
F370          C92C    CMP #','          ;bis Komma
F372          F009    BEQ MESRST       ;behandle Rest
```

```
F374          25A7    AND LAE
F376          F0F5    BEQ BTEXT       ;wenn 0, nicht anzeigen
F378          207AE9  JSR OUTPUT      ;auf Display
F37B          D0F0    BNE BTEXT       ;weiter
F37D MESRST   209BF3  JSR NUMRD       ;Spur einlesen
F380          8D7BA4  STA SPUR
F383          2006F1  JSR LDAIEC      ;Komma überlesen
F386          209BF3  JSR NUMRD       ;Sektor einlesen
F389          8D7CA4  STA SEKTOR
F38C BISEOF   2006F1  JSR LDAIEC
F38F          AD17A4  LDA STATUS      ;bis Status=EOF weiterlesen
F392          F0F8    BEQ BISEOF
F394          2087E5  JSR UNTALK      ;Bus freigeben
F397          68      PLA             ;Fehlernummer holen
F398          C914    CMP #20         ;C=0 bei Fehler < 20
F39A          60      RTS
F39B
F39B          ;eine Dezimalzahl aus zwei ASCII-Zeichen bestehend vom Bus
F39B          ;einlesen und als Hex-Zahl im Accu abliefern.
F39B NUMRD    D8      CLD             ;kein Dezimalmodus
F39C          2006F1  JSR LDAIEC      ;hole Zeichen
F39F          290F    AND #$0F        ;Bits 4-7 auf 0
F3A1          8D7DA4  STA RESBYT
F3A4          0A      ASL A           ;*4
F3A5          0A      ASL A
F3A6          18      CLC
F3A7          6D7DA4  ADC RESBYT      ;+RESBYT: *5
F3AA          0A      ASL A           ;*2
F3AB          A8      TAY             ;in Y retten
F3AC          2006F1  JSR LDAIEC      ;nächste Ziffer
F3AF          290F    AND #$0F        ;Bits 0 bis 3 abtrennen
F3B1          8D7DA4  STA RESBYT
F3B4          98      TYA
F3B5          18      CLC
F3B6          6D7DA4  ADC RESBYT      ;erste Zahl addieren
F3B9          60      RTS
F3BA BLK9
F3BA
F3BA                  *=$EBDB
EBDB          ;Text von der Tastatur einlesen.
EBDB          ;Displaybuffer dient als Pufferspeicher
EBDB          ;Textstart und Länge werden in TXT und LAE
EBDB          ;übergeben.
EBDB TGET     AD15A4  LDA CURPO2      ;Start des Textes
EBDE          85A7    STA LAE         ;retten
EBE0          A000    LDY #0          ;Y auf 0 für RDRUB
EBE2 CHGET    205FE9  JSR RDRUB       ;Tastatur lesen
EBE5          C90D    CMP #CR         ;beende bei CR
EBE7          F00A    BEQ EOL
EBE9          AD15A4  LDA CURPO2
EBEC          C93C    CMP #60         ;60 Zeichen ?
EBEE          B0F2    BCS CHGET       ;ja, Y nicht erhöhen
EBF0          C8      INY
EBF1          D0EF    BNE CHGET
EBF3 EOL      AD15A4  LDA CURPO2      ;CURPO2=60 ?
EBF6          C93C    CMP #60
EBF8          D001    BNE NORMAL      ;nein
EBFA          C8      INY             ;Y um 1 erhöhen
EBFB NORMAL   A938    LDA #<DIBUFF    ;addiere Textstart
EBFD          18      CLC
EBFE          65A7    ADC LAE         ;zu $A438 (=DIBUFF)
EC00          85A8    STA TXT         ;in Textpointer
EC02          A9A4    LDA #>DIBUFF
EC04          85A9    STA TXT+1
EC06          84A7    STY LAE         ;Länge übergeben
EC08          98      TYA             ;N=1 bei Länge = 0
EC09          60      RTS
EC0A
EC0A                  *=$FFC5
FFC5          ;Text aus Buffer auf IEC-Bus ausgeben.
FFC5          ;Textstart muß in TXT stehen, Textlänge
FFC5          ;(max. 256) in LAE.
FFC5          ;IEC-File muß als Listener aktiv sein
FFC5 TPUT     A000    LDY #0          ;Zeiger auf 0
FFC7          C4A7    CPY LAE         ;keine Leerstrings
FFC9          F00E    BEQ PUTEND
```

```
FFCB PUTNXT B1A8    LDA (TXT),Y      ;hole Zeichen
FFCD        C90D    CMP #CR          ;bei CR beenden
FFCF        F008    BEQ PUTEND
FFD1        20EFF0  JSR STAIEC       ;Zeichen auf IEC-Bus
FFD4        C8      INY
FFD5        C6A7    DEC LAE          ;Länge-1
FFD7        D0F2    BNE PUTNXT       ;Schleife
FFD9 PUTEND 60      RTS
FFDA BLK11
FFDA
FFDA                *=BLK9
F3BA        ;Filenamen mit TGET holen, File eröffnen.
F3BA        ;X=0: Input File, X=1: Output File.
F3BA FFNAM  A04D    LDY #TMSGM1      ;'F=' anzeigen
F3BC        20AFE7  JSR KEP
F3BF        20DBEB  JSR TGET         ;Filenamen holen
F3C2 ALLF   D007    BNE FF1          ;mit Filename
F3C4        A92A    LDA #'*'         ;sonst Filename "*"
F3C6        207AE9  JSR OUTPUT       ;anzeigen
F3C9        E6A7    INC LAE          ;Stringlänge = 1
F3CB FF1    BD76A4  LDA INDEV,X      ;Gerät und Kanal
F3CE        204EFD  JSR UPDEV        ;entpacken
F3D1        18      CLC
F3D2        202CF3  JSR FOPEN        ;File eröffnen
F3D5        203FF3  JSR MES15        ;ok ?
F3D8        9003    BCC FLOD         ;ja, Bits setzen
F3DA        4CA1E1  JMP COMIN        ;zurück zum Monitor
F3DD FLOD   E8      INX              ;X=00000001 oder X=00000010
F3DE        8A      TXA              ;in ACCU
F3DF        0D74A4  ORA IECFLG       ;File als geöffnet markieren
F3E2        8D74A4  STA IECFLG
F3E5        CA      DEX
F3E6        4CBEF2  JMP OPINO        ;File vorbereiten
F3E9 BLK10
F3E9
F3E9                *=$F2BE
F2BE        ;File vorbereiten: Für Output File TAPTR auf 0
F2BE        ;für Input File ersten Block einlesen
F2BE OPINO  F005    BEQ OPIN         ;X=0: Input File
F2C0        CA      DEX              ;X auf 0
F2C1        8E37A4  STX TAPTR2       ;Bufferpointer auf 0
F2C4        60      RTS
F2C5 OPIN   4CB6E5  JMP INF          ;Block einlesen
F2C8 MK1
F2C8
F2C8        ;Kommando auf Floppy Kanal 15 ausgeben
F2C8        ;Gerätenummer wird abgefragt
F2C8        ;CR oder Space: dann letztes angesprochenes Gerät
F2C8                *=$EEA8
EEA8 FCMD   2013EA  JSR CRLOW        ;Display löschen
EEAB        A000    LDY #DEVMSG-UTEXT  ;'DEVICE '?
EEAD        20D8E3  JSR KEPU         ;anzeigen
EEB0        2085E7  JSR GCNT         ;Nummer holen
EEB3        B00E    BCS ISCMD        ;C=1, war CR oder Space
EEB5        C916    CMP #$16         ;>=$16 ?
EEB7        B0EF    BCS FCMD         ;zu groß
EEB9        C90A    CMP #$A
EEBB        9003    BCC PUTNR        ;<10, ok
EEBD        38      SEC
EEBE        E906    SBC #6           ;sonst 6 abziehen
EEC0 PUTNR  8D75A4  STA LASTDV       ;in LASTDV
EEC3 ISCMD  2013EA  JSR CRLOW        ;CRLF
EEC6        A021    LDY #CMDMSG-UTEXT  ;'CMD:' anzeigen
EEC8        20D8E3  JSR KEPU
EECB        20DBEB  JSR TGET         ;Befehl holen
EECE        F009    BEQ FC1          ;nur Meldung holen
EED0        AD75A4  LDA LASTDV       ;File auf Kanal 15 öffnen
EED3        A00F    LDY #15
EED5        18      CLC              ;mit String
EED6        4C2CF3  JMP FOPEN
EED9 FC1    4C3FF3  JMP MES15        ;Meldung holen
EEDC
EEDC                *=$FD4E
FD4E        ;Ein Byte, bestehend aus Gerätenummer und
FD4E        ;Sekundäradresse entpacken.
FD4E        ;Sekundäradresse in Y, Gerätenummer in A
FD4E UPDEV  48      PHA              ;retten
FD4F        290F    AND #$0F         ;Sekundäradresse
```

```
FD51          A8      TAY             ;in Y
FD52          68      PLA
FD53          4A      LSR A           ;Gerätenummer ins rechte Halbbyte
FD54          4A      LSR A
FD55          4A      LSR A
FD56          4A      LSR A
FD57          60      RTS

FD58                  *=BLK4
F279          ;File auf IEC-Bus schließen
F279          ;X=0: Input File, X=1: Output File
F279 CLSFIL   E8      INX             ;X=1 ODER X=2
F27A          8A      TXA             ;in Accu
F27B          CA      DEX
F27C          49FF    EOR #$FF        ;Komplement bilden
F27E          2D74A4  AND IECFLG      ;Bit löschen
F281          8D74A4  STA IECFLG
F284          BD76A4  LDA INDEV,X     ;Gerät und Kanal
F287          4CC8F2  JMP CLINO       ;zum Fileschließen
F28A          EA      NOP
F28B
F28B                  *=MK1
F2C8          ;File schließen
F2C8 CLINO    204EFD  JSR UPDEV       ;entpacken
F2CB          207DF1  JSR FCLOSE      ;File schließen
F2CE          203FF3  JSR MES15       ;Meldung holen
F2D1          9003    BCC CLN         ;C=0, dann ok
F2D3          4CA1E1  JMP COMIN       ;sonst zum Monitor
F2D6 CLN      60      RTS
F2D7
F2D7                  *=BLK5
E3AE          ;Input File schließen
E3AE          ;falls nicht 'F, dann weiter mit DU13
E3AE CLF      AD12A4  LDA INFLG       ;Input Flag
E3B1 CLFP     C946    CMP #'F'        ;Floppy ?
E3B3          F003    BEQ ISTF        ;ja
E3B5 CLF1     4C20E5  JMP DU13
E3B8 ISTF     A200    LDX #0          ;0 für Input File
E3BA          2079F2  JSR CLSFIL      ;schließen
E3BD          90F6    BCC CLF1
E3BF BLK6
E3BF
E3BF                  *=$EE40
EE40          ;Output File schließen
EE40          ;falls nicht 'F, dann weiter mit DU11+3
EE40 CDF      AD13A4  LDA OUTFLG      ;Outflag
EE43          C946    CMP #'F'        ;Floppy ?
EE45          F003    BEQ ISDF        ;ja
EE47          4C0DE5  JMP $E50D       ;teste andere
EE4A ISDF     AD37A4  LDA TAPTR2      ;Bufferpointer = 0 ?
EE4D          C900    CMP #0
EE4F          F008    BEQ CLSDFE      ;ja, kein Rest im Buffer
EE51          A900    LDA #0          ;sonst mit Nullen füllen
EE53          208BF1  JSR TOBYTE
EE56          4C4AEE  JMP ISDF
EE59 CLSDFE   A201    LDX #1          ;1 für Dump File
EE5B          2079F2  JSR CLSFIL      ;schließen
EE5E          4C20E5  JMP DU13
EE61 BLK13
EE61
EE61                  *=$E80B
E80B          ;alle IEC-Monitorfiles schließen
E80B CLSALL   AD74A4  LDA IECFLG      ;teste Bits 0 und 1
E80E          4A      LSR A           ;Bit 0 auf 1 ?
E80F          9005    BCC CLS1        ;kein Input File
E811          48      PHA             ;rette Accu
E812          20B8E3  JSR ISTF        ;schließe Input File
E815          68      PLA
E816 CLS1     4A      LSR A           ;BIT 1 auf 1 ?
E817          9012    BCC CLS2        ;kein Output file
E819          A946    LDA #'F'
E81B          8D13A4  STA OUTFLG      ;setze OUTFLG auf Floppy
E81E          A90D    LDA #CR         ;noch 2 CR ausgeben
E820          208BF1  JSR TOBYTE
E823          A90D    LDA #CR
E825          208BF1  JSR TOBYTE
E828          204AEE  JSR ISDF        ;Output File schließen
E82B CLS2     60      RTS
```

```
E82C                    *=BLK6
E3BF            ;Input File nach letztem Block immer schließen
E3BF ENDTST AD17A4 LDA STATUS
E3C2        F011    BEQ ENDT1           ;EOF ?
E3C4 ENDTA  AD74A4 LDA IECFLG
E3C7        48      PHA                 ;IECFLG retten
E3C8        29DF    AND #%11011111      ;keine Meldung
E3CA        8D74A4 STA IECFLG
E3CD        2079F2 JSR CLSFIL           ;File schließen
E3D0        68      PLA
E3D1        8D74A4 STA IECFLG           ;Flag wiederherstellen
E3D4        60      RTS

E3D5 ENDT1  4C87E5 JMP UNTALK           ;Bus frei
E3D8
E3D8            ;Text auf Display ausgeben.
E3D8            ;Y dient als Zeiger
E3D8 KEPU   B9BCFE LDA UTEXT,Y
E3DB        48      PHA
E3DC        297F    AND #$7F
E3DE        207AE9 JSR OUTPUT           ;anzeigen
E3E1        C8      INY
E3E2        68      PLA
E3E3        10F3    BPL KEPU            ;weiter, wenn Bit 7=0 war
E3E5        60      RTS                 ;sonst zurück
E3E6 BLK7
E3E6
E3E6                    *=$FEBC
FEBC            ;Texte für KEPU:
FEBC UTEXT
FEBC DEVMSG 2044    .BYT ' DEVICE',$A0
FEC3        A0
FEC4 NODEV  4D49    .BYT 'MISSIN',$C7
FECA        C7
FECB WTMOUT 572D    .BYT 'W-TIMEOU',$D4
FED3        D4
FED4 RTMOUT 522D    .BYT 'R-TIMEOU',$D4
FEDC        D4
FEDD CMDMSG 434D44 .BYT 'CMD',$BA
FEE0        BA
FEE1
FEE1            ;Anbindung der Floppy an TOBYTE
FEE1                    *=$F19C
F19C        4CE9F3 JMP OUTTOF
F19F                    *=BLK10
F3E9            ;Ergänzung zu TOBYTE
F3E9            ;Einen Block (80 Zeichen) bei OUTFLG=F auf Floppy schreiben
F3E9 OUTTOF AD13A4 LDA OUTFLG           ;'T' oder 'F'
F3EC        C954    CMP #'T'
F3EE        D006    BNE NOTTAP          ;ist Floppy
F3F0        20E7F1 JSR BKCKSM           ;weiter mit alter Routine
F3F3        4C9FF1 JMP $F19F
F3F6 NOTTAP AD77A4 LDA OUTDEV           ;Output File öffnen
F3F9        204EFD JSR UPDEV
F3FC        20F1FF JSR DOLIST
F3FF        A200    LDX #0
F401 ONOCH  20D2F1 JSR CKBUFF           ;1 Byte aus Buffer
F404        20EFF0 JSR STAIEC           ;auf Floppy schreiben
F407        E8      INX
F408        E050    CPX #80
F40A        D0F5    BNE ONOCH           ;bis Bufferende
F40C        A900    LDA #0
F40E        8D37A4 STA TAPTR2           ;Bufferpointer auf 0
F411        2097E5 JSR UNLIS            ;Bus frei
F414        4CCEF1 JMP TABY3            ;TOBYTE beenden
F417
F417            ;Anbindung der Floppy an TIBYTE
F417                    *=$ED45
ED45        20ACE5 JSR INTOF
ED48                    *=BLK2
E5AC            ;Ergänzung zu TIBYTE
E5AC            ;Einen Block (80 Zeichen) bei INFLG=F von der Floppy lesen
E5AC INTOF  AD12A4 LDA INFLG            ;'T' oder 'F'
E5AF        C954    CMP #'T'
E5B1        D003    BNE INF             ;ist Floppy
E5B3        4C53ED JMP TIBY1            ;einen Block von Tape laden
```

```
E5B6 INF     AD76A4 LDA INDEV        ;Input File öffnen
E5B9         204EFD JSR UPDEV
E5BC         20EFFE JSR DOTALK
E5BF         A200   LDX #0
E5C1 INOCH   2006F1 JSR LDAIEC       ;1 Byte einlesen
E5C4         9D1601 STA TABUFF,X     ;in Buffer
E5C7         E8     INX
E5C8         E050   CPX #80
E5CA         D0F5   BNE INOCH        ;bis Buffer voll
E5CC         A200   LDX #0
E5CE         8E36A4 STX TAPTR        ;Bufferpointer auf 0
E5D1         4CBFE3 JMP ENDTST       ;letzter Block ?
E5D4
E5D4         ;Teste auf aktiven Buffer. C=0 wenn nur 1 Buffer benutzt
E5D4         ;wird. Bei 2 Buffern ist C=1. 2 Buffer bei INFLG,OUTFLG =
E5D4         ;T,T  T,F  F,T  F,F.
E5D4               *=$F1D2
F1D2 CKBUFF  AD12A4 LDA INFLG        ;'T' oder 'F'
F1D5         C954   CMP #'T'
F1D7         4CDAFF JMP ECKBUF       ;zur Behandlung
F1DA               *=BLK11
FFDA         ;Ergänzung zu CKBUFF
-------------------------------------------------------------------------
FFDA ECKBUF  F007   BEQ TSTOUT       ;falls 'T'
FFDC         C946   CMP #'F'
FFDE         F003   BEQ TSTOUT       ;falls 'F'
FFE0 CB1     4CE2F1 JMP CBUFF1       ;nur 1 Buffer
FFE3 TSTOUT  AD13A4 LDA OUTFLG       ;auch OUTFLG testen
FFE6         C954   CMP #'T'
FFE8         F004   BEQ CB2          ;T,T oder F,T
FFEA         C946   CMP #'F'
FFEC         D0F2   BNE CB1          ;nicht T oder F
FFEE CB2     4CDEF1 JMP CBUFF2       ;2 Buffer
FFF1
FFF1         ;Bei OUTFLG= 'F' oder 'T' Byte nicht als 2 ASCII ausgeben
FFF1               *=$E53C
E53C         4C23EC JMP EOUTCK       ;teste auf T oder F
E53F               *=$EC23
EC23         ;Ergänzung zu OUTCK1: 1 Byte als 2 ASCII ausgeben
EC23         ;oder bei T, F als 1 Hex
EC23 EOUTCK  AD13A4 LDA OUTFLG       ;output device
EC26         C946   CMP #'F'         ;Floppy ?
EC28         D004   BNE ONORM        ;nein
EC2A         68     PLA
EC2B         4CBCE9 JMP OUTALL       ;Zeichen ausgeben
EC2E ONORM   4C3FE5 JMP $E53F        ;teste T
EC31 BLK12
EC31
EC31         ;Bei 'T','F' ein Byte als Hex, sonst als 2 ASCII einlesen
EC31         ;Bei Floppy: Abbrechen mit Taste ESC,Pause mit Taste Space
EC31               *=$E3FD
E3FD RBYTE   4C25E4 JMP ERBYTE       ;teste auf T oder F
E400               *=$E425
E425         ;Ergänzung zu RBYTE:
E425 ERBYTE  AD12A4 LDA INFLG        ;Input Device
E428         C946   CMP #'F'         ;Floppy ?
E42A         D00C   BNE RNORM        ;nicht 'F'
E42C         209EEB JSR PHXY         ;X,Y retten
E42F         2007E9 JSR RCHEK        ;ESC, Space ?
E432         20ACEB JSR PLXY         ;X,Y holen
E435         4C93E9 JMP INALL        ;ja, nur Hex
E438 RNORM   4C00E4 JMP $E400        ;weiter
E43B
E43B         ;Floppy-Loadfile bei Checksum-Error schließen
E43B               *=$E385
E385         2061EE JSR ECKERR       ;Loadfile schließen
E388               *=BLK13
EE61         ;Ergänzung zu CKERR
EE61 ECKERR  20AEE3 JSR CLF          ;File schließen bei Floppy
EE64         4C8EE3 JMP CKER0        ;weiter wie normal
EE67
EE67         ;Input File schließen bei Editor Input
EE67               *=$FF03
FF03         AD12A4 LDA INFLG        ;Input Flag
FF06         4CE6E3 JMP EDICLO       ;schließe bei 'T' oder 'F'
FF09               *=BLK7
E3E6 EDICLO  C954   CMP #'T'         ;Tape ?
E3E8         D003   BNE EDI1         ;nein
```

```
E3EA          4C29E5 JMP $E529        ;ja, DU14
E3ED EDI1     C946   CMP #'F'         ;Floppy ?
E3EF          D003   BNE EDI2
E3F1          4CB8E3 JMP ISTF         ;Input File schließen
E3F4 EDI2     60     RTS
E3F5
E3F5          ;Alle Floppyfiles bei Editor-Error (z.B. bei Bufferende)
E3F5          ;schließen
E3F5                 *=$FA6F
FA6F          4C31EC JMP ERR01
FA72                 *=BLK12
EC31 ERR01    200BE8 JSR CLSALL       ;Input- und Output File schließen
EC34          4C78FA JMP ERR0
EC37
EC37          ;Floppyanbindung in WHEREI:
EC37                 *=$E85C
E85C WHE1     C946   CMP #'F'         ;Floppy ?
E85E          D008   BNE WHE2
E860          A200   LDX #0           ;Input File
E862          4CBAF3 JMP FFNAM        ;File eröffnen
E865          EA     NOP
E866          EA     NOP
E867          EA     NOP
E868 WHE2
E868
E868          ;Floppyanbindung in WHEREO:
E868                 *=$E885
E885 WHR01    C946   CMP #'F'         ;Floppy ?
E887          D005   BNE WHR02
E889          A201   LDX #1           ;Output File
E88B          4CBAF3 JMP FFNAM        ;File eröffnen
E88E WHR02
E88E
E88E          ;'F' statt 'K' in INALL
E88E                 *=$E99D
E99D          C946   CMP #'F'         ;Floppy ?
E99F          D003   BNE *+5
E9A1          4C3BED JMP TIBYTE       ;Byte vom IEC-Bus lesen
E9A4
E9A4          ;'F' statt 'K' in OUTALL
E9A4                 *=$E9C8
E9C8 OUTA1    C946   CMP #'F'         ;Floppy
E9CA          D004   BNE OUTA2
E9CC          68     PLA
E9CD          4C8BF1 JMP TOBYTE       ;Byte auf IEC-Bus schreiben
E9D0 OUTA2
E9D0
E9D0          ;Loadfile bei <L>-Kommando schließen
E9D0                 *=$E32C
E32C          4CAEE3 JMP CLF          ;Loadfile schließen
E32F
E32F          ;Dumpfile bei <D>-Kommando schließen
E32F                 *=$E50A
E50A          4C40EE JMP CDF          ;Dumpfile schließen
E50D
E50D          ;immer mit DU11 beenden
E50D                 *=$FF12
FF12          EA     NOP
FF13          EA     NOP
FF14
FF14          ;Bei Taste ESC alle Floppyfiles schließen
FF14                 *=$FF48
FF48          200BE8 JSR CLSALL       ;alle IEC-Files schließen
FF4B          20FEE8 JSR LL
FF4E
FF4E          ;Input File bei <3>-Kommando und Objectfile schließen
FF4E                 *=$FF6F
FF6F          4CAEE3 JMP CLF
FF72
FF72          ;Input File bei <3>-Kommando und Sourcefile schließen
FF72                 *=$E6BA
E6BA          4CAEE3 JMP CLF
E6BD
E6BD
E6BD                 .END
ERRORS=0000
```

Roland Löhr

IEC: Die seriellen Busroutinen

Übersicht

Für den Betrieb der seriell anzuschließenden CBM Floppy 1541 enthält dieses Heft mit DISKMON 1.0 ein vollständiges Betriebssystem für den AIM 65. Die nachfolgenden Ausführungen sollen Erklärungen und Alternativen bringen; vor allem für die Betreiber anderer Computer und auch für solche mit AIM, die sich vielleicht aus Platzgründen die Arbeit machen wollen, mit kürzerem Code auszukommen. Die Floppy wurde nach Mitteilung eines Lesers Mitte Januar zu etwa DM 600,- verkauft. Bei diesem oder ähnlichem Preis lohnt der Eigenbau kaum noch. Man muß für ihn das Laufwerk, das Netzteil und das Gehäuse kalkulieren, ferner die Mühe der programm-mäßigen Einbindung. Commodore Floppies haben für den Betreiber zudem den Vorteil der eingebauten Intelligenz für alle Verwaltungsaufgaben.

Für die nachfolgenden Routinen ist eine identische Beschaltung wie beim DISKMON 1.0 vorzunehmen. Sie wurden hier beim Herausgeber entworfen und getestet und zur Kontrolle auch bei Herrn Martin Albrecht erprobt. Sie ermöglichen einen störungsfreien Betrieb. Der hier häufig benutzte BIT-Befehl zur Erkennung von Signalwechseln könnte aber gelegentlich weniger sicher sein als seine Routine REACT.

Es ist darauf aufmerksam zu machchen, daß die folgenden Routinen kein Betriebsssystem bilden, bei dem z.B. der Filename und die Art der Aufzeichnung (PRG, SEQ usw. oder READ/WRITE) sowie der Floppy-Kanal interaktiv abgefragt würden. Es wurde auch noch keine Reaktion für ein mögliches Timeout vorgesehen (man strandet dann auf BREAKs).

Die serielle Schnittstelle

Zunächst einige Erklärungen zur Schnittstelle und zum Datenverkehr Rechner/Peripheriegerät. Wenn man einmal von der notwendigen Leitung GND (=Masse) und vom IFC (Interface Clear = Reset, optionale Leitung) absieht, dann bildet der serielle IEC-Bus eine 3-Draht-Verbindung. Die Leitung ATN hat die einzige Aufgabe, mit ATN=Low oder True Befehlsbytes von Daten- oder Programmbytes zu unterscheiden. Sie wird also für Primäradressen, Sekundäradressen sowie für UNLISTEN und UNTALK auf Low gezogen. Mit ATN=High werden bei der Eröffnung einer Datei die Bytes des Filename gesendet. Ebenso erfolgt der Transport jeder Art von Programm- oder Datenbytes. ATN darf nur vom Rechner (Master auf dem Bus) gezogen werden, nicht von den Peripherieeinheiten.

Es bleiben damit die Leitungen CLOCK (=Takt, s. S. 10 in diesem Heft) und DATA (=Daten) für den Transport aller Information und für jede Art von Handshake zwischen Rechner und Peripherie. Auf der Datenleitung wird bit-seriell gesendet, und zwar das niederwertigste Bit zuerst (siehe bei Label L9 im Listing). Low bedeutet: Bit=0 und umgekehrt. Da wir jedoch beim Senden mit Invertern auf den Bus gehen, erzeugen wir das inverse (entgegengesetzte) Signal an den Pins der VIA.

Empfangsseitig wird ein Bit durch die steigende Flanke des Signales CLOCK 'eingeclockt', siehe bei Label GB2. Für das einzelne Bit erfolgt kein Handshake. Innerhalb eines Bytes kann daher so schnell (oder nur so schnell) gesendet werden, wie die Peripherie einclocken kann. Es ist damit denkbar, daß die zur Datenkonsolidierung und für die Schaffung von Reaktionsspielräumen vorgesehene Routine WAIT zeitlich noch abgekürzt werden könnte. Während der Datenübermittlung beherrrscht der jeweilige Talker (Sender) beide Leitungen CLOCK und DATA. Das ändert sich an den Schnittstellen zwischen den übermittelten Bytes.

Handshakes

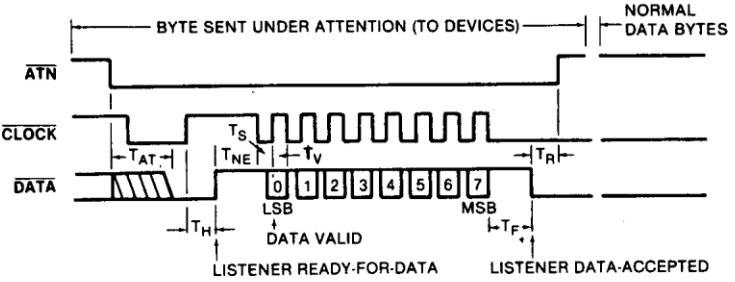
Handshakes werden bei folgenden Gelegenheiten ausgetauscht:

- Nach dem ATN=True des Masters

Als Frame Handshake nach dem Senden eines vollständigen Bytes
Als EOI-Acknowledge
Beim 'Turnaround', wenn der Master die Peripherie zum TALK aufgefordert hat und diese ein Acknowledge für die Rollenumkehr sendet.

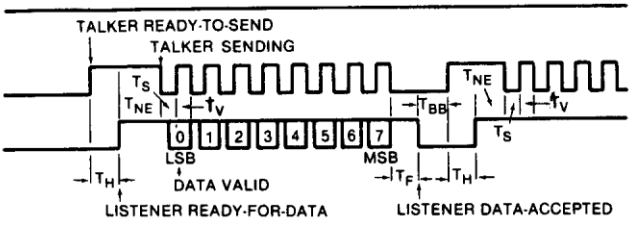
ATN-Handshake

Wenn der Master ATN=True setzt und wenig später CLOCK=True, so erwartet er, daß die Peripherie nach spätestens 1 ms DATA=True zieht als Handshake 'Ready for Data' (bei Label BYTOUT). Ist das der Fall (Label L5), so nimmt er CLOCK=False zurück. Kurz danach nimmt auch die Peripherie ihr Signal auf der Datenleitung zu DATA=High zurück. Der Master beginnt wenig später mit der Aussendung des ersten Datenbits.



Frame Handshake

Kurz nach dem Einclocken des letzten (höchstwertigen) Datenbits setzt der Talker sein CLOCK-Signal =True (Low) und wartet nun auf das Handshake der Peripherie mit DATA=True, das typisch nach 20 Mikro-Sek. und spätestens nach 1 ms kommen soll. Wenn nicht, dann Timeout (bei Label LW). Wenn der Talker das Handshake erkennt, setzt er zu ihm passender Zeit das CLOCK-Signal wieder auf High.



SERIAL BUS TIMING

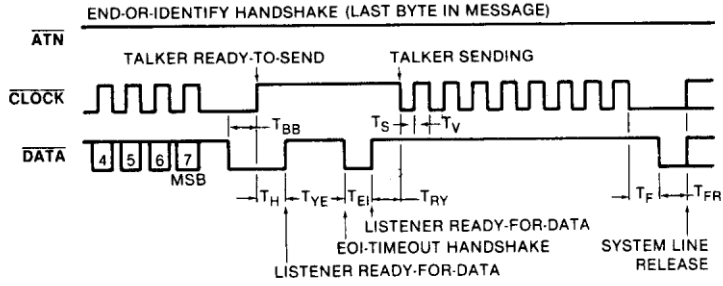
Description	Symbol	Min.	Typ.	Max.
ATN RESPONSE (REQUIRED)[1]	T_{AT}	—	—	1000µ
LISTENER HOLD-OFF	T_H	0	—	∞
NON-EOI RESPONSE TO RFD[2]	T_{NE}	—	40µs	200µs
BIT SET-UP TALKER[4]	T_S	20µs	70µs	—
DATA VALID	T_V	20µs	20µs	—
FRAME HANDSHAKE[3]	T_F	0	20	1000µs
FRAME TO RELEASE OF ATN	T_R	20µs	—	—
BETWEEN BYTES TIME	T_{BB}	100µs	—	—
EOI RESPONSE TIME	T_{YE}	200µs	250µs	—
EOI RESPONSE HOLD TIME[5]	T_{EI}	60µs	—	—
TALKER RESPONSE LIMIT	T_{RY}	0	30µs	60µs

BYTE-ACKNOWLEDGE[4]	T_{PR}	20µs	30µs	—
TALK-ATTENTION RELEASE	T_{TK}	20µs	30µs	100µs
TALK-ATTENTION ACKNOWLEDGE	T_{DC}	0	—	—
TALK-ATTENTION ACK. HOLD	T_{DA}	80µs	—	—
EOI ACKNOWLEDGE	T_{FR}	60µs	—	—

Notes:
1. If maximum time exceeded, device not present error.
2. If maximum time exceeded, EOI response required.
3. If maximum time exceeded, frame error.
4. T_V and T_{PR} minimum must be 60µs for external device to be a talker.
5. T_{EI} minimum must be 80µs for external device to be a listener.

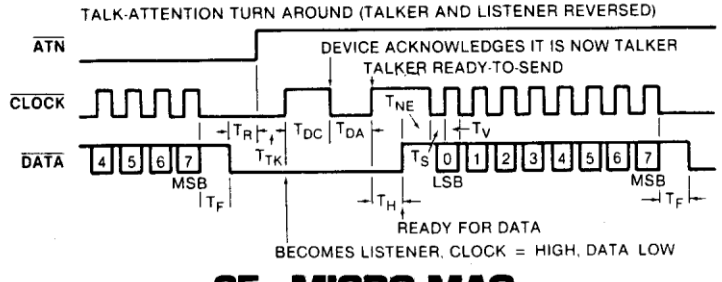
EOI-Acknowledge

Das letzte Byte eines Programmes/Datenfiles wird mit EOI=True (Low) gesendet, und zwar sowohl beim Schreiben durch den Rechner wie auch vom Peripheriegerät, wenn es als Talker aufgerufen ist. Da keine besondere Leitung für das EOI-Signal zur Verfügung steht, wird es vom jeweiligen Talker durch ein Hinauszögern bei der Übertragung des 1. Datenbits um typisch 0,2 ms nach dem Zeitpunkt CLOCK=High erzeugt. Die Device muß mitteilen, daß sie diese Überdehnung als EOI erkannt hat. Sie zieht dazu für minimal 60 Mikro-Sek. die Datenleitung auf Low (Label EO2, EN1). Danach können die 8 Bit des letzten Byte gesendet werden.



Handshake beim Turnaround

Nach der Befehlsfolge 'Device Talk' (in der Primäradresse) und der Kanalangabe in der Sekundäradresse (s.u.) nimmt der Master das ATN zu High zurück. Ferner setzt er CLOCK=High (Aufgabe der Kontrolle über diese Leitung) und DATA=Low (Ready for Data). Jetzt muß die Peripherie bestätigen, daß sie ihre künftige Talker-Rolle erkannt hat. Das geschieht diesmal auf der Leitung CLOCK, wo sie einen Low-Puls sendet (bei TDA im Diagramm). Der frühere Talker erkennt diesen Puls und setzt wenig später DATA=High. Daten- und Clockleitung werden nun von der Peripherie beherrscht bis entweder ein Frame Handshake erfolgen sollI, bis ein ATN-Befehl auftritt oder bis nach EOI eine Quittung erwartet wird.



Zusammenfassung zum Handshake

Das Handshake folgt im wesentlichen der Logik des parallelen IEEE 488-Busses. Weil aber weniger Leitungen zur Verfügung stehen, muß die gegenseitige Signalisierung in zeitlich definierten Abfolgen geschehen.

Befehlsbytes auf dem IEC-Bus

Befehle werden unter ATN=True auf den Bus gesendet. Die Regeln sind dabei für den parallelen und den seriellen IEC-Bus gleich. Es gibt Befehle, die nur aus einem Byte bestehen. Sie wenden sich an alle Peripherie und benötigen daher keine Geräteadressierung. Es sind dies die Hexbytes 3F=UNLISTEN und 5F=UNTALK.

Befehle aus 2 Bytes unter ATN bestehen aus der Primäradresse, mit der ein Gerät mit seiner eingestellten Gerätenummer aufgerufen (selektiert) und in die Funktion als Listener oder Talker gesetzt wird, und der Sekundäradresse, die einen Datenkanal des Gerätes anspricht. Bei Commodore sind Floppies ab Fabrik als Gerät '8' eingestellt, Drucker als '4'.

Die Primäradresse: Das untere Halbbyte (nibble) enthält binär die Geräteadresse x. Das obere Halbbyte enthält die Funktion. Befehlsbytes mit 2x und 3x rufen Geräte als Listener auf (LAG, Listener Address Group), wobei 3F den Sonderfall UNLISTEN bildet. Primäradressen mit 4x und 5x rufen Geräte als Talker auf (TAG, Talker Address Group), Sonderfall 5F=UNTALK. Ein zusätzlich gesetztes Bit 7 in der Primäradresse soll auf die Geräteauswahl keinen Einfluß haben.

Sekundäradressen sind gleich oder größer als hex 60. Die unteren 5 Bit im Byte bezeichnen den Datenkanal.

Commodore-Floppies

Hier gelten folgende Kanalfestlegungen in der Sekundäradresse: 0 = Programm von der Floppy lesen. 1 = Programm auf Floppy speichern. Kanäle 3-14 sind Datenkanäle, 15 ist der Befehlskanal. In OPEN-Statements hat die Sekundäradresse die Form hex Fx (x=Kanal). CLOSE-Statements haben die Form Ex. - Ein Betriebssystem für den Bus muß daher die Geräteadresse und den dazugehörigen Datenkanal zwischenspeichern. Je nach Ansprache der Peripherie werden Bits zur Primär- und zur Sekundäradresse hinzugeODERt. Neben dieser notwendigen Abspeicherung hat Commodore den Begriff des logischen Files, dem Gerät und Kanal zugeordnet sind und für das kontrolliert wird, ob das File auch geöffnet ist. Das erlaubt dann Befehle wie PRINT#x, wobei x die Nummer des logischen Files ist. Der Benutzer muß also, wenn er das File einmal geöffnet hat, nicht mehr nach der Gerätenummer und Sekundäradresse schauen.

Zur Programmierung

An einigen Stellen im nachfolgenden Listing taucht mitten in der Programmfolge die Assembler-Anweisung .BYT $2C auf. Commodore-Programmierer kennen diesen Trick seit Jahren: 2C ist der Opcode des 3 Byte langen BIT-Befehls, der den Akku nicht zerstört. Damit springt man über den 1 Byte weiter liegenden zweiten Einsprungspunkt im Programm hinweg, wo man z.B. den Akku mit einem alternativen Wert lädt.

Schlußbemerkungen

Für die bequeme Implementierung der Schnittstelle Computer/Floppy 1541 war nach den hier entwickelten Routinen ein Schnittstellenprozessor mit 28 Pin von Motorola vorgesehen, der wegen übergroßer Nachfrage derzeit leider nicht lieferbar ist und der wohl frühestens im Sommer wieder zur Verfügung stünde. Die hier und auch im DISKMON abgedruckten Routinen mögen dem Leser helfen, zwischenzeitlich eigene Lösungen zu realisieren. Verbesserungen und Überarbeitungen werden gerne entgegengenommen.

In den Betriebssystemen der Commodore VC-20 und C-64 und im DISKMON findet ein 'versetztes' Senden von Bytes statt, was dort im Zusammenhang mit dem am Ende auszusendenden EOI steht. Notwendig ist das nicht. In nachfolgender Liste wird ein Byte immer sofort gesendet. Es

ist eine Fage der Einbindung, ob man ggfs. vorher auf letztes Byte mit EOI geprüft hat, wie beim Sprung auf das Label BOUT1, wenn das Ende des Textes erkannt wurde.

Literatur

Vorstehende Diagramme wurden dem Buch Commodore 64 Programmer's Referende Guide entnommen (1983 by Howard W. Sams & Co.). Verwiesen wird ferner auf die Literaturliste zum DISKMON und auf Fisher, E.; Jensen C.W.: PET and the IEEE 488 Bus (GPIB), 1980 by McGraw-Hill sowie auf Walz, G.: Grundlagen und Anwendung des IEC-Bus. 1982 im Verlag Markt & Technik, Haar.

```
AIM 65 WITH CBM FLOPPY 1541
0000          0001 ;COPYRIGHT 1983 BY ROLAND LöHR
0000          0002 ;23.11.83

0000          0004 ;DIE FLOPPY WIRD SERIELL BEDIENT
0000          0005 ;SIGNALBELEGUNG AN DER VIA 6522:
0000          0006 ;DATA IN: PB7, CLOCK IN: PB6
0000          0007 ;INVERTER-TREIBER FÜR FOLGENDE PINS:
0000          0008 ;DATA OUT: PB5, CLOCK OUT: PB4, ATN OUT: PB3

SYMBOLERKLÄRUNG
0000          0010 VIA     =$A000
0000          0011 PORTB   =VIA
0000          0012 DDRB    =VIA+2          ;DATENRICHTUNGSREGISTER
0000          0013 ACR     =VIA+$B         ;AUXILIARY CONTROL REGISTER
0000          0014 IFR     =VIA+$D         ;INTERRUPT FLAG REGISTER
0000          0015 T1CL    =VIA+4          ;TIMER 1 COUNTER + LATCH LOW
0000          0016 T1CH    =VIA+5          ;COUNTER HIGH, START COUNTER
0000          0017 NOWLN   =$DF
0000          0018 AD1     =$F928

INTERFACE-OPERATOREN
0000          0020 ONSHOT  =$00            ;TIMER MODE
0000          0021 CLCK    =%00010000      ;CLOCKLINE SEND-BIT
0000          0022 ATN     =%00001000      ;ATNLINE SEND-BIT
0000          0023 DATA    =%00100000      ;DATALINE SEND-BIT

ALLOCATIONS
0000          0025 DATE    =$0             ;BUFFER TO ACCEPT SERIAL BYTE
0000          0026 FLAG    =$3
0000          0027 START   =$3000

REFERENCES TO THE AIM MONITOR
0000          0029 OUTALL  =$E9BC          ;OUTPUT BYTE IN A
0000          0030 TOPNO   =$F8BC

INITIALIZE PER ASSEMBLER
0000          0032         *=DDRB
A002 3F00     0033         .WOR $3F
A004          0034         *=VIA
A000 1000     0035         .WOR $10        ;SEND CLCKLO

MAIN PROGRAM
A002          0037         *=START
3000          0038 INIT
3000 A93F     0039         LDA #$3F        ;DEFINE OUTPUTS
3002 8D02A0   0040         STA DDRB
3005 A910     0041         LDA #$10        ;CLCK LO, DATA, ATN HIGH
3007 8D00A0   0042         STA PORTB       ;CLCK=LO
300A A900     0043         LDA #ONSHOT
300C 8D0BA0   0044         STA ACR         ;1 SHOT MODE FOR TIMER 1
300F 60       0045         RTS

SUBROUTINE TO SEND BYTES
3010 0940     0048 TALK    ORA #$40        ;DEVICE-# MUST BE IN A
3012 2C       0049         .BYT $2C        ;SKIP PER BIT
```

```
3013 0920     0050 LISTEN ORA #$20        ;DEVICE-# +$20
3015 8500     0051 L0     STA DATE        ;KEEP IN OUTPUT-BUFFER
3017          0052 L01
3017 201B31   0053        JSR CLCKHI      ;CLOCK HI BEFORE ATN
301A 201B31   0054        JSR DATAHI      ;ENABLE DEVICE TO HANDSHAKE
301D 200631   0055        JSR ATNLO

-------------------------------------------------------------
SEND A BYTE
3020 202431   0057 BYTOUT JSR WAIT1       ;WAIT 1 MS
3023 1001     0058        BPL L5          ;DEVICE HAS HANDSHAKED
3025 00       0059        BRK             ;*** MAKE TIMEOUT MESSAGE HERE
3026 201B31   0060 L5     JSR CLCKHI      ;CLOCK FALSE
3029 2C00A0   0061 L5A    BIT PORTB
302C 10FB     0062        BPL L5A         ;WAIT FOR DATA HI AGAIN
302E 203031   0063 L6     JSR WAIT
3031 A208     0064        LDX #8          ;COUNTER FOR 8 BIT
3033 200931   0065 L7     JSR CLCKLO      ;START 1 BIT
3036 6600     0066 L9     ROR DATE        ;LAST BIT FIRST
3038 B006     0067        BCS L9A         ;SEND HIGH
303A 200C31   0068        JSR DATALO      ;SEND LO LEVEL
303D 4C4330   0069        JMP L10
3040 201B31   0070 L9A    JSR DATAHI
3043 201B31   0071 L10    JSR CLCKHI      ;STROBE IN BIT
3046 203031   0072        JSR WAIT        ;JUST WAIT
3049 201B31   0073        JSR DATAHI      ;TAKE AWAY THE BIT
304C CA       0074        DEX
304D D0E4     0075        BNE L7          ;NEXT BIT
304F 200931   0076        JSR CLCKLO      ;NOW WAIT FOR HANDSHAKE
3052 A904     0077        LDA #$04        ;1 MS WARTEN MAX FOR FRAME
3054 8D05A0   0078        STA T1CH        ;START TIMER 40 MICROSEK
3057 2C00A0   0079 LW     BIT PORTB
305A 1006     0080        BPL LH          ;DEVICE HAS HANDSHAKED
305C 2C0DA0   0081        BIT IFR
305F 50F6     0082        BVC LW
3061 00       0083        BRK             ;*** TIME-OUT ERROR
3062 60       0084 LH     RTS
-------------------------------------------------------------
BYTE OUT WITH EOI TRUE
3063          0086 EOIOUT
3063 8500     0087        STA DATE        ;BYTE IM AKKU UEBERGEBEN
3065 201B31   0088        JSR CLCKHI
3068 2C00A0   0089 EO1    BIT PORTB       ;DATAHI VOM LISTENER?
306B 10FB     0090        BPL EO1         ;NOCH NICHT
306D A228     0091        LDX #40         ;CLOCKHI FOR 200
306F 2C00A0   0092 WL     BIT PORTB       ;DEVICE HANDSHAKES EOI?
3072 1003     0093        BPL EO2
3074 CA       0094        DEX             ;JUST A LOOP TO WAIT
3075 10FB     0095        BPL WL
3077 2C00A0   0096 EO2    BIT PORTB
307A 30FB     0097        BMI EO2
307C 2C00A0   0098 EN1    BIT PORTB       ;END OF EOI-ACKNOWLEDGE?
307F 10FB     0099        BPL EN1
3081 4C2E30   0100        JMP L6
-------------------------------------------------------------
NORMAL BYTE OUT
3084 8500     0102 BSOUT  STA DATE        ;BYTE TO BE SENT
3086 4C2630   0103        JMP L5
-------------------------------------------------------------
SEND SECONDARY ADDRESS UNDER ATN
3089 8500     0105 SECADR STA DATE
308B 202630   0106        JSR L5
308E 4C1531   0107        JMP ATNHI
-------------------------------------------------------------
SEND UNTALK
3091 A95F     0109 UNTALK LDA #$5F        ;UNTALK-BYTE
3093 2C       0110        .BYT $2C        ;SKIP ON PSEUDO-BIT
-------------------------------------------------------------
UNLISTEN
3094 A93F     0112 UNLIS  LDA #$3F        ;UNLISTEN-BYTE
3096 201530   0113        JSR L0          ;EXECUTE WITH ATN TRUE
3099 201531   0114 UNO    JSR ATNHI
309C A20A     0115        LDX #$0A
```

```
309E CA       0116 UN1     DEX
309F D0FD     0117         BNE UN1          ;WAIT CA. 40 MICROSEK
30A1 60       0118         RTS

-------------------------------------------------------------
TURN-AROUND, DEVICE BECOMES TALKER AFTER TALK
30A2 200C31   0120 TURNRD  JSR DATALO       ;READY FOR DATA
30A5 201B31   0121         JSR CLCKHI       ;LISTENER NOW
30A8 2C00A0   0122 TA1     BIT PORTB        ;WAIT FOR ACKNOWLEDGE
30AB 70FB     0123         BVS TA1          ;OF DEVICE TO BE TALKER NOW

-------------------------------------------------------------
GET A NORMAL BYTE FROM DEVICE
30AD A208     0125 GETBYT  LDX #8           ;GET 8 BIT
30AF A9C0     0126         LDA #$C0         ;LATCH TIMER 1 FOR
30B1 8D06A0   0127         STA T1CL+2       ;200 MICRO-SEC
30B4 A900     0128         LDA #0
30B6 8503     0129         STA FLAG         ;INDICATE NO EOI
30B8 2C00A0   0130 GETB1   BIT PORTB        ;END OF HANDSHAKE?
30BB 50FB     0131         BVC GETB1        ;TALKER READY TO SEND?
30BD 201831   0132         JSR DATAHI       ;OWN HANDSHAKE READY FOR DATA
30C0 A900     0133         LDA #0
30C2 8D05A0   0134         STA T1CH         ;START TIMER TO CATCH POSSIBLE EOI
30C5 2C0DA0   0135 GT1     BIT IFR          ;TIME?
30C8 701E     0136         BVS TIMOUT
30CA 2C00A0   0137         BIT PORTB        ;WAIT FOR CLCKLO
30CD 70F6     0138         BVS GT1
30CF 2C00A0   0139 GB2     BIT PORTB        ;CLOCK LO IN ALL CASES
30D2 50FB     0140         BVC GB2          ;NO BIT-STROBE AS YET
30D4 AD00A0   0141         LDA PORTB        ;GET THE DATA BIT
30D7 0A       0142         ASL A            ;INTO THE CARRY
30D8 6600     0143         ROR DATE         ;AND INTO INPUT BUFFER, LSB FIRST
30DA 2C00A0   0144 GB3     BIT PORTB        ;WAIT FOR CLCKLO AGAIN
30DD 70FB     0145         BVS GB3          ;NOT YET
30DF CA       0146         DEX              ;MORE BITS?
30E0 D0ED     0147         BNE GB2
30E2 200C31   0148 FRAME   JSR DATALO       ;OWN FRAME HANDSHAKE
30E5 A500     0149         LDA DATE
30E7 60       0150         RTS

-------------------------------------------------------------
EOI TIME-OUT
30E8 C603     0152 TIMOUT  DEC FLAG
30EA 200C31   0153         JSR DATALO       ;OWN HANDSHAKE
30ED A914     0154         LDA #20
30EF 8D04A0   0155         STA T1CL
30F2 A900     0156         LDA #0
30F4 8D05A0   0157         STA T1CH
30F7 2C0DA0   0158 TOEF    BIT IFR
30FA 50FB     0159         BVC TOEF
30FC 201B31   0160         JSR DATAHI       ;END OF HDSH
30FF 2C00A0   0161 TIM1    BIT PORTB
3102 70FB     0162         BVS TIM1         ;CLOCK STILL HI
3104 50C9     0163         BVC GB2

-------------------------------------------------------------
SET SERIAL SIGNALS LOW, WORKING WITH INVERTERS
3106 A908     0165 ATNLO   LDA #ATN
3108 2C       0166         .BYT $2C
3109 A910     0167 CLCKLO  LDA #CLCK
310B 2C       0168         .BYT $2C
310C A920     0169 DATALO  LDA #DATA
310E 0D00A0   0170 SETLO   ORA PORTB        ;SET SELECTED BIT LOW
3111 8D00A0   0171         STA PORTB
3114 60       0172         RTS

-------------------------------------------------------------

3115 A937     0174 ATNHI   LDA #$3F-ATN
3117 2C       0175         .BYT $2C
3118 A91F     0176 DATAHI  LDA #$3F-DATA
311A 2C       0177         .BYT $2C         ;SIMULATE BIT
311B A92F     0178 CLCKHI  LDA #$3F-CLCK
311D 2D00A0   0179 SETHI   AND PORTB        ;MASK OFF SELECTED BIT, SET HIGH
3120 8D00A0   0180         STA PORTB
3123 60       0181         RTS
```

```
-------------------------------------------------------------
WAIT 1 MS
3124 200931   0183 WAIT1  JSR CLCKLO      ;ENABLE HANDSHAKE
3127 A2B8     0184        LDX #$B8
3129 CA       0185 WA1    DEX
312A D0FD     0186        BNE WA1
312C 2C00A0   0187        BIT PORTB
312F          0188                        ;GET STATUS
312F 60       0189        RTS
-------------------------------------------------------------
JUST WAIT A BIT
3130 EA       0191 WAIT   NOP
3131 EA       0192        NOP
3132 EA       0193        NOP
3133 EA       0194        NOP
3134 60       0195        RTS
3135          0196        .END

  ERRORS= 0000
```

Roland Löhr

6805/68705 - Assembler

unter Fig-FORTH

Einleitung

Die Mikroprozessorfamilie 6805/68705xx von Motorola wurde in Heft 27 dieser Zeitschrift vorgestellt. In ihr sind die Einchipper mit RAM, EPROM (nebst Urladerprogramm), Timer und Interfacepins für viele Steuerungsaufgaben interessant, vor allem, wenn es um kleine Serien geht oder wenn Betriebsparameter individuell eingebrannt werden müssen. - Der Befehlssatz ist sehr regelmäßig angelegt (s. Abb. 'Instruction Set Opcode Map') und gleicht in der Wirkung dem der CMOS-CPU R65C02, vor allem mit den Befehlen zum Setzen und Löschen von Bits in der Zero Page und mit den Befehlen Branch on Bit Set/Clear. Daneben hat das X-Register (8 Bit breit) einen Inherent-Befehlssatz, der mit dem für den Akku deckungsgleich ist (z.B. also auch Verschieben in X). Der in 6502 geübte Programmierer findet sich mithin schnell in die hier gegebenen Möglichkeiten ein und gelangt nach einiger Übung zu sehr knapp formulierten und effektiven Befehlssequenzen z.B. mit 3 Bytes für das Warten auf ein Ereignis an Port A:

```
WAIT  BRCLR5 PORTA WAIT  ;Rückverzweigung
```

Der nachfolgende Artikel möchte nun dem Leser einerseits ein Hilfsmittel in die Hand geben, um diese Prozessorfamilie mit einem Cross-Assembler programmieren zu können, wenn er FORTH auf seiner Maschine hat. Der Assembler ist also portabel, soweit ein Fig-FORTH auf der Maschine des Anwenders läuft.

Andererseits soll gezeigt werden, wie man einen 1-Pass-Assembler für eine fremde CPU bei Bedarf kreiert. Nach dem Studium der nachfolgenden Quelltext-Liste wird der Leser feststellen, daß er dann eigentlich nur neue Mnemonics benötigt und bei anders indizierbaren Befehlen oder solchen mit Pre- und Postbytes eine andere Aufbereitung des Grund-Opcodes und der Befehlslänge vornehmen muß wie z.B. hier in den benutzten DEFINING WORDs MT (bei REM 10) oder RMW (bei REM 11) für indizierbare arithmetische bzw. Read/Modify/Write-Befehle. In der Tat wurde mit den hier abgedruckten Grundstrukturen ebenfalls ein Cross-Assembler für den MC6809 entwickelt. Für ihn wurden auch die Kontrollstrukturen bei BEGIN, UNTIL, usw. auf die mit den Long Branches gegebenen Möglichkeiten angepaßt. .

Kritik an bisherigen FORTH-Assemblern

Assembler unter FORTH für 65xx basieren offensichtlich auf Arbeiten von W. F. Ragsdale, veröffentlicht u.a. in Dr. Dobb's Journal 9/1981. Es ist ihr Ziel, maschinensprachlichen Code in FORTH-Worten ausführbar zu machen. Ihr Nutzen liegt bei Aufgaben wie z.B. 'Strings für

M6805 FAMILY INSTRUCTION SET OPCODE MAP

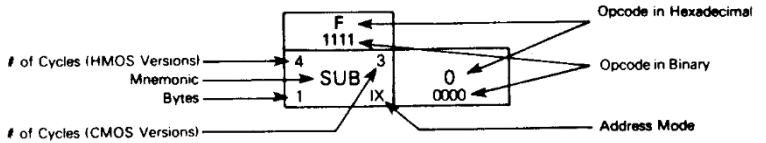
Hi / Low	Bit Manipulation BTB 0 0000	Bit Manipulation BSC 1 0001	Branch REL 2 0010	Read/Modify/Write DIR 3 0011	Read/Modify/Write INH(A) 4 0100	Read/Modify/Write INH(X) 5 0101	Read/Modify/Write IX1 6 0110	Read/Modify/Write IX 7 0111	Control INH 8 1000	Control INH 9 1001	Register/Memory IMM A 1010	Register/Memory DIR B 1011	Register/Memory EXT C 1100	Register/Memory IX2 D 1101	Register/Memory IX1 E 1110	Register/Memory IX F 1111	Hi / Low
0 0000	BRSET0 BTB DIR	BSET0 BSC DIR	BRA REL	NEG DIR	NEGA INH	NEGX INH	NEG IX1	NEG IX	RTI INH		SUB IMM	SUB DIR	SUB EXT	SUB IX2	SUB IX1	SUB IX	0 0000
1 0001	BRCLR0 BTB DIR	BCLR0 BSC DIR	BRN REL						RTS INH		CMP IMM	CMP DIR	CMP EXT	CMP IX2	CMP IX1	CMP IX	1 0001
2 0010	BRSET1 BTB DIR	BSET1 BSC DIR	BHI REL								SBC IMM	SBC DIR	SBC EXT	SBC IX2	SBC IX1	SBC IX	2 0010
3 0011	BRCLR1 BTB DIR	BCLR1 BSC DIR	BLS REL	COM DIR	COMA INH	COMX INH	COM IX1	COM IX	SWI INH		CPX IMM	CPX DIR	CPX EXT	CPX IX2	CPX IX1	CPX IX	3 0011
4 0100	BRSET2 BTB DIR	BSET2 BSC DIR	BCC REL	LSR DIR	LSRA INH	LSRX INH	LSR IX1	LSR IX			AND IMM	AND DIR	AND EXT	AND IX2	AND IX1	AND IX	4 0100
5 0101	BRCLR2 BTB DIR	BCLR2 BSC DIR	BCS REL								BIT IMM	BIT DIR	BIT EXT	BIT IX2	BIT IX1	BIT IX	5 0101
6 0110	BRSET3 BTB DIR	BSET3 BSC DIR	BNE REL	ROR DIR	RORA INH	RORX INH	ROR IX1	ROR IX			LDA IMM	LDA DIR	LDA EXT	LDA IX2	LDA IX1	LDA IX	6 0110
7 0111	BRCLR3 BTB DIR	BCLR3 BSC DIR	BEQ REL	ASR DIR	ASRA INH	ASRX INH	ASR IX1	ASR IX		TAX INH		STA DIR	STA EXT	STA IX2	STA IX1	STA IX	7 0111
8 1000	BRSET4 BTB DIR	BSET4 BSC DIR	BHCC REL	LSL DIR	LSLA INH	LSLX INH	LSL IX1	LSL IX		CLC INH	EOR IMM	EOR DIR	EOR EXT	EOR IX2	EOR IX1	EOR IX	8 1000
9 1001	BRCLR4 BTB DIR	BCLR4 BSC DIR	BHCS REL	ROL DIR	ROLA INH	ROLX INH	ROL IX1	ROL IX		SEC INH	ADC IMM	ADC DIR	ADC EXT	ADC IX2	ADC IX1	ADC IX	9 1001
A 1010	BRSET5 BTB DIR	BSET5 BSC DIR	BPL REL	DEC DIR	DECA INH	DECX INH	DEC IX1	DEC IX		CLI INH	ORA IMM	ORA DIR	ORA EXT	ORA IX2	ORA IX1	ORA IX	A 1010
B 1011	BRCLR5 BTB DIR	BCLR5 BSC DIR	BMI REL							SEI INH	ADD IMM	ADD DIR	ADD EXT	ADD IX2	ADD IX1	ADD IX	B 1011
C 1100	BRSET6 BTB DIR	BSET6 BSC DIR	BMC REL	INC DIR	INCA INH	INCX INH	INC IX1	INC IX		RSP INH		JMP DIR	JMP EXT	JMP IX2	JMP IX1	JMP IX	C 1100
D 1101	BRCLR6 BTB DIR	BCLR6 BSC DIR	BMS REL	TST DIR	TSTA INH	TSTX INH	TST IX1	TST IX		NOP INH	BSR REL	JSR DIR	JSR EXT	JSR IX2	JSR IX1	JSR IX	D 1101
E 1110	BRSET7 BTB DIR	BSET7 BSC DIR	BIL REL						STOP* INH		LDX IMM	LDX DIR	LDX EXT	LDX IX2	LDX IX1	LDX IX	E 1110
F 1111	BRCLR7 BTB DIR	BCLR7 BSC DIR	BIH REL	CLR DIR	CLRA INH	CLRX INH	CLR IX1	CLR IX	WAIT* INH	TXA INH		STX DIR	STX EXT	STX IX2	STX IX1	STX IX	F 1111

Abbreviations for Address Modes

INH	Inherent
IMM	Immediate
DIR	Direct
EXT	Extended
REL	Relative
BSC	Bit Set/Clear
BTB	Bit Test and Branch
IX	Indexed (No Offset)
IX1	Indexed, 1 Byte (8-Bit) Offset
IX2	Indexed, 2 Byte (16-Bit) Offset

* CMOS Versions Only

LEGEND



65xx MICRO MAG

FORTH' (Heft 24 65xx MICRO MAG) klar auf der Hand. Assembler-Code ist aber immer von der notwendigen Verwaltungsinformation des FORTH umgeben und gelangt an die nächst freie Stelle im DICTIONARY, wenn man nicht den Dictionary Pointer DP manipuliert. Für das Herstellen eines größeren Maschinenprogrammes, das zudem als Vorlage für das Brennen von EPROMs dienen soll, sind so gebaute Assembler nicht geeignet. In einer solchen Umgebung ist ja außerdem zu fordern, daß der Code für einen virtuellen Speicherraum zu erzeugen ist, der mit dem Adreßraum, in dem die Ablage des Codes erfolgt, nicht identisch ist oder sein muß.

Die zweite Kritik ist, daß bei FORTH-Assemblern (wie z.B. für den AIM 65) bisher nicht mit externen Symbolen oder internen Labeln gearbeitet werden konnte, auch nicht mit Assembleranweisungen z.B. für die Byte- und Textkettenablage.

Der nachstehende 1-Pass-Cross-Assembler kennt solche Beschränkungen nicht. Er erzeugt Code an jeder freien Stelle im RAM, Code, der von den FORTH-Headern (Kopfinformation in den Worten) frei ist und der sich auf einen beliebigen virtuellen Speicherraum beziehen darf. Es kann mit Symbolen und Labeln gearbeitet werden, Bytes und Textketten können abgelegt und Speicherplätze reserviert oder bezeichnet werden. Auf eine noch nicht ganz zufriedenstellende Art ist in Verbindung mit dem Text-Editor des AIM auch eine Assembler-Liste herstellbar.

Zur Programmierung mit FORTH-Assemblern informiere man sich im FORTH User's Manual (Rockwell für den AIM 65) und in dieser Zeitschrift an folgenden Stellen: FORTH (3) (in Heft 26), Cross-Assembler unter FORTH (Heft 27), Macro-Assembler unter FORTH (Heft 28).

Zum Programmaufbau

Ein weiterer Artikel wird auf die Programmierung mit diesem Assembler eingehen, so daß hier zunächst einmal Erklärungen zum Aufbau gegeben werden sollen. Es werden dabei Bezüge zu den Bemerkungen (REM) in der Liste hergestellt.

Auch dieser Assembler verlangt die umgekehrt polnische Eingabe der Operanden, der Adressierungsart und des mnemonischen Befehls. Der Assembler bildet ein eigenes Vokabular (REM 1), das keinerlei Kenntnis vom 6502-Assembler hat, dem aber alle Grundleistungen des FORTH zu Gebote stehen. Die neuen Worte sind DEFINITIONS im Vokabular ASS.

Variable und Konstanten

Es werden die Variablen Programmzähler PC und Zieladressenbereich RAMAREA geführt. Die Wortfolge z.B.

3000 ASSIGN (bei REM 4)

legt die Startadresse des Zielbereiches in RAMAREA ab. Der Befehl 80 ORG setzt den virtuellen Programmzähler z.B. auf 80 (alles in HEX !). Das Wort VLOC (Virtual Location) holt jederzeit den augenblicklichen Stand des Programmzählers PC auf den Datenstack, wo der zu Labels oder zum Ausdruck verwertet werden kann. Also z.B. VLOC LABEL HIER. Weitere Hilfsworte dienen der inneren Verwaltung, wie REL, LOCATE (bei REM 4) BACK (bei REM 6) usw..

Prozessor-Konstanten

Bei REM 2 sind Standard-Adressen des Prozessors als Systemkonstante für die symbolische Programmierung definiert worden, Z.B. PORTA, DDRA (Datenrichtungsregister) etc. Mit ihnen kann man z.B. programmieren:

```
PORTA LDA,      lade aus Port A
80 RESET .WOR   lege das Wort 0080 bei RESET ab
```

Man beachte, daß einige Systemkonstante, die hier für den Typ 68705P3 definiert wurden, bei anderen Mitgliedern der Familie gem. Datenblatt anzupassen sind.

Menmonische Befehle

Alle Befehle haben die Schreibweise lt. vorstehender Instruction Set Opcode Map mit angekängtem Komma (!), also z.B.

BCLR6, RTS, STX, usw.

Die Adressierungsarten

Die Art der beabsichtigten Adressierung wird in der FORTH-Variablen MODE erfaßt, die nach jeder Programmzeile automatisch mit 0MODE auf 0 zurückgesetzt wird, was für Adressierung Direct Page oder extended steht. MODE wird durch die Operatoren für die Adressierungsart verändert. - Sofern nicht indiziert wird, versucht der Assembler zunächst immer eine kurze Zero Page-Adressierung herzustellen. Gelingt das nicht, so bildet er Extended-Befehle mit 2 Bytes für die Adresse. Wir haben damit folgende Schreibweisen für die Adressierungsarten:

Operator	Adressierungsart
#	unmittelbar
ohne	Direct Page oder extended, natürlich auch inherent,
(X)	effektive Adresse in Register X
,X	indiziert mit X, automatisch vom Assembler optimiert zu IX1 (1 Byte Offset), ersatzweise zu IX2 (2 folgende Adreßbytes)

Zahlenbasis

Zahlen werden grundsätzlich als Hexzahlen interpretiert. Eine Zahl mit vorangestelltem BIN oder % (siehe vor REM 1) wird als Binärzahl interpretiert, z.B.

```
% 101 # LDA,          lade Akku unmittelbar mit 3
```

Der binäre Modus gilt nur für die laufende Zeile.

DEFINING WORDs für die Befehlsgruppen

Für die einzelnen Befehlsgruppen wurden DEFINING WORDs angelegt, die den Grund-Opcode zur Ausführungzeit auf den Datenstack bringen und danach abhängig von der Adressierungsart in MODE die logische Aufbereitung und Abwandlung durch Addition vornehmen. Die meisten Befehle in der Aufbereitung betreffen dabei Prüfungen, das Duplizieren und das Umstellen von Argumenten. Diese DEFINING WORDs sind BR für Branches (vor REM 7), INH für Inherent-Befehle (bei REM 7), BSC bei REM 8 für die Gruppe Bit Set/Clear, MT bei REM 10 für ladende, speichernde, arithmetische und logische Befehle und bei REM 11 mit RMW die Gruppe Read/Modify/Write für Befehle wie NEG, COM, etc.. - Der endgültige Opcode wird mit WR1 ins RAM geschrieben, die Adressen mit WR1 und WR2 (für 2 Bytes).

Kontrollstrukturen

Wir haben die bekannten Abfolgen bei REM 16. Kontroll-Statements haben immer ein angehängtes Komma (!), um sie von denen des FORTH zu unterscheiden. Im erlaubten Umfange und soweit die inneren Strukturen immer zuerst aufgelöst werden, dürfen die Kontrollstrukturen beliebig verschachtelt werden. Die Worte REP und THENN sind Hilfsworte für ähnlich klingende. Die Hauptworte wie REPEAT, THEN usw. sind als IMMEDIATE deklariert, damit sie ihre Wirkung bereits zur Zeit des Compilierens erbringen. - Bei diesen Strukuren wird grundsätzlich mit Verzweigungsbefehlen gearbeitet und nicht mit JMP, so daß man auch 'von Hand' keinen kürzeren Code erzeugt. Die richtige Abfolge der Strukturanweisungen wird mit ?PAIRS auf Zulässigkeit überprüft (Compiler-Sekurität).

Bedingte Verzweigungen

Zu den Kontrollstrukturen mit IF, .. THEN, usw. gehören Abfragen. Wir finden sie bei REM 13. Abfragen des Prozessorstatus beginnen immer mit einem Punkt vor dem Mnemonic, also z.B.

```
.NE IF,    (solange noch nicht EQUAL)
```

Abfragen dieser Gruppe erzeugen immer die antivalente Verzweigung zur eigentlich gedachten. Im vorstehenden Fall wird also mit .NE der Befehl BEQ auf die Stelle bei ELSE, (wenn es folgen sollte) oder THEN, erzeugt, an der das Programm fortzusetzen ist. Das IF, bewirkt eine interne Speicherung des entsprechenden Standes von PC. Das später folgende THEN, setzt dann bei IF, und ggfs. auch bei ELSE, die Verzweigungsweiten ein.

Bedingte Verzweigungen für die Gruppe Branch on Bit Set/Clear

In der ersten Spalte der obenstehenden Tabelle für die Motorola-Befehle sehen wir die Gruppe BRSET0 bis BRCLR7. Diese Verzweigungsbefehle sind 3 Byte lang: 1. Opcode, 2. Adresse in der Zero Page und 3. Verzweigungsweite (Offset) nach den üblichen Regeln berechnet (der Folgebefehl hat die Weite 0). Auch diese Verzweigungsanweisungen sollten in den strukturierten Statements benutzt werden können. Mit vorangestelltem Punkt hätten sich Wortungetüme ergeben. Deswegen wurden bei REM 15 folgende kurzen Verzweigungsworte geschaffen:

.BC0 bis .BC7 und .BS0 bis .BS7

Diese Statements arbeiten ebenso wie die vorstehenden mit IF, ELSE, THEN' usw. zusammen und erzeugen wie diese die jeweils antivalenten Verzweigungsbefehle. Es mag sein, daß bei zu großer Verzweigungsweite (Out of Range) nicht immer ein WARN5 ausgegeben wird.

Verzweigungsbefehle ohne Kontrollstrukuren

Puristen höherer Sprachen fordern, daß man nur strukturiert programmieren dürfte. Die bisherigen FORTH-Implementierungen sind in diesem Sinne puristisch und enthalten auch im Assembler keine Branches, sondern nur Conditionals für Strukturen. In Assemblerspache tritt für eine Vorwärtsreferenz (ohne Benutzung eines Labels) die einfache Überlegung auf: Überspringe den nächsten Befehl. Wir sagen dann z.B. BEQ *+4.

Ein strukturierter 1-Pass-Assembler läßt normalerweise keine Vorwärtsreferenzen dieser Art zu, was oft zu Formulierungsschwierigkeiten führt. Als Nicht-Purist hat der Autor daher alle Verzweigungsbefehle mit ihren Mnemonics implementiert. Als Mnemonics mit angehängtem Komma (im Gegensatz zu den Conditionals - s.o. - mit beginnendem Punkt) legen sie nur ihren Opcode an der laufenden Stelle ab. Es steht in der Verantwortung des Programmierers, den Offset dann 'per Hand' zu programmieren. Will man z.B. 3 Bytes nach vorn springen, so sagt man

BEQ, 03 WR1 oder entsprechend.

Für im Programmverlauf vorhergehende erklärte Label ist auch eine Formulierung wie folgt möglich:

BHI, Name <REL,

womit die Verzweigungsweite zu Name errechnet und ins RAM eingetragen wird. Auch symbolische Vorwärtsreferenzen sind möglich und sollen später einmal erwähnt werden.

Die Mnemonics für einfache Verzweigungsbefehle stehen in der Liste ab REM 12 (immer mit angehängtem Komma) und für die Gruppe on Bit Set/Clear ab REM 14. Den dortigen Wortungetümen wurde nicht auch noch ein Komma angehängt. Es steht dem Leser frei, dort kürzere einsehbare Mnemonics zu verwenden.

Assembler-Anweisungen (Direktiven)

Ab REM 17 am Schluß der Liste sehen wir die Reihe der Assembler-Anweisungen, die sowohl dem 6502- wie auch dem Motorola-Sprachgewandten entgegen kommen sollen. Viele Formulierungen sind wahlfrei oder gleichwertig, auch doppelt, um Tippfehler verzeihlicher zu machen:

.FCB	Form Constant Byte. Bei 6502: .BYT
.FDB	Form Double Byte = .WOR, Wortablage
.RMB	Reserve Memory Bytes. 6502: *=*+x
"	Beginn eines abzulegenden ASCII-Textes. Auf " muß ein Space folgen!
.FCC	Form Constant Characters, wie vor: Ablage einer ASCII-Textkette
.SPC	= SKIP
.EQ	oder EQ: Symbolerklärung

Für die Erzeugung einer Assembler-Liste haben wir die Anweisung .OPTLIS bzw. .OPTNOL, wobei die Formatierung noch keineswegs befriedigend ist.

Hilfsanweisungen

T bewirkt beim AIM den direkten Übergang von FORTH auf die TOP-Zeile des Editors.

KIL bringt einen Kontrolltext und den Datenstack auf die aktive Ausgabe, um zu sehen, ob man in einer Struktur ankommt.

LOCATE bringt die Adresse auf den Datenstack, in der die nächste Abspeicherung erfolgt. LOCATE kann aber auch nach ORG (Seten des virtuellen Programmzählers) benutzt werden, um in das Zielgebiet des Codes einen Bereich für das EPROM zunächst auf FF zu bringen, also z.B. LOCATE HEX 1000 FF FILL.

Die Programm-Liste

Nachfolgend nun die Implementierung des Cross-Assemblers unter FORTH. Ein weiterer Beitrag wird Hantierungsbeispiele bringen. - Die bisherigen Ausführungen sollten aber auch schon für einen 'Auto-Start' ausreichen. Verbesserungshinweise werden gerne entgegen genommen. - Wer sich die Mühe des Eintippens und Abstimmens ersparen will, kann beim Autor 2 EPROMs mit dem compilierten Code zu DM 100,- (Endpreis) beziehen.

```
( TOP)
( FORTH-ASSEMBLER FUER DEN MC68705P3 VON MOTOROLA)
( COPYRIGHT 19.11.83 BY ROLAND LOEHR)
( RECORDED ASS39.SEQ)
FORGET TASK
HEX
F6CF VARIABLE EDI ( T-ENTRY INTO EDITOR ADDR)
: T EDI EXECUTE ;

: BIN 2 BASE ! ;
: % 2 BASE ! ; ( FOR BINARY ARGUMENTS)
: KIL ( CONTROL WORD)
CR ." ******** KILROY WAS HERE" CR ." THE STACK IS:" .S CR
;

( REM 1)
VOCABULARY ASS IMMEDIATE ASS DEFINITIONS

0 VARIABLE PC  ( PC OF MC68705 ASSEMBLER)
0 VARIABLE RAMAREA    ( ALLOCATE ROOM FOR COMPILATION)
0 VARIABLE LISTEN

CREATE LIST 20 C, 27 C, F7 C, 4C C, 5A C, B0 C,
  SMUDGE ( SET TO CURRENT LINE)
: ?LIST LISTEN § IF LIST THEN  ; ( SHALL I LIST?)
: .OPTLIS 1 LISTEN ! ;   ( LIST ON)
: .OPTNOL 0 LISTEN ! ; ( OFF)

: WARN CR ." WARNING: WRITE-ONLY REGISTER" CR ;
 : WARN5 CR ." BRANCH OUT OF RANGE" CR ;

( REM 2)
( ADRESSES OF THE INTERFACE CHIP REGISTERS)
0 CONSTANT PORTA
1 CONSTANT PORTB
2 CONSTANT PORTC
3 CONSTANT PORTD

( DATA DIRECTION REGISTERS)
: DDRA 4 WARN ;
: DDRB 5 WARN ;
: DDRC 6 WARN ;
: DDRD 7 WARN ;

8 CONSTANT TDR    ( TIMER DATA REGISTER)
```

```
9 CONSTANT TCR ( TIMER CONTROL REGISTER)
A CONSTANT MISC
B CONSTANT PCR    ( PROGRAM CONTROL REGISTER)
( C, D NOT USED)
E CONSTANT A/DCR    ( A/D CONTROL REGISTER)
F CONSTANT A/DDR    ( A/D DATA REGISTER)

( RESET & INTERRUPT VECTORS)

7FE CONSTANT RESET
7FC CONSTANT SWI
7FA CONSTANT INT
7F8 CONSTANT TIMER/INT2

( REM 3)
784 CONSTANT MOR    ( MASK OPTION REGISTER)
785 CONSTANT BOOT    ( EPROM PROGRAM)
10 CONSTANT RAMLOW    ( LOWEST ADDRESS OF USER RAM)
7F CONSTANT RAMTOP

( REM 4)
: VLOC PC § ;    ( GET VIRTUAL ADDRESS)
: ?RAM+ RAMAREA § + ;
: LOCATE VLOC ?RAM+ ;
: ASSIGN RAMAREA !
CR ." CODE STORED TO LOCATION  " LOCATE 0 D. CR ;
: <REL VLOC 1+ - ;   ( OFFSET FOR SHORT BRANCHES)


: OMODE 0 MODE ! VLOC CR ." *=" 0 D. ;
: OL OMODE ?LIST ;

: ERR OMODE CR 3 ERROR ;    ( OUTPUT ERROR MESSAGE)

( REM 5)
: WR1 HEX DUP LOCATE C! . SPACE 1 PC +! ;
: WR2 HEX DUP DUP FF00 AND IF ELSE 30 EMIT 30 EMIT THEN
. SPACE
0 100 U/ SWAP 100 * + LOCATE ! 2 PC +! ;

( REM 6)
: BACK <REL DUP    ( CALC. OFFSET TO BEGIN)
FF80 U< ( OUT OF RANGE)
IF 3 ERR
ELSE FF AND WR1
THEN
;
( DEFINING WORDS FOR GROUPS OF OPERATIONS & MNEMONICS)
: BR ( DEFINING WORD FOR SHORT BRANCHES)
<BUILDS C, DOES>
C§ WR1 <REL DUP DUP ( OFFSET)
0< ( 1) IF FF80 U< ( 2) IF WARN5 ( 2) THEN
( 1) ELSE 7F > ( 3) IF WARN5 ( 3) THEN
( 1) THEN
FF AND WR1
OMODE
;
```

```
( REM 7)
: INH ( DEFINING WORD FOR 1-BYTE OPCODES)
   <BUILDS C, DOES> C$ WR1 OL ; ( WRITE OPCODE TO)

97 INH TAX,    9F INH TXA,    99 INH SEC,
98 INH CLC,    9B INH SEI,    9A INH CLI,
83 INH SWI,    81 INH RTS,    80 INH RTI,
9C INH RSP,    9D INH NOP,

: WARN1 ." WARNING: FOR CMOS ONLY" CR ;
: STOP, 8E WR1 WARN1 ?LIST ;
: WAIT, 8F WR1 WARN1 ?LIST ;

( COMMANDS WORKING ON OPERAND IN ACC)

40 INH NEGA,    43 INH COMA,    44 INH LSRA,
46 INH RORA,    47 INH ASRA,    48 INH LSLA,
49 INH ROLA,    4A INH DECA,    4C INH INCA,
4D INH TSTA,    4F INH CLRA,    48 INH ASLA,

( COMMANDS WORKING ON OPERAND IN REGISTER X )

50 INH NEGX,    53 INH COMX,    54 INH LSRX,
56 INH RORX,    57 INH ASRX,    58 INH LSLX,
59 INH ROLX,    5A INH DECX,    5C INH INCX,
5D INH TSTX,    5F INH CLRX,
58 INH ASLX,    5A INH DEX,    5C INH INX,

( REM 8)
: BSC ( DEFINING WORD FOR BIT SET & CLEAR GROUP)
<BUILDS C, DOES>
C$ WR1 WR1 ( GET OPCODE, LAY DOWN, GET ADDR. LAY DOWN)
OL
;
10 BSC BSET0,    12 BSC BSET1,    14 BSC BSET2,
16 BSC BSET3,    18 BSC BSET4,    1A BSC BSET5,
1C BSC BSET6,    1E BSC BSET7,

11 BSC BCLR0,    13 BSC BCLR1,    15 BSC BCLR2,
17 BSC BCLR3,    19 BSC BCLR4,    1B BSC BCLR5,
1D BSC BCLR6,    1F BSC BCLR7,

: BSET, 2 * 10 + WR1 WR1 OL ;   ( ALLOW STATEMENTS)
: BCLR, 2 * 11 + WR1 WR1 OL ;    ( ADDRESS BIT MN)

( REM 9)
( ADDRESSING MODES)
: # 80 MODE ! ; ( SET A FLAG)
: ,X 30 MODE ! ; ( 2 INDEXED ADDRESSING MODES)
: (X) 50 MODE ! ; ( EFFECTIVE ADDRESS IS IN REGISTER X)

( REM 10)
( COMMANDS WITH MULTIPLE ADDRESSING MODES)
: MT ( DEFINING WORD)
   <BUILDS C, DOES>
C$    ( BASIS OPCOCE)
MODE C$ DUP 50 =
```

```
   IF + WR1 OMODE    ( IT IS INDIRECT REGISTER X)
   ELSE 80 =    ( IMMEDIATE MODE)
    IF DUP A7 = IF ERR
                ELSE DUP AC = IF ERR
                              ELSE DUP AF =
                               IF ERR
                               ELSE OVER FF00 AND
                                  IF ERR
                                  ELSE WR1 WR1 THEN
                               THEN
                              THEN
                THEN
    ELSE OVER FF00 AND    ( IT'S DIRECT, EXTENDED, INDEXED?)
      IF ( 16-BIT) MODE § 0=
         IF 20 + WR1 WR2 ( EXTENDED)
         ELSE 30 + WR1 WR2    ( 16-BIT OFFSET, INDEXED)
         THEN
      ELSE ( 8-BIT) MODE § 0=
         IF 10 + WR1 WR1 ( DIRECT PAGE)
         ELSE 40 + WR1 WR1    ( 8-BIT OFFSET, INDEXED)
         THEN
      THEN
     THEN
    THEN
OL
;
A0 MT SUB,    A1 MT CMP,    A2 MT SBC,    A3 MT CPX,
A4 MT AND,    A5 MT BIT,    A6 MT LDA,    A7 MT STA,
A8 MT EOR,    A9 MT ADC,    AA MT ORA,    AB MT ADD,
AC MT JMP,    AD MT JSR,    AE MT LDX,    AF MT STX,

( REM 11)
( READ/MODIFY/WRITE GROUP OF COMMANDS)

: RMW    ( DEFINING WORD)
  <BUILDS C, DOES>    ( STORE BASIC OPCODE)
  C§ OVER FF00 AND    ( OPCODE FEDCH, SEE IF EXTENDED)
    IF ERR    ( NO EXTENDED ADDRESSING ALLOWED)
  ELSE ( 8-BIT) MODE § DUP 0=
   IF DROP WR1 WR1 ( WAS DIRECT PAGE)
   ELSE DUP 50 = IF DROP 40 + WR1 (  X-REGISTER INDIR)
                 ELSE  30 = IF 30 + WR1 WR1 ( IX-MODE)
                           ELSE ." WRONG MODE !!!" ERR
THEN THEN THEN THEN
OL
;

( THE COMMANDS OF THE READ/MODIFY/WRITE GROUP)

30 RMW NEG,    33 RMW COM,    34 RMW LSR,
36 RMW ROR,    37 RMW ASR,    38 RMW LSL,
39 RMW ROL,    3A RMW DEC,    3C RMW INC,
3D RMW TST,    3F RMW CLR,    38 RMW ASL,

( BRANCHING INSTRUCTIONS)
```

```
( REM 12)
( CONDITIONALS LAY DOWN OPCODE BUT NOT OFFSET)

22 INH BHI, 23 INH BLS,    ( HIGHER, HIGHER OR SAME)
24 INH BCC, 25 INH BCS,    ( CARRY CLEAR, SET)
24 INH BHS, 25 INH BLO,    ( HIGHER OR SAME, LOWER)
26 INH BNE, 27 INH BEQ,    ( NOT EQUAL, EQUAL ZERO)
28 INH BHCC, 29 INH BHCS,    ( HALF CARRY CLEAR, SET)
2A INH BPL, 2B INH BMI,    ( PLUS, MINUS=NEGATIVE)
2C INH BMC, 2D INH BMS,    ( INTERRUPT MASK CLEAR, SET)
2E INH BIL, 2F INH BIH,    ( INTERRUPT LINE LOW, HIGH)

( REM 13)
( CONDITIONALS FOR STRUCTURED STATEMENENTS WITH)
( WHILE, UNTIL, ETC.)
23 INH .GT    ( WHILE GREATER THAN)
22 INH .LE    ( WHILE LESS OR EQUAL)
25 INH .GE    ( GREATER OR EQUAL)
24 INH .LT    ( LESS THAN)
27 INH .NE    ( WHILE NOT EQUAL)
26 INH .EQ    ( EQUAL ZERO)
29 INH .HCC   ( WHILE HALF CARRY CLEAR
28 INH .HCS   ( HALF CARRY SET)
25 INH .CC    ( WHILE CARRY CLEAR)
24 INH .CS    ( CARRY SET)
2B INH .+     ( WHILE POSITIVE)
2A INH .-     ( WHILE NEGATIVE)
2D INH .IMC   ( WHILE INTERRUPT MASK CLEAR)
2C INH .IMS   ( WHILE INTERRUPT MASK SET)
2F INH .ILL   ( INTERRUPT LINE LOW)
2E INH .ILH   ( WHILE INTERRUPT LINE HIGH)


AD INH BRS,
21 INH BRN,
20 INH BRA,


( REM 14)
( BTB BIT TEST & BRANCH GROUP OF INSTRUCTIONS)

00 BSC BRSET0 02 BSC BRSET1 04 BSC BRSET2 06 BSC BRSET3
08 BSC BRSET4 0A BSC BRSET5 0C BSC BRSET6 0E BSC BRSET7
01 BSC BRCLR0 03 BSC BRCLR1 05 BSC BRCLR2 07 BSC BRCLR3
09 BSC BRCLR4 0B BSC BRCLR5 0D BSC BRCLR6 0F BSC BRCLR7

( REM 15)
( BIT TEST & BRANCH FOR CONTROL STRUCTURES LIKE WHILE)
00 BSC .BC0 02 BSC .BC1 04 BSC .BC2 06 BSC .BC3
08 BSC .BC4 0A BSC .BC5 0C BSC .BC6 0E BSC .BC7

01 BSC .BS0 03 BSC .BS1 05 BSC .BS2 07 BSC .BS3
09 BSC .BS4 0B BSC .BS5 0D BSC .BS6 0F BSC .BS7

: BRSET, 2 * WR1 WR1 ;    ( ALLOW STATEMENTS LIKE)
: BRCLR, 2 * 1+ WR1 WR1 ;
   ( CONDITION, ADDRESS, BIT-#, MNEMONICS)
```

```
( REM 16)
( CONTROL STRUCTURES)

: AGAIN, ?EXEC 1 ?PAIRS
BRA, BACK
; IMMEDIATE
: BEGIN, VLOC 1 ?LIST ; IMMEDIATE
: WHILE, ?EXEC 1 ?PAIRS 0 WR1 VLOC 3 OL
; IMMEDIATE
: REP
   VLOC 2+    ( GET OFFSET)
OVER - DUP 80 < IF SWAP ?RAM+ 1- C! ( <$80 ?)
     ELSE 3 ERR THEN
;

: REPEAT, ?EXEC 3 ?PAIRS REP 20 FF AND WR1 BACK ;
: IF, VLOC 0 WR1 ( FOR OFFSET) 2 ( FLAG) ?LIST ; IMMEDIATE

: THENN VLOC 1- OVER - DUP 80 <
   IF SWAP ?RAM+ C!
   ELSE 3 ERR THEN
;
: UNTIL, ?EXEC 1 ?PAIRS
   VLOC 1+ - DUP    ( CALCULATE OFFSET TO BEGIN,)
   FF80 U<    ( OUT OF RANGE?)
   IF 3 ERR
   ELSE WR1    ( WRITE OFFSET, 1 BYTE)
   THEN
   ?LIST
; IMMEDIATE

: THEN, DUP ( FLAG) 2 =
   IF ( WAS IF,) DROP THENN ( INSERT OFFSET)
   ELSE 4 ?PAIRS ( SEE IF ELSE, PR"CEE"ED)

   2DUP THENN - SWAP ?RAM+ C! ( IF: SKIP TO ELSE)
   THEN
   ?LIST
; IMMEDIATE

: ELSE, 2 ?PAIRS 20 WR1 ( OPCODE OF BRANCH ALWAYS)
  VLOC 4 ( FLAG) 0 WR1 ?LIST ; IMMEDIATE ( OFFSET 0-DEFLT)

( REM 17)
( ASSEMBLER DIRECTIVES)

: ORG PC !  ( SET PC OF ASSEMBLER)
   CR
   ." WARNING: IF PC EXCEEDS EPROM-ADDRESS" CR OL ;
IMMEDIATE
: .END ?EXEC ?CSP
   -2 ALLOT B792 , , SMUDGE
  OL
; IMMEDIATE

: .FCB WR1 OL ;    ( WRITE 1 BYTE TO MEMORY)
: .BYT WR1 OL ;    ( DITTO)
```

```
: .FDB WR2 OL ;    ( WRITE WORD TO MEMORY)
: .WOR WR2 OL ;    ( DITTO)
: .RMB PC +! OL ;    ( RESERVE MEMORY BYTES)
: " 22 WORD HERE C$ 0 DO HERE 1+ I + C$ WR1 LOOP OL ;
: .FCC " ;    ( SAME WORD)
: SPC CR ;    ( ADVANCE TO NEXT LINE)
: .SPC CR ;
: .EQU <BUILDS , OL DOES> $ ;  ( FORM A CONSTANT)
: EQ .EQU ;
: LABEL .EQU ;
: BYT .BYT ;
: WOR .WOR ;
: RMB .RMB ;



FORTH DEFINITIONS
( REM 18)
: TASK ;

ASS
0 ORG 2000 ASSIGN 2000 150 55 FILL
FINIS
$$$$$$$$$$$$$$$$$$$$
```

Bernhard Kokula, 6800 Mannheim

65C02-Befehle mit dem normalen Assembler

Die bisherigen Assembler für 6502 kennen die neuen Befehle nicht. Die CMOS-CPU kostet nur ca. DM 30,--, die Assembler dafür zwischen 200,- und 400,- DM. Für Leser, die diese Ausgabe nicht tätigen wollen anbei Vorschläge für einen zeitweiligen Behelf. Die Grundidee ist, dem vorhandenen Assembler die neuen Befehle mit der EQUATE-Anweisung in einer symbolischen Form zu erklären. Sie werden dann mit der .BYT-Anweisung in den Speicher geschrieben. Mit einem Trick geht das auch für die 3 Byte langen Verzweigungsbefehle Branch on Bit Set/Reset. - Man kann sich in verkürzter Form ein File schaffen, das man vor dem Assemblieren in den Editor lädt. Dafür ein Vorschlag am Schluß. Hier zunächst Definitionen und Programmierbeispiele:

```
0000            0011 ;                          In den neuen Befehlen bedeutet:
0000            0012 ; Z        ZEROPAGE
0000            0013 ; I        indirekt.       z.B. JMP (ADDR)
0000            0014 ; X        indiziert       z.B. LDA ADDR,X
0000            0015 ; ADDR     absolute Adresse
0000            0016 ; ZADR     Zeropageadresse
0000            0017
0000            0018
-------------------------------------------------------------------------------

0000            0020 ;Erklaerungen damit der Assembler jetzt nicht meutert.
0000            0021 ZADR   =$D0                ;Adresse in der Zeropage
0000            0022 ADDR   =$E123              ;absolute Adresse
0000            0023 MASKE  =%10010111
0000            0024        *=$200              ;
0200            0025
0200            0026
-------------------------------------------------------------------------------
```

```
0200             0028 ;----- 1-Byte-Befehle
0200             0029 DECA   =$3A              ;DEC A
0200             0030 INCA   =$1A              ;INC A
0200             0031 PHX    =$DA
0200             0032 PLX    =$FA
0200             0033 PHY    =$5A
0200             0034 PLY    =$7A
0200  1A         0035        .BYT INCA         ;Beispiel
0201             0036
0201             0037
```

```
0201             0039 ;----- Zeropage-Befehle, (ZP) =ZADR
0201             0040 ADCZI  =$72              ;ADC (ZP)    indirekt
0201             0041 ANDZI  =$32              ;AND (ZP)    indirekt
0201             0042 BITK   =$89              ;BIT #BYTE   immediate
0201             0043 BITZX  =$34              ;BIT ZP,X    X-indiziert
0201             0044 CMPZI  =$D2              ;CMP (ZP)    indirekt
0201             0045 EORZI  =$52              ;EOR (ZP)    indirekt
0201             0046 LDAZI  =$B2              ;LDA (ZP)    indirekt
0201             0047 ORAZI  =$12              ;ORA (ZP)    indirekt
0201             0048 SBCZI  =$F2              ;SBC (ZP)    indirekt
0201             0049 STAZI  =$92              ;STA (ZP)    indirekt
0201             0050 ST0Z   =$64              ;Store ZERO in Zeropage
0201             0051 ST0ZX  =$74              ;Store ZERO Zeropage,X
0201  B2         0052        .BYT LDAZI,ZADR   ;Beispiel
0202  D0         0052
```

```
0203             0056 ;----- Befehle mit absoluter Adresse
0203             0057 BITX   =$3C              ;BIT-test absolut,X
0203             0058 JMPX   =$7C              ;JMP (absolut),X
0203             0059 ST0    =$9C              ;Store ZERO absolut
0203             0060 ST0X   =$9E              ;Store ZERO absolut,X
0203  9E         0061        .BYT ST0X,(ADDR,)ADDR ;so..
0204  23         0061
0205  E1         0061
0206  9E         0062        .BYT ST0X         ;oder so zu schreiben
0207  23 E1      0063        .WOR ADDR
0209             0064
0209             0065
```

```
0209             0067 ;----- 2-BYTE-Branch-Befehl
0209             0068 BRA    =$80              ;Branch always
0209  80         0069        .BYT BRA,ZIEL1-*-1 ;Spruns vorwaerts
020A  01         0069
020B  EA         0070        NOP
020C  EA         0071 ZIEL1  NOP
020D  EA         0072        NOP
020E  80         0073        .BYT BRA,$FF+ZIEL1-* ;Spruns rueckwaerts
020F  FC         0073
0210             0074
```

```
0210             0077 ;----- 3-BYTE-Branch-Befehle zur Bit-Manipulation, ZP
0210             0078 BBR    =$07              ;Branch if in ZP.adr. BITx =0
0210             0079 BBS    =$8F              ;Branch if in ZP.adr. BITx =1
0210             0080 SMB    =$87              ;Set       in ZP.adr. BITx =1
0210             0081 RMB    =$07              ;Reset     in ZP.adr. BITx =0
0210             0082 ;Beispiel, das Ziel-BIT x $10 ist zum Befehl zu addieren
0210  A7         0083        .BYT SMB+$20,ZADR,ZIEL2-*-1 ;vorwaerts (Bit 2)
0211  D0         0083
0212  01         0083
0213  EA         0084        NOP
0214  EA         0085 ZIEL2  NOP
0215  EA         0086        NOP
0216  77         0087        .BYT BBR+$70,ZADR,$FF+ZIEL2-* ;rueckwaerts (Bit 7)
0217  D0         0087
0218  FB         0087
```

```
0219             0091 ;----- 2-BYTE-Befehle zur Bit-Manipulation, ZP
0219             0092 TSBZ   =$04             ;Test &   Set Bit mit Maske im (A)
0219             0093 TRBZ   =$14             ;Test & Reset Bit mit Maske im (A)
0219  A9 97        0094        LDA #MASKE       ;Beispiel
021B  04          0095        .BYT TSBZ,ZADR
021C  D0          0095
021D  F0 F5       0096        BEQ ZIEL2
021F              0097
021F              0098 ;
```

```
021F             0100 ;----- 3-BYTE-Befehle zur Bit-Manipulation, absolut
021F             0101 TSB    =$0C             ;Test &   Set Bit mit Maske im (A)
021F             0102 TRB    =$1C             ;Test & Reset Bit mit Maske im (A)
021F  A9 97      0103        LDA #MASKE       ;Beispiel
0221  0C         0104        .BYT TSB,(ADDR,)ADDR
0222  23         0104
0223  E1         0104
0224  F0 EE      0105        BEQ ZIEL2        ;Ergebnis im ZERO-Flag
0226             0106
0226             0107
```

```
0226             0109 ;----- Ersatz der langsamen Routinen PHXY, PLXY
0226  5A         0110        .BYT PHY,PHX     ; anstelle von JSR PHXY
0227  DA         0110                         .
0228  EA         0111        NOP              ;kann entfallen, ist nur 3.Byte
0229  FA         0112        .BYT PLX,PLY     ; anstelle von JSR PLXY
022A  7A         0112
022B  EA         0113        NOP
022C             0114
022C             0115        .END
 ERRORS=0000
```

```
.PAG '65C02-Befehle mit normalem 6502-Assembler schreiben'
;Kurzform als 'TAPE-MACRO'                    Kokula/1.2.84

;----- 1-Byte-Befehle
DECA =$3A ;DEC A
INCA =$1A ;INC A
PHX =$DA
PLX =$FA
PHY =$5A
PLY =$7A

;----- Zeropage-Befehle, (ZP) =ZADR
ADCZI =$72 ;ADC (ZP)    indirekt
ANDZI =$32 ;AND (ZP)    indirekt
BITK =$89 ;BIT #BYTE   immediate
BITZX =$34 ;BIT ZP,X    X-indiziert
CMPZI =$D2 ;CMP (ZP)    indirekt
EORZI =$52 ;EOR (ZP)    indirekt
LDAZI =$B2 ;LDA (ZP)    indirekt
ORAZI =$12 ;ORA (ZP)    indirekt
SBCZI =$F2 ;SBC (ZP)    indirekt
STAZI =$92 ;STA (ZP)    indirekt
ST0Z =$64 ;Store ZERO in Zeropage
ST0ZX =$74 ;Store ZERO Zeropage,X

;----- Befehle mit absoluter Adresse
BITX =$3C ;BIT-test absolut,X
JMPX =$7C ;JMP (absolut),X
ST0 =$9C ;Store ZERO absolut
ST0X =$9E ;Store ZERO absolut,X

;----- 2-BYTE-Branch-Befehl
BRA =$80 ;Branch always
```

```
;----- 3-BYTE-Branch-Befehle zur Bit-Manipulation, ZP
BBR =$07 ;Branch if in ZP.adr. BITx =0
BBS =$8F ;Branch if in ZP.adr. BITx =1
SMB =$87 ;Set       in ZP.adr. BITx =1
RMB =$07 ;Reset     in ZP.adr. BITx =0
;Das Ziel-BIT x $10 ist zum Befehl zu addieren

;----- 2-BYTE-Befehle zur Bit-Manipulation, ZP
TSBZ =$04 ;Test &   Set Bit mit Maske im (A)
TRBZ =$14 ;Test & Reset Bit mit Maske im (A)

;----- 3-BYTE-Befehle zur Bit-Manipulation, absolut
TSB =$0C ;Test &   Set Bit mit Maske im (A)
TRB =$1C ;Test & Reset Bit mit Maske im (A)
LDA #MASKE ;Beispiel
```

Horst Brettin, 1000 Berlin 44

Simon's BASIC

Ein Erfahrungsbericht

Simon's BASIC ist eine BASIC-Erweiterung für den Commodore 64. Das Handbuch verspricht über 100 neue Befehle. Aber was taugen diese Befehle? - Es wird unterschieden nach Befehlen zur Belegung der Funktionstasten, Programmierhilfen, Hilfen zur Fehlerbeseitigung und zum Listen der Programme. Es gibt einen Programmschutz, Zeichenketten-Operationen, Hilfen zur Zahlenbehandlung und Diskettenbefehle. Sehr viel Grafikunterstützung und eine Einführung in strukturierte Programmierung. Letztendlich unterstützt es Musikprogramme und die Kontrolle der Joystick-Ports. Ein Programm, das sehr viel kann, sollte man meinen. Aber bei näherer Betrachtung entdeckt man die vielen Kompromisse, die gemacht wurden. Viel können heißt auch viel Speicherplatz belegen.

Aber nun zu den neuen Befehlen. Als nützlich haben sich die Befehle für die Funktionstasten herausgestellt. Dem Benutzer stehen 16 (!) frei belegbare Funktionstasten zur Verfügung. Die acht neuen Tasten werden mit der Commodore-Taste bzw. Shift und Commodore-Taste erreicht. Allerdings sollte man eines nicht vergessen: Wer merkt sich die 16 unterschiedlichen Belegungen der F-Tasten? - Als nützlich kann man auch die AUTO-Funktion für die automatische Zeilenumerierung ansehen. Der nächste Befehl, RENUMBER, ist allerdings ein schlechter Witz. Es werden nur die Zeilenummern geändert, aber nicht die Statements. Hier wird auf die Verwendung von Labels verwiesen (weiter hinten im Handbuch). Wenn man mit Simon's BASIC lediglich Programme erstellen möchte, die hinterher im normalen BASIC laufen sollen, kann man diese Funktion vergessen. Möchte man aber bei Simon's bleiben, kann man den nächsten Befehl recht gut gebrauchen. Mit PAUSE kann man definierte Unterbrechungen ins Programm einbauen und kann auf umständliche FOR-NEXT-Schleifen verzichten.

Daß das Simon's BASIC auch über 'MERGE' verfügt, möchte ich der Vollständigkeit wegen erwähnen. Eigentlich ist es nur ein Append, bindet den neu zu ladenden Programmteil also hinten an das Programm an. Die Listhilfen sind relativ praktisch. Man kann ein Listing in einzelne Seiten unterteilen. Wer aber schon einmal mit dem sog. Scrolling gearbeitet hat, findet diese Methode umständlich. Beim Scrolling kann das Listing aufwärts und abwärts gerollt werden. - Weitere Befehle dienen zur Verlangsamung der Listgeschwindigkeit und zum Hervorheben der neuen Simon's-Befehle. Auch die Funktion FIND zum Auffinden bestimmter Begriffe ist vorhanden.

Programmschutz: Sinn oder Unsinn bei privater Nutzung? Darüber kann man lange streiten. Der Programmschutz ist leider nicht aufzuheben. Sollte sich später ein Fehler im Programm zeigen, so ist man auf erneutes Eintippen der Zeilen angewiesen. Ebenso fragwürdig sind die Zeichenketten-Operationen. Es läßt sich alles, wenn auch mit mehr Aufwand, im normalen BASIC realisieren. Das trifft auch auf die Befehle der Tastaturabfrage zu.

Das nächste Kapitel im Handbuch beschäftigt sich mit der Zahlenbehandlung. Dieser Abschnitt beinhaltet sechs Befehle. Auf ihre Vor- und Nachteile einzgehen lohnt nicht. - Für die Diskettenbefehle habe ich schon einfachere Varianten gesehen. Der DOS-Support auf der Dienstdiskette zur VC-1541 ist wesentlich einfacher und eleganter.

Nun nähern wir uns der großen Stärke des Simon's BASIC: Der Grafik. Wer schon einmal versucht hat, eine hochauflösende Grafik in BASIC zu erstellen oder wer mit Sprites spielen wollte, hat das umständliche Verfahren und die langsame Ausführung verdammt. Mit wenigen Befehlen, die übrigens mit sehr guten Beispielen dokumentiert sind, kann man mit Simon's gute Grafiken erstellen. Ist man mit dem Ergebnis zufrieden, dann kann man die Grafik auf Diskette abspeichern oder auf den Drucker ausgeben. Dier Herstellung von Sprites, hier MOB (moveable object block) genannt, ist denkbar einfach. Ebenso einfach ist es, die MOBs zu bewegen, ihre Kollision zu überwachen, ihre Farbe zu ändern etc.. Es wundert nicht, daß das Kapitel Grafik im Handbuch den meisten Platz einnimmt. Es ist auch mit wenigen Befehlen möglich, einzelne Zeichen neu zu gestalten.

Der nächste Abschnitt im Handbuch geht auf strukturierte Programmierung ein. Es wird die Bedingung ELSE als Ergänzung zu IF .. THEN-Verknüpfungen angeboten. REPEAT .. UNTIL können FOR-NEXT-Schleifen ersetzen. Es folgen dann noch weitere Befehle für logische Verknüpfungen. Schließlich wird die Möglichkeit geboten, auf Labels zurückzugreifen. Es wird ein Unterprogramm mit einem Namen versehen und bei Bedarf aufgerufen. Die Befehle GOTO bzw. GOSUB kann man vergessen wenn man Labels anwendet. In diesem Fall kann man auch die schon erwähnte Renumber-Funktion voll nutzen, es muß ja keine neue Sprungweite berechnet werden. Ein weiterer Punkt ist die Fehlerbehandlung. Es ist möglich, ein mit Fehlern behaftetes Programm bis zum Ende laufen zu lassen. Nach Beendigung des Programms werden dann die Fehler ausgegeben. Diese Routine kann durchaus nützlich sein. ON ERROR .. GOTO zum Abfangen von Fehlern ist auch vorhanden.

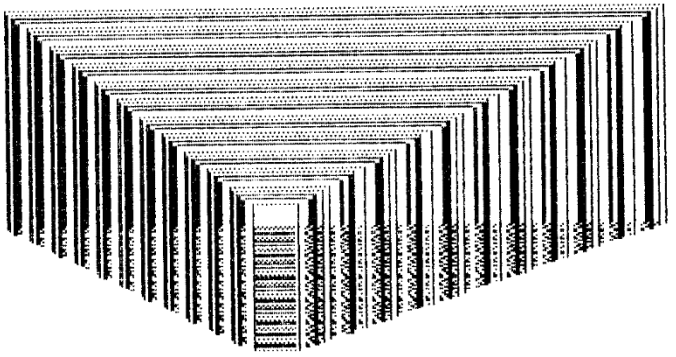
Im vorletzten Kapitel des Handbuches wird auf die Erzeugung von Musik eingegangen. Mit wenigen Befehlen kann ein Ton definiert werden. Dann kann man mehrere Stimmen synchronisieren und die leicht einzugebenden Noten abspielen lassen. Die Macher des Handbuches haben sich leider nur wenig Mühe gegeben, um diese Stärke ihres Programms hervorzuheben. Es fehlen Beispiele, wie man unterschiedliche Instrumente imitieren kann. Das Handbuch des C-64 ist auch keine große Hilfe; man muß also probieren.
Im letzten Abschnitt wird die Abfrage der Joystick-Ports erläutert. Einfacher geht es fast nicht mehr. Die umständliche PEEKerei entfällt, dafür werden einleuchtende Variablen-Namen benutz, um Joystick, Paddle oder sogar Lightpen abzufragen.

Schlußbetrachtung: Simon's BASIC hat seine großen Stärken in der Grafik, der Musik und der Portabfrage. Die weiteren Befehle sind leicht zu ersetzen oder sie bieten nur Halbheiten an. Sie stellen eindeutig einen Kompromiß zwischen Speicherplatz und Funktion dar. Die Funktion ist dabei leider etwas zu kurz gekommen. Wer sich dieses BASIC zulegen möchte, sollte vorher überlegen, ob er nicht lieber spezielle Programme vorziehen sollte. Grafikprogramme gibt es mit der Zeit auch für weniger Geld. Sie können ebensoviel und sparen sogar noch Speicherplatz ein. Reine BASIC-Erweiterungen, wie z.B. EXBASIC Level II, können auf anderen Gebieten mehr. Leider sind sie nicht so preiswert wie Simon's BASIC. Die Vertreiber dieser Software sollten überlegen, ob sie den Gewinn nicht lieber über die Menge machen wollen. Zufriedene Hobbyisten würden es ihnen danken, denn sonst kommen sie um ein Universalprogramm wie Simon's BASIC nicht herum.

Nachstehend einige Programmbeispiele und die von ihnen erzeugten Grafiken.

```
100 POKE53280,11
110 HIRES 1,11:MULTI 12,8,6
120 FORX=5TO60STEP1
130 F=F+1
140 IFF=5THENF=0
150 REC X,1,9*X,160-5/2*X,120-9*X/13,F
```

```
160 NEXT X
170 LOW COL 3,7,13      HIRES=HIGH RESOLUTION
180 COPY:END            MULTI=MULTI-COLOR-MODUS
                        REC=RECHTECK ZEICHNEN
                        LOW COL=FARBWECHSEL
READY.                  COPY=HIRES-BILDSCHIRM AUF DRUCKER SENDEN
```



Ing. (grad.) Wolfgang Krebs, 1000 Berlin 41

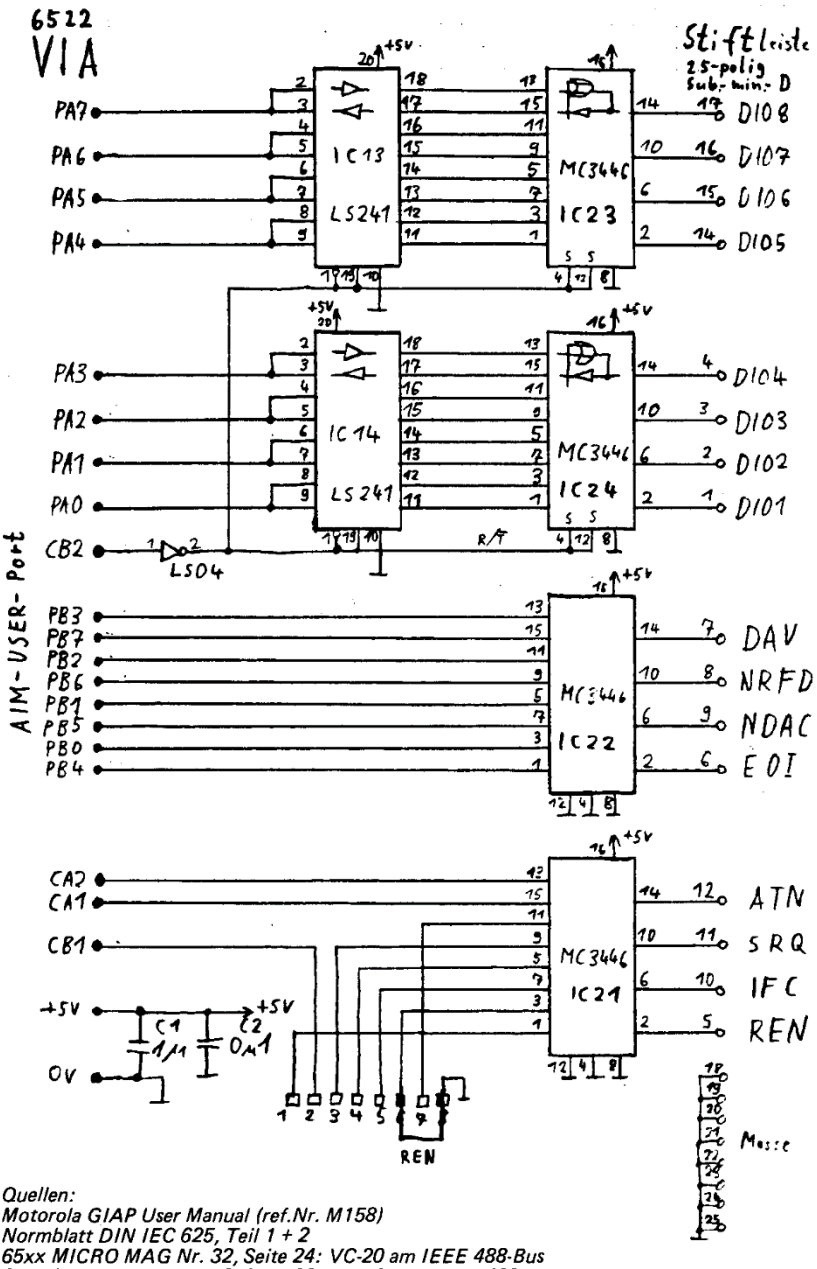
IEC-625-Bus Controllerroutinen

auf dem AIM 65

Das Programm realisiert die wichtigsten Befehle und Controllerfunktionen für den IEC-625-Bus. Zwischen dem AIM Userport und dem IEC-Bus muß eine Treiberschaltung installiert werden, wie sie in beifolgender Skizze vorgeschlagen ist. Grundlage des Programmes war dasjenige im 65xx MICRO MAG Nr. 19, Seite 38.

Das Programm besteht aus drei Teilen. Je ein Treiber über UIN und UOUT, sowie ein Menueprogramm, von dem aus die Befehle für den IEC-Bus aufgerufen werden. Nach dem Starten mit 'N' (weil hier für den Adressenbereich des Assemblers programmiert wurde) verlangt das Programm die IEC-Bus Adresse, listet die Befehle aus und geht in die Hauptkommandoschleife. Als Prompt erscheint ein '#'. Wenn der Printer eingeschaltet war, werden die möglichen Befehle ausgedruckt. Wir haben folgende Befehle:

Hex	*Befehl*		*verwendet*
Adressierte Befehle			
01	*GTL*	*Go To Local*	*ja*
04	*SDC*	*Selected Device Clear*	*ja*
08	*GET*	*Group Executive Trigger*	*ja*
05	*PPC*	*Parallel Poll Configure*	*nein*
09	*TCT*	*Take Control*	*nein*
Universalbefehle			
11	*LLO*	*Local Lockout*	*ja*
14	*DCL*	*Device Clear*	*ja*
15	*PPU*	*Parallel Poll Unconfigure*	*nein*
18	*SPE*	*Serial Poll Enable*	*ja*
19	*SPD*	*Serial Poll Disable*	*ja*



Quellen:
Motorola GIAP User Manual (ref.Nr. M158)
Normblatt DIN IEC 625, Teil 1 + 2
65xx MICRO MAG Nr. 32, Seite 24: VC-20 am IEEE 488-Bus
65xx MICRO MAG Nr. 19, Seite 38: AIM 65 am IEEE 488-Bus.

```
0005  0000                ;**********************************
0006  0000                ;
0007  0000                ;   IEC-BUS-TREIBER
0008  0000                ;
0009  0000                ;VERSION FUER DEN AIM 65
0010  0000                ; NUR KONTROLLER
0011  0000                ;
0012  0000                ;**********************************
0013  0000                ;**********************************
0014  0000                ;
0015  0000                IECADR =$190            ;SPEICHER FUER IEC-BUS-ADRESSE
0016  0000                ;
0017  0000                       * =$A000         ;AIM65 USER-VIA
0018  A000                PORTB  =*
0019  A000                PORTA  =*+1
0020  A000                DDRB   =*+2
0021  A000                DDRA   =*+3
0022  A000                ACR    =*+11
0023  A000                PCR    =*+12
0024  A000                IFR    =*+14
0025  A000                IER    =*+15
0026  A000                ;
0027  A000                ;-------------------------------
0028  A000                ; KONSTANTEN :
0029  A000                WRITE  =$E0
0030  A000                READ   =$C0
0031  A000                LF     =$0A
0032  A000                CR     =$0D
0033  A000                CA1EN  =$82
0034  A000                RISE   =1
0035  A000                FALL   =0
0036  A000                ATNHI  =$0E
0037  A000                ATNLOW =$0C
0038  A000                ;
0039  A000                ;-------------------------------
0040  A000                ; UP'S AIM 65 :
0041  A000                OUTPUT =$E97A
0042  A000                RCHEK  =$E907
0043  A000                REDOUT =$E973
0044  A000                CRLF   =$E9F0
0045  A000                BLANK  =$E83E
0046  A000                COMIN  =$E1A1           ;AIM65-MONITOR
0047  A000                NUMA   =$EA46
0048  A000                ADDIN  =$EAAE
0049  A000                ADDR   =$A41C
0050  A000                ;
0051  A000                       *=$108           ;AIM65 BENUTZER-TREIBER-VECTOREN
0052  0108  5D D0         UIN    .WOR UIN1
0053  010A  C9 D0         UOUT   .WOR UOUT1
0054  010C                       * =$10C          ;AIM65 F-TASTEN
0055  010C  4C 19 D3      FKEYS  JMP START        ;F1
0056  010F  4C 03 D0             JMP INITI        ;F2
0057  0112  4C 19 D3             JMP START        ;F3
0058  0115                STAR2=START
0059  0115                ;
0060  0115                ;**********************************
0063  0115                ;DAV  D7  D3      8
0064  0115                ;NRFD D6  D2      4
0065  0115                ;NDAC D5  D1      2
0066  0115                ;EOI  D4  D0      1
0067  0115                ;
0068  0115                ;**********************************
0069  0115                ;
0070  0115                       *=$D000          ;FUER EPROM-VERSION
0071  D000  4C 19 D3             JMP START
0072  D003                ;**********************************
0073  D003                ;
0074  D003                ;INITIALISIERE AIM 65 VECTOREN
0075  D003  20 09 D0      INITI  JSR INIVEC
0076  D006  4C A1 E1             JMP COMIN
0077  D009                ;
```

```
0078  D009  A9 9F        INIVEC LDA #<USINIT     ;VECTOR OF USER OUTPUT
0079  D00B  8D 0A 01             STA UOUT
0080  D00E  A9 D0              LDA #>USINIT
0081  D010  8D 0B 01             STA UOUT+1
0082  D013  A9 5D              LDA #<UIN1
0083  D015  8D 08 01             STA UIN
0084  D018  A9 D0              LDA #>UIN1
0085  D01A  8D 09 01             STA UIN+1
0086  D01D  A9 4C              LDA #$4C        ;JMP
0087  D01F  8D 0C 01             STA FKEYS
0088  D022  8D 0F 01             STA FKEYS+3
0089  D025  8D 12 01             STA FKEYS+6
0090  D028  A9 19              LDA #<STAR2
0091  D02A  8D 0D 01             STA FKEYS+1
0092  D02D  A9 D3              LDA #>STAR2
0093  D02F  8D 0E 01             STA FKEYS+2
0094  D032  A9 03              LDA #<INITI
0095  D034  8D 10 01             STA FKEYS+4
0096  D037  A9 D0              LDA #>INITI
0097  D039  8D 11 01             STA FKEYS+5
0098  D03C  A9 19              LDA #<START
0099  D03E  8D 13 01             STA FKEYS+7
0100  D041  A9 D3              LDA #>START
0101  D043  8D 14 01             STA FKEYS+8
0102  D046  20 5F D2             JSR GETADR      ;HOHLE IEC-BUS-ADR.
0103  D049  60                  RTS
0104  D04A                  ;
0105  D04A                  ;*******************************
0106  D04A                  ;
0107  D04A                  ; EMPFANGE VOM IEC-625-BUS
0108  D04A                  ; EIN ANDERES GERAET WIRD ALS TALKER ADRESSIERT
0109  D04A                  ;
0110  D04A                  ; LISTEN UND EMPFANG UP'S
0111  D04A                  ;
0112  D04A  AD 90 01  UINI1  LDA IECADR      ;IEC-BUS ADRESSE
0113  D04D  09 40             ORA #$40        ;TALKER
0114  D04F  20 A6 D0           JSR SMATNL      ;SENDE MIT ATN LOW
0115  D052  AD 00 A0           LDA PORTB       ;NOCH NICHT BEREIT
0116  D055  09 09              ORA #$09
0117  D057  29 F9              AND #$F9        ;NRFD+NDAC LOW
0118  D059  8D 00 A0           STA PORTB
0119  D05C  60                 RTS
0120  D05D                  ;
0121  D05D  90 EB        UIN1   BCC UINI1       ;INITIALISIEREN WENN C=0
0122  D05F  AD 00 A0     REC    LDA PORTB       ;ZUR DATENUEBERNAHME BEREIT
0123  D062  09 04              ORA #$04        ;-> NRFD HIGH
0124  D064  8D 00 A0           STA PORTB
0125  D067  AD 00 A0     REC1   LDA PORTB
0126  D06A  48                 PHA
0127  D06B  20 07 E9           JSR RCHEK       ;BREAK VON TASTATUR ?
0128  D06E  68                 PLA
0129  D06F  29 80              AND #$80
0130  D071  D0 F4              BNE REC1        ;DAV NOCH HIGH ?
0131  D073                  ;
0132  D073  AD 00 A0           LDA PORTB       ;NRFD LOW
0133  D076  29 FB              AND #$FB
0134  D078  8D 00 A0           STA PORTB
0135  D07B  A9 CE              LDA #READ+ATNHI ;LESEN
0136  D07D  8D 0C A0           STA PCR
0137  D080  AD 01 A0           LDA PORTA
0138  D083  49 FF              EOR #$FF
0139  D085  48                 PHA             ;EMPFANGENES DATEN-BYTE
0140  D086  AD 00 A0           LDA PORTB
0141  D089  09 02              ORA #$02        ;NDAC HIGH
0142  D08B  8D 00 A0           STA PORTB
0143  D08E                  REC2
0144  D08E  AD 00 A0           LDA PORTB
0145  D091  29 80              AND #$80
0146  D093  F0 F9              BEQ REC2        ;DAV NOCH LOW ?
0147  D095  AD 00 A0           LDA PORTB
```

```
0148  D098  29 F9        AND #$F9          ;NDAC LOW
0149  D09A  8D 00 A0     STA PORTB
0150  D09D  68           PLA               ;DATEN-BYTE
0151  D09E  60           RTS
0152  D09F              ;
0153  D09F              ;*********************************
0154  D09F              ;
0155  D09F              ; SENDE AUF DEN IEC-625-BUS
0156  D09F              ; DAS ANDERE GERAET WIRD ALS LISTENER ADRESSIERT
0157  D09F              ;
0158  D09F  B0 28       USINIT BCS USOUT        ;C=1 -> EIN BYTE AUSGEBEN
0159  D0A1              UOUTI1                  ;INITIALISIERE LISTENER
0160  D0A1  AD 90 01    LISADR LDA IECADR       ;IEC-BUS-ADRESSE
0161  D0A4  09 20        ORA #$20          ;LISTENER
0162  D0A6  48           SMATNL PHA              ;SENDE MIT ATN LOW, BYTE IN
0163  D0A7  A9 0F        LDA #$0F          ;GEBE ADRESSE AUF IEC-BUS
0164  D0A9  8D 02 A0     STA DDRB          ;DATA DIRECTION-REGISTER
0165  D0AC  8D 00 A0     STA PORTB         ;ALLE LEITUNGEN HIGH
0166  D0AF  A9 EC        LDA #WRITE+ATNLOW
0167  D0B1  8D 0C A0     STA PCR
0168  D0B4  A9 FF        LDA #$FF          ;ALLE BITS AUSGAENGE
0169  D0B6  8D 03 A0     STA DDRA          ;FUER DATEN-BUS
0170  D0B9  8D 01 A0     STA PORTA         ;OUTPUT
0171  D0BC  20 0A D1     JSR SAMPLE        ;TESTE OB BUS FREI
0172  D0BF  68           PLA
0173  D0C0  20 D0 D0     JSR SEND
0174  D0C3  A9 EE        LDA #WRITE+ATNHI
0175  D0C5  8D 0C A0     STA PCR           ;ATN HIGH
0176  D0C8  60           RTS
0177  D0C9              ;
0178  D0C9              UOUT1
0179  D0C9  68           USOUT  PLA              ;HOHLE BYTE
0180  D0CA  48           PHA
0181  D0CB  20 D0 D0     US0    JSR SEND
0182  D0CE  68           PLA
0183  D0CF  60           RTS
0187  D0D0              ;TALKER, UP ZUM SENDEN
0188  D0D0              ;
0189  D0D0  49 FF       SEND   EOR #$FF
0190  D0D2  8D 01 A0     STA PORTA         ;SCHREIBE EIN BYTE ZUM BUS
0191  D0D5  AD 00 A0     LDA PORTB
0192  D0D8  09 06        ORA #$06          ;NRFD+NDAC NICHT VOM KONTROLLER
0193  D0DA  8D 00 A0     STA PORTB         ;LOW
0194  D0DD              NRFD0
0195  D0DD  AD 00 A0     LDA PORTB
0196  D0E0  29 40        AND #$40          ;NRFD
0197  D0E2  F0 F9        BEQ NRFD0         ;WARTE BIS NRFD HIGH
0198  D0E4  AD 00 A0     LDA PORTB
0199  D0E7  29 F1        AND #$F1          ;EOI UNVERAENDERT
0200  D0E9  09 06        ORA #$06          ;DAV LOW
0201  D0EB  8D 00 A0     STA PORTB
0202  D0EE              ;
0203  D0EE  AD 00 A0     SEND1  LDA PORTB
0204  D0F1  29 60        AND #$60
0205  D0F3  C9 20        CMP #$20          ;DAV+NRFD SIND NOCH LOW
0206  D0F5  D0 F7        BNE SEND1         ;WARTE AUF NRDF=LOW UND NDAC=HIGH
0207  D0F7  A9 0F        LDA #$0F          ;ALLE HIGH
0208  D0F9  8D 00 A0     STA PORTB
0209  D0FC  A9 FF        LDA #$FF
0210  D0FE  8D 01 A0     STA PORTA         ;DATEN-LEITUNGEN HIGH
0211  D101  20 0A D1     JSR SAMPLE
0212  D104  A9 0B        LDA #$0B          ;NRFD LOW
0213  D106  8D 00 A0     STA PORTB
0214  D109  60           RTS
0215  D10A              ;
0216  D10A              ;IST EIN GERAET AM BUS ANGESCHLOSSEN ?
0217  D10A  AD 00 A0     SAMPLE LDA PORTB
0218  D10D  29 60        AND #$60          ;NRFD+NDAC
0219  D10F  C9 60        CMP #$60          ;BEIDE HIGH?
0220  D111  F0 01        BEQ ERROR
0221  D113  60           RTS
```

```
0223  D114  68          ERROR   PLA
0224  D115  68                  PLA
0225  D116  A2 00              LDX #$00          ;FEHLERBEHANDLUNG
0226  D118  20 1E D1          JSR STROUT        ;FEHLERMELDUNG
0227  D11B  4C 1F D3          JMP HLOOP
0228  D11E                    ;
0229  D11E                    ;-------------------------------
0230  D11E                    ;TEXT-AUSGABE
0231  D11E                    ;
0232  D11E  20 F0 E9          STROUT JSR CRLF
0233  D121  BD 46 D1          STRO1  LDA TEXT,X
0234  D124  30 0A                    BMI SOEND
0235  D126  C9 0D                    CMP #CR           ;CR ?
0236  D128  F0 06                    BEQ SOEND
0237  D12A  20 7A E9                 JSR OUTPUT
0238  D12D  E8                       INX
0239  D12E  D0 F1                    BNE STRO1
0240  D130                    ;
0241  D130  E8                SOEND  INX
0242  D131  60                       RTS
0243  D132  20 F0 E9          STOUT1 JSR CRLF
0244  D135  BD 43 D2          STO1   LDA M11,X
0245  D138  30 F6                    BMI SOEND
0246  D13A  C9 0D                    CMP #CR
0247  D13C  F0 F2                    BEQ SOEND
0248  D13E  20 7A E9                 JSR OUTPUT
0249  D141  E8                       INX
0250  D142  D0 F1                    BNE STO1
0251  D144  E8                       INX
0252  D145  60                       RTS
0253  D146                    ;
0254  D146                    TEXT
0255  D146  20 45             M0     .BYT ' ERROR KEIN GERAET ',$0D
0255  D159  0D
0256  D15A  20 49             M1     .BYT ' IEC-BUS-ADRESSE(HEX)',$80
0256  D16F  80
0257  D170  49 45             M2     .BYT 'IEC-BUS-TREIBER-BEFEHLE :',$0D
0257  D189  0D
0258  D18A  41 20             M2A    .BYT 'A =NEUE IEC-BUS-ADRESSE',$0D
0258  D1A1  0D
0259  D1A2  4C 20             M2B    .BYT 'L =LISTEN',$0D
0259  D1AB  0D
0260  D1AC  54 20             M3     .BYT 'T =TALK',$0D
0260  D1B3  0D
0261  D1B4  47 20             M4     .BYT 'G =GET (TRIGGER)',$0D
0261  D1C4  0D
0262  D1C5  4D 20             M5     .BYT 'M =GTL, GO TO LOKAL',$0D
0262  D1D8  0D
0263  D1D9  55 20             M6     .BYT 'U =UNLISTEN,UNTALK',$0D
0263  D1EB  0D
0264  D1EC  53 20             M7     .BYT 'S =SERIAL POLLING',$0D
0264  D1FD  0D
0265  D1FE  4F 20             M8     .BYT 'O =LOCAL LOCK OUT',$0D
0265  D20F  0D
0266  D210  52 20             M9     .BYT 'R =REMOTE',$0D
0266  D219  0D
0267  D21A  44 20             M10    .BYT 'D =DEVICE CLEAR',$0D
0267  D229  0D
0268  D22A  43 20             M10A   .BYT 'C =SELEKTIV DEVICE CLEAR',$0D
0268  D242  0D
0269  D243  50 20             M11    .BYT 'P =PARALLEL POLLING',$0D
0269  D256  0D
0270  D257  48 20             M12    .BYT 'H =HELP',$0D
0270  D25E  0D
0271  D25F                    ;
0272  D25F                    ;-------------------------------
0273  D25F                    ; HOHLE IEC-BUS ADRESSE VON TASTATUR
0274  D25F                    ;
0275  D25F  A2 14             GETADR LDX #M1-M0
0276  D261  20 1E D1                 JSR STROUT
0277  D264  20 AE EA                 JSR ADDIN
```

```
0278  D267  AD 1C A4          LDA ADDR
0279  D26A  29 0F             AND #$0F
0280  D26C  8D 90 01          STA IECADR
0281  D26F  60                RTS
0282  D270                   ;
0283  D270                   ;------------------------------
0284  D270                   ;
0285  D270                   ; HELP-LISTEN DER BEFEHLE
0286  D270                   HELP
0287  D270  A2 14             LDX #M1-M0
0288  D272  20 1E D1          JSR STROUT
0289  D275  A9 3D             LDA #'=
0290  D277  20 7A E9          JSR OUTPUT
0291  D27A  A9 24             LDA #'$
0292  D27C  20 7A E9          JSR OUTPUT
0293  D27F  AD 90 01          LDA IECADR
0294  D282  20 46 EA          JSR NUMA
0295  D285  20 F0 E9          JSR CRLF
0296  D288  A0 0B             LDY #11
0297  D28A  A2 2A             LDX #M2-M0
0298  D28C  20 1E D1    HELP1 JSR STROUT
0299  D28F  88                DEY
0300  D290  10 FA             BPL HELP1
0301  D292  A2 00             LDX #00          ;M11
0302  D294  20 32 D1          JSR STOUT1
0303  D297  A2 14             LDX #M12-M11
0304  D299  20 32 D1          JSR STOUT1
0305  D29C  20 F0 E9          JSR CRLF
0306  D29F  60                RTS
0307  D2A0                   ;
0308  D2A0                   ;******************************
0309  D2A0                   ;
0310  D2A0                   ;UP ADRESSIERTES KOMMANDO
0311  D2A0  48                ADKOM  PHA
0312  D2A1  20 A1 D0          JSR LISADR
0313  D2A4  68                PLA
0314  D2A5  20 A6 D0          JSR SMATNL       ;KOMMANDO
0315  D2A8  4C 67 D3          JMP UNLIS        ;UNLISTEN
0316  D2AB                   ;.............................
0317  D2AB  AD 90 01    LI    LDA IECADR       ;LISTENER
0318  D2AE  09 20             ORA #$20
0319  D2B0  20 A6 D0          JSR SMATNL
0320  D2B3  A9 3A             LDA #':
0321  D2B5  20 7A E9          JSR OUTPUT
0322  D2B8  20 73 E9    LINEXT JSR REDOUT       ;HOHLE ZEICHEN
0323  D2BB  C9 0D             CMP #$0D         ;<CR> ?
0324  D2BD  F0 06             BEQ LIEND
0325  D2BF  20 D0 D0          JSR SEND
0326  D2C2  4C B8 D2          JMP LINEXT
0327  D2C5  20 D0 D0    LIEND JSR SEND
0328  D2C8  20 67 D3          JSR UNLIS
0329  D2CB  4C 1F D3          JMP HLOOP
0330  D2CE                   ;
0331  D2CE  AD 90 01    TA    LDA IECADR       ;ADRESSIEREN ALS TALKER
0332  D2D1  09 40             ORA #$40         ;TALKER
0333  D2D3  20 A6 D0          JSR SMATNL
0334  D2D6  AD 00 A0          LDA PORTB        ;NOCH NICHT BEREIT
0335  D2D9  09 09             ORA #$09
0336  D2DB  29 F9             AND #$F9         ;NRFD+NDAC LOW
0337  D2DD  8D 00 A0          STA PORTB
0338  D2E0  20 5F D0    TINEXT JSR REC          ;EMPFANGE BYTE
0339  D2E3  48                PHA
0340  D2E4  20 7A E9          JSR OUTPUT
0341  D2E7  68                PLA
0342  D2E8  29 7F             AND #$7F
0343  D2EA  C9 0D             CMP #$0D         ;<CR>
0344  D2EC  F0 04             BEQ TAEND
0345  D2EE  C9 0A             CMP #$0A         ;<LF>
0346  D2F0  D0 EE             BNE TINEXT
0347  D2F2  20 6D D3    TAEND JSR UNTALK       ;UNTALK
0348  D2F5  4C 1F D3          JMP HLOOP        ;ZURUECK
```

```
0349  D2F8                    ;
0350  D2F8  A9 08       GET   LDA #$08        ;TRIGGER
0351  D2FA  20 A0 D2          JSR ADKOM
0352  D2FD  4C 1F D3          JMP HLOOP
0353  D300                    ;
0354  D300  A9 01       GTL   LDA #$01        ;GO TO LOKAL
0355  D302  20 A0 D2          JSR ADKOM
0356  D305  4C 1F D3          JMP HLOOP
0357  D308                    ;
0358  D308  A9 11       LLO   LDA #$11        ;LOKAL LOCK OUT
0359  D30A  20 A0 D2          JSR ADKOM
0360  D30D  4C 1F D3          JMP HLOOP
0361  D310                    ;
0362  D310  20 70 D2     HEL   JSR HELP
0363  D313  4C 1F D3          JMP HLOOP
0364  D316                    ;
0365  D316  4C AB D2     LI1   JMP LI
0366  D319                    ;-------------------------------
0367  D319  20 09 D0     START JSR INIVEC      ;INITIALISIERE + IEC-BUS-ADR.
0368  D31C  20 70 D2          JSR HELP         ;HELP-LISTE ANZEIGEN
0369  D31F  20 F0 E9     HLOOP JSR CRLF        ;HAUPT-KOMMANDO-SCHLEIFE
0370  D322  A9 23              LDA #'#
0371  D324  20 7A E9          JSR OUTPUT
0372  D327  20 73 E9          JSR REDOUT       ;BEFEHL?
0373  D32A                    ;SPRUNGVERTEILER ZU DEN BEFEHLEN
0374  D32A  C9 54              CMP #'T
0375  D32C  F0 A0              BEQ TA
0376  D32E  C9 4C              CMP #'L
0377  D330  F0 E4              BEQ LI1
0378  D332  C9 47              CMP #'G
0379  D334  F0 C2              BEQ GET
0380  D336  C9 4D              CMP #'M
0381  D338  F0 C6              BEQ GTL
0382  D33A  C9 55              CMP #'U
0383  D33C  F0 35              BEQ UN
0384  D33E  C9 53              CMP #'S
0385  D340  F0 3A              BEQ SPOLL
0386  D342  C9 4F              CMP #'O
0387  D344  F0 C2              BEQ LLO
0388  D346  C9 52              CMP #'R
0389  D348  F0 72              BEQ REM
0390  D34A  C9 44              CMP #'D
0391  D34C  F0 66              BEQ DCL
0392  D34E  C9 43              CMP #'C
0393  D350  F0 54              BEQ SDC
0394  D352  C9 48              CMP #'H
0395  D354  F0 BA              BEQ HEL
0396  D356  C9 41              CMP #'A
0397  D358  F0 0A              BEQ ST1
0398  D35A  C9 50              CMP #'P
0399  D35C  F0 6C              BEQ PPOLL
0400  D35E                    ;
0401  D35E  20 F0 E9          JSR CRLF         ;DISPLAY CLEAR
0402  D361  4C 1F D3          JMP HLOOP
0403  D364                    ;
0404  D364  4C EF D3     ST1   JMP START1
0405  D367                    ;
0406  D367  A9 3F       UNLIS LDA #$3F        ;UNLISTEN
0407  D369  20 A6 D0          JSR SMATNL
0408  D36C  60                RTS
0409  D36D                    ;
0410  D36D  A9 5F       UNTALK LDA #$5F       ;UNTALK
0411  D36F  20 A6 D0          JSR SMATNL
0412  D372  60                RTS
0413  D373                    ;
0414  D373  20 67 D3     UN    JSR UNLIS
0415  D376  20 6D D3          JSR UNTALK
0416  D379  4C 1F D3          JMP HLOOP
0417  D37C                    ;
```

```
0418  D37C                  SPOLL                   ;SERIAL POLLING
0419  D37C  A9 18                  LDA #$18        ;SERIAL POLL ENABLE
0420  D37E  20 A6 D0              JSR SMATNL
0421  D381  AD 90 01              LDA IECADR      ;ABFRAGE MIT ADRESSIEREN ALS TALK
ER
0422  D384  09 40                  ORA #$40        ;MY TALKER ADRESS
0423  D386  20 A6 D0              JSR SMATNL
0424  D389  AD 00 A0              LDA PORTB       ;NOCH NICHT BEREIT
0425  D38C  09 09                  ORA #$09
0426  D38E  29 F9                  AND #$F9        ;NRFD+NDAC LOW
0427  D390  8D 00 A0              STA PORTB
0428  D393  20 5F D0              JSR REC         ;EMPFANGE STATUS BYTE
0429  D396  20 46 EA              JSR NUMA
0430  D399                  ;
0431  D399  A9 19                  LDA #$19        ;SERIAL POLL DISABLE
0432  D39B  20 A6 D0              JSR SMATNL
0433  D39E  A9 5F                  LDA #$5F        ;UNTALK
0434  D3A0  20 A6 D0              JSR SMATNL
0435  D3A3  4C 1F D3              JMP HLOOP
0436  D3A6                  ;
0437  D3A6  20 A1 D0        SDC   JSR LISADR      ;SELEKTIV DEVICE CLEAR
0438  D3A9  A9 04                  LDA #04
0439  D3AB  20 A6 D0              JSR SMATNL
0440  D3AE  20 67 D3              JSR UNLIS
0441  D3B1  4C 1F D3              JMP HLOOP
0442  D3B4                  ;
0443  D3B4  A9 14           DCL   LDA #$14        ;DEVICE CLEAR
0444  D3B6  20 A6 D0              JSR SMATNL
0445  D3B9  4C 1F D3              JMP HLOOP
0446  D3BC                  ;
0447  D3BC  AD 90 01        REM   LDA IECADR      ;REMOTE
0448  D3BF  09 20                  ORA #$20        ;LISTEN
0449  D3C1  20 A6 D0              JSR SMATNL
0450  D3C4  20 67 D3              JSR UNLIS       ;UNLISTEN
0451  D3C7  4C 1F D3              JMP HLOOP
0452  D3CA                  ;
0453  D3CA                  PPOLL                   ;PARALLEL POLLING
0454  D3CA  A9 EC                  LDA #WRITE+ATNLOW
0455  D3CC  8D 0C A0              STA PCR
0456  D3CF  AD 00 A0              LDA PORTB       ;NOCH NICHT BEREIT
0457  D3D2  09 08                  ORA #$08
0458  D3D4  29 F8                  AND #$F8        ;NRFD+NDAC+EOI LOW
0459  D3D6  8D 00 A0              STA PORTB
0460  D3D9  20 5F D0              JSR REC
0461  D3DC  20 46 EA              JSR NUMA
0462  D3DF  A9 EE                  LDA #WRITE+ATNHI
0463  D3E1  8D 0C A0              STA PCR         ;ATN HIGH
0464  D3E4  AD 00 A0              LDA PORTB
0465  D3E7  09 01                  ORA #$01
0466  D3E9  8D 00 A0              STA PORTB       ;EOI HIGH
0467  D3EC  4C 1F D3              JMP HLOOP
0468  D3EF                  ;
0469  D3EF                  ;-------------------------------
0470  D3EF  20 5F D2        START1 JSR GETADR     ;NEUE IEC-BUS-ADRESSE EINGEBEN
0471  D3F2  4C 1F D3              JMP HLOOP       ;UND ZURUECK
0472  D3F5                  ;
0473  D3F5                        .END
```

Editorial

Für die begleitenden Zeilen des Herausgebers hat dieses Heft kaum noch Platz, daher die Kürze. Mit dem hier zur Sicherheit des Lesers mit allen sonstigen Änderungen abgedruckten DISKMON wird der AIM 65 weiter veredelt: Es kann eine preiswerte handelsübliche Floppy als Systemeinheit betrieben werden. Die IEC-Bus-Routinen erlauben eine individuelle Anpassung und die Übertragung des Lösungsweges auch für andere Rechner, die mit einer CBM 1541 betrieben werden sollen. - Auch für die Programmierung der CMOS-CPU stehen mit den Vorschlägen in diesem Heft neue Hilfsmittel zur Verfügung, nicht zuletzt auch mit den jetzt lieferbaren 2-Pass Assemblern für AIM 65 (Löhr) und CBM (Porbadnigk), siehe Anzeigenteil.

Peter Engels, 5308 Rheinbach

APPLE-MATH

Für alle APPLE-Rechner und kompatible werden nachfolgend 4 Mathematik-Fuktionen als Maschinenprogramm abgedruckt, die im normalen BASIC nicht vorhanden sind. Es handelt sich dabei um:

1.	*Zehnerlogarithmus*	*Aufruf: CALL 768 (X)*
2.	*X-te Wurzel von Y*	*CALL 841 (X,Y)*
3.	*Umrechnung Altgrad in Bogenmaß*	*CALL 885 (X)*
4.	*Umrechnung Bogenmaß in Altgrad*	*CALL 901 (X)*

X oder Y können beliebige Ausdrücke oder Variablen sein. Die Syntax wurde so gewählt, daß die Argumente in Klammern stehen müssen. Werden die Funktionen wie im Beispiel aufgerufen, dann wird das Ergebnis direkt ausgedruckt. Wird allerdings dem Argument eine durch ein Komma getrennte Variable nachgestellt, dann wird das Ergebnis nicht gedruckt, sondern wird an die Variable übergeben. Z.B. CALL 768 (1000,X).

Das Ergebnis befindet sich jetzt in der Variablen X. Alle vier Routinen befinden sich im Bereich ab $300. Eine Beschreibung ist wegen des ausführlichen Assemblerlistings hoffentlich nicht nötig. Eine Besonderheit bilden die Routinen DEG und RAD: Um die Geschwindigkeit zu erhöhen, wurde die Konstante PI/180 als 5-Byte-Zahl im Speicherformat ab hex 03E0 abgelegt. Sie wird bei Gebrauch durch entsprechende Applesoft-Routinen in die Floating Point Akkus geladen und von dort verarbeitet. - Ein Hinweis für diejenigen, die sich über das Assembler-Listing wundern: Wegen des Fehlens eines APPLE-Assemblers wurde das Programm auf einem CBM 3000 mittels MAE assembliert.

```
                        0010 ; APPLE-MATH
                        0020 ; DEZ-LOG, X-TE WURZEL,
                        0030 ; DEG, RAD
                        0040 ;
                        0050 ; BY P. ENGELS
                        0060 ;
                        0070 KLAUF      .DE $DEBB
                        0080 KLZU       .DE $DEB8
                        0090 KOMMA      .DE $DEBE
                        0100 CGET       .DE $00B7
                        0110 SYNTAX     .DE $DEC9
                        0120 ARGIN      .DE $DD67
                        0130 STOMFP     .DE $EB2B
                        0140 LODMFP     .DE $EAF9
                        0150 OUTMFP     .DE $ED2E
                        0160 MEMDIV     .DE $EA66
                        0170 MEMMUL     .DE $E97F
                        0180 POWER      .DE $EE97
                        0190 MFP>SFP    .DE $EB66
                        0200 SFP/MFP    .DE $EA69
                        0210 READVAR    .DE $DFE3
                        0220 PUTVAR     .DE $EB2B
                        0230 MISMATCH   .DE $DD76
                        0240 LOG        .DE $E941
                        0250 STRINGFLAG .DE $11
                        0260 REAL10     .DE $EA50
                        0270 REAL1      .DE $F106
                        0280 ;
                        0290            .BA $0300
                        0300 ;
0300- 20 BB DE  0310 DLOG       JSR KLAUF
0303- 20 67 DD  0320            JSR ARGIN       ; ARGUMENT HOLEN
0306- 20 41 E9  0330            JSR LOG         ; LOG MFP
0309- A0 01     0340            LDY #$01
030B- A2 00     0350            LDX #$00        ; MFP AB $0100
030D- 20 2B EB  0360            JSR STOMFP      ; SPEICHERN
```

```
0310- A9 50      0370              LDA #L,REAL10          ; REALZAHL 10
0312- A0 EA      0380              LDY #H,REAL10          ; IN MFP
0314- 20 F9 EA   0390              JSR LODMFP
0317- 20 41 E9   0400              JSR LOG                ; LOG(10)
031A- A9 00      0410              LDA #$00
031C- A0 01      0420              LDY #$01
031E- 20 66 EA   0430              JSR MEMDIV             ; LOG(X)/LOG(10)
0321- 20 B7 00   0440 VAR          JSR CGET
0324- C9 29      0450              CMP #$29               ; WENN NICHT ')'
0326- D0 06      0460              BNE SEARCH             ; VAR. SUCHEN
0328- 20 BB DE   0470              JSR KLZU
032B- 4C 2E ED   0480              JMP OUTMFP             ; MFP DRUCKEN
032E- 20 BE DE   0490 SEARCH       JSR KOMMA
0331- 20 E3 DF   0500              JSR READVAR            ; VARIABLE SUCHEN
0334- 2C 11 00   0510              BIT STRINGFLAG         ; TEST OB
0337- 10 03      0520              BPL GOON               ; NUM-VARIABLE
0339- 4C 76 DD   0530              JMP MISMATCH
033C- AA         0540 GOON         TAX                    ; MFP AN VARIABLE
033D- 20 2B EB   0550              JSR PUTVAR             ; ÜBERGEBEN
0340- 20 B7 00   0560              JSR CGET
0343- F0 03      0570              BEQ RETURN
0345- 4C C9 DE   0580              JMP SYNTAX
0348- 60         0590 RETURN       RTS
                 0600 ;
0349- 20 BB DE   0610 WURZEL       JSR KLAUF
034C- 20 67 DD   0620              JSR ARGIN              ; EXPONENT HOLEN
034F- A9 06      0630              LDA #L,REAL1
0351- A0 F1      0640              LDY #H,REAL1
0353- 20 66 EA   0650              JSR MEMDIV             ; 1/EXPONENT
0356- A2 00      0660              LDX #$00
0358- A0 01      0670              LDY #$01               ; AB $0100 SPEICHERN
035A- 20 2B EB   0680              JSR STOMFP
035D- 20 BE DE   0690              JSR KOMMA
0360- 20 67 DD   0700              JSR ARGIN              ; BASIS HOLEN
0363- 20 66 EB   0710              JSR MFP>SFP            ; UND IN SFP LEGEN
0366- A9 00      0720              LDA #$00
0368- A0 01      0730              LDY #$01               ; EXPONENT/1 ZURÜCK
036A- 20 F9 EA   0740              JSR LODMFP
036D- A5 9D      0750              LDA *$9D
036F- 20 97 EE   0760              JSR POWER              ; X^(1/Y)
0372- 4C 21 03   0770              JMP VAR                ; VARIABLE ODER DRUCK
                 0780 ;
0375- 20 BB DE   0790 DEG          JSR KLAUF
0378- 20 67 DD   0800              JSR ARGIN
037B- A0 03      0810              LDY #H,PI180           ; HI & LO BYTE
037D- A9 E0      0820              LDA #L,PI180           ; KONSTANTE
037F- 20 7F E9   0830              JSR MEMMUL             ; X*(PI/180)
0382- 4C 21 03   0840              JMP VAR                ; VARIABLE ODER DRUCK
                 0850 ;
0385- 20 BB DE   0860 RAD          JSR KLAUF
0388- 20 67 DD   0870              JSR ARGIN              ; WINKEL HOLEN
038B- 20 66 EB   0880              JSR MFP>SFP            ; IN SFP
038E- A0 03      0890              LDY #H,PI180
0390- A9 E0      0900              LDA #L,PI180
0392- 20 F9 EA   0910              JSR LODMFP
0395- 20 69 EA   0920              JSR SFP/MFP            ; X/(PI/180)
0398- 4C 21 03   0930              JMP VAR                ; VARIABLE ODER DRUCK
                 0940 ;
                 0950              .BA $03E0
                 0960 ; KONSTANTE PI/180
                 0970 ; =0.174532925
                 0980 ; IN 5-BYTE SPEICHERFORMAT
                 0990 ;
03E0- 7B 0E FA   1000 PI180        .BY $7B $0E $FA $35 $11
03E3- 35 11
                 1010 ;
                 1020              .EN
ENDE VON MAE PASS !
```

Ing. grad. Peter Zechner, 8000 München 70

FORTH Disassembler (2)

Das in Heft 31 auf Seite 35 vorgestellte Programm 'FORTH-Disassembler' läuft nicht mit dem Rockwell-FORTH. Es sind möglicherweise auch einige Fehler enthalten. Die Idee ist gleichwohl so gut, daß ich mich daran gemacht habe, ein funktionierendes 'UN' zusammenzustellen. Nachfolgend das überarbeitete Quellprogramm:

```
: CASE ?COMP CSP @ !CSP 6 ; IMMEDIATE
 : OF 6 ?PAIRS COMPILE OVER COMPILE = COMPILE 0BRANCH
   HERE 0 ,  COMPILE DROP 7 ; IMMEDIATE
 : ENDOF 7 ?PAIRS COMPILE BRANCH HERE 0 ,
 SWAP 2 [COMPILE] ENDIF 6 ;  IMMEDIATE
: ENDCASE 6 ?PAIRS COMPILE DROP BEGIN SP@ CSP @ = 0=
  WHILE 2  [COMPILE] ENDIF REPEAT CSP ! ; IMMEDIATE
 : U< OVER OVER 0< SWAP 0< XOR
 IF SP@ 3 + C@ SP@ 3 + C@ < ROT ROT 2DROP
 ELSE - 0<
 THEN ;

 HEX
  : CFA-C <BUILDS 2- , DOES> @ ;
 ' LIT NFA CONSTANT LIT.NFA
 ' CLIT CFA-C CLIT.CFA
 ' LIT CFA-C LIT.CFA ' COMPILE CFA-C COMPILE.CFA
 ' ;S CFA-C ;S.CFA ' (;CODE) CFA-C (;CODE).CFA
 ' (LOOP) CFA-C (LOOP).CFA ' (+LOOP) CFA-C (+LOOP).CFA
 ' 0BRANCH CFA-C 0BRANCH.CFA ' BRANCH CFA-C BRANCH.CFA
 ' (.") CFA-C (.").CFA ' : 12 + CONSTANT :.CFA
' USER 4 + CONSTANT USER.CFA
 ' CONSTANT 8 + CONSTANT CST.CFA
 ' DOES> CONSTANT (DOES>).CFA
 ' VARIABLE 4 + CONSTANT VAR.CFA
A419 CONSTANT COUNT.M A425 CONSTANT SAVPC.M
CODE DISAS-JSR A421 STA, A422 STX, E72B JSR, A421 LDA,
A422 LDX, NEXT JMP, END-CODE
: DISASS SWAP SAVPC.M ! BEGIN 1 COUNT.M !
DISAS-JSR DUP SAVPC.M @ OVER OVER = ROT ROT U< OR
  ?TERMINAL OR  UNTIL ;
: T1 BL WORD HERE CONTEXT @ @ (FIND) DROP DROP ;
: ADR. CR DUP DUP 0 4 D.R @ DUP 0 5 D.R SPACE ;
: ?LIT DUP @ LIT.CFA = ;
: ?CPLE DUP @ COMPILE.CFA = ;
: WRITELN ADR. DUP CLIT.CFA =
IF ." CLIT " DROP 2+ DUP C@ 0 D. 1-
ELSE 2+ NFA ID.
THEN ;
: ?STOP DUP @ DUP ;S.CFA = OVER (;CODE).CFA = OR SWAP
DROP ;
: ?BRA DUP @ DUP
(LOOP).CFA = OVER (+LOOP).CFA = OR OVER
BRANCH.CFA = OR OVER 0BRANCH.CFA = OR SWAP DROP ;
: ?STRING DUP @ (.").CFA = ;
: STRNG 2+ DUP COUNT DUP >R TYPE R> 1- + ;
: CODE.C OVER ;
```

```
: SUCH-NFA LATEST SWAP OVER
           BEGIN ROT DROP DUP ROT ROT
           PFA LFA @
           DUP 3 PICK OVER OVER 1+ U< ROT ROT
          SWAP 1000 + U<
           AND ?TERMINAL OR
        UNTIL SWAP DROP ;
: SUCH DUP CFFF U< OVER
       LIT.NFA SWAP U< AND
IF SUCH-NFA ."    " ID. CR DROP
 ELSE ." AUSSER DIR-BEREICH" CR DROP
 THEN ;
: DECOD BEGIN
         WRITELN ?STOP 0= ?TERMINAL 0= AND
        WHILE
        ?LIT IF 2+ ADR. DROP ELSE
        ?BRA IF 2+ DUP @ OVER + 0 D. ELSE
        ?CPLE IF 2+ DUP @ 2+ NFA ID. ELSE
        ?STRING IF STRNG
        THEN THEN THEN THEN
 2+
 REPEAT
 DUP @ (;CODE).CFA =
 IF  2+ DUP SUCH-NFA DROP CR  DISASS
 ELSE DROP
 THEN ;

: UN: T1 DUP NFA DUP CR ID.
     C@ 40 AND IF ." IMMEDIATE" THEN CR
     BASE @ SWAP HEX DUP CFA @
     CASE VAR.CFA  OF CR ." VARIABLE  " @ 0 D. ENDOF
 CST.CFA OF CR ." CONSTANT " @ 0 D. ENDOF
 USER.CFA OF CR ." USER    " C@ 200 + @ 0 D. ENDOF
 (DOES>).CFA OF CR ." DEFINING WORD" DROP ENDOF
 CODE.C OF CR ." CODE DEFINITION "  CR DUP
 SUCH-NFA DROP DISASS ENDOF
 :.CFA OF DECOD ENDOF
 CR ." WORD DEFINED BY : " SUCH
 ENDCASE BASE ! ;
 DECIMAL ;S
 FINIS
```

AIM Spezial (16)

Herr Peter Zechner, München, weist auf mögliche Abnutzungserscheinungen an der Tastatur hin: Nach langjährigem Gebrauch arbeiten häufig benutzte Tasten nicht mehr zuverlässig. Mögliche Ursache ist, daß sich rückseitig die gelöteten Kontakte aus der Platine lösen. Abhilfe: Nachlöten!

Wie Grund und Boden, so sind auch freie Adressen in der Zeropage nicht beliebig vermehrbar. Beim AIM 65 werden 20 Zellen nur einmal vom Editor benutzt und zwar bei der Funktion FIND. Es sind dies die Zellen hex EB bis FF. Mit zwei kleinen Änderungen, die man sich in das F-ROM schießt, werden sie frei. Dazu verlegt man den Buffer für den Suchstring (da ja nur temporär benutzt wird) an eine andere passende Adresse, z.B. in die Stackseite ab hex 115. Hierzu die folgenden Programm-Änderungen, die platzmäßig ohne 'Schieben' aufgehen:

```
0000                0001 ;----- STRING =$EB - $FF von Zeropage verbannen
0000                0002 STRING =$0115          ;Neue Lage von STRING (z.B.)
0000                0003 CURPO2 =$A415
0000                0004 WIDTH  =30             ;neue Zeichenzahl moeglich
0000                0005
0000                0006        *=$F838         ;1.Aenderung
F838  99 15 01      0007        STA STRING,Y    ;----------
F83B                0008
F83B  C8            0009        INY             ;nur wenn Zeichenzahl zu aendern
F83C  C0 1E         0010        CPY #WIDTH
F83E                0011
F83E                0012
F83E                0013
F83E                0014
F83E                0015        *=$F85E         ;2.Aenderung
F85E  DD 15 01      0016        CMP STRING,X    ;----------
F861  F0 05         0017        BEQ $F868       ;FC9
F863  EE 15 A4      0018        INC CURPO2
F866  F0 E6         0019        BEQ $F84E       ;FC6
F868                0020
F868                0021        .END
 ERRORS=0000
```

□○□

Aus der Branche

OPEN ACCESS für den IBM-PC und kompatible, DEC Rainbow, TI, NCR und Wang. Eine beachtliche Präsentation ihres anwenderfreundlichen Software-Interfaces veranstaltete die SPI, Software Products International (Deutschland) GmbH in ihrem Etablissement in Hamburg 1, Amsinckstr. 44 (T.: 040-23 600 60). Hamburg ist mit derzeit 20 Mitarbeitern die Europazentrale. Weitere Niederlassungen in europäischen Großstädten sind in Vorbereitung. SPI ist die Tochter des von Dr. Peter Eichhorst gegründeten Softwarehauses SPI in San Diego, CA mit 120 Mitarbeitern.

OPEN ACCESS ist im Kern ein relationales Datenbanksystem mit 5 weiteren Facetten: Kalkulation (Spreadsheet), Wordprocessor, 3D-Grafik, Terminkalender und Datenfernübertragung zum Einschleifen in Netzwerke. Es sind also sechs professionelle Programme in einem Paket, das ca. DM 2.500 kostet + DM 500 für Installation und Wartung. Alle Programme können die Daten kommunizieren und miteinander verknüpfen. Auf den einzelnen Befehlsebenen sind die Funktionstasten gleichmäßig belegt. Es können auf dem Bildschirm mehrere Fenster monochrom und auch farbig dargestellt werden für Informationen, die im Zusammenhang stehen oder voneinander unabhängig sind. Typisch ist das Hauptfenster mit der Information, rechts daneben das Fenster mit den Begriffen zur Information (z.B. Name, Ort) und darunter dann breitgestreckt das Menue, was man in dieser Befehlsebene mit der Information tun kann (Drucken, Einsetzen, Entnehmen usw.). - Der Benutzer ist damit in der Lage, die umfangreichen Dienstleistungen des OPEN ACCESS nach kurzer Übung ohne Handbuch zu benutzen. Die Benutzerführung erfolgt in deutscher Sprache, auch die etwa 10 cm dicke Dokumentationskassette ist in Deutsch. - Der Benutzer kann sich bequem Ein- und Ausgabemasken schaffen und Kommandofiles anlegen, die ihn schnell auf den beabsichtigten Punkt der Bearbeitung führen.

Beachtenswert ist die 3D-Grafik (Text und Abbildung) mit der Möglichkeit des Drehens, der Schraffur, der Farbgabe usw.. Natürlich alles wie mit der Maus vom Menue gepickt. Grafiken können abgespeichert und zu einer 'Dia-Schau' abgerufen werden. Bei der Datenfernübertragung kann man ebenso vom Menue oder Kommandofile her die notwendigen Rapporte generieren. In der Demo schaute man in aller Kürze 'farbig' in den Briefkasten einer kalifornischen Datenbank.

Der Herausgeber glaubt, daß Personal Computer für den von den Herstellern beabsichtigten Vormarsch solche leistungsfähigen Softwareinterfaces benötigen und mit ihnen dann tatsächlich die weite Verbreitung finden. Man kann eben von den künftigen Anwendern nicht verlangen, daß sie selber programmieren. - Ähnliche freundliche Interfaces wurden in Heft 33 (LISA) und 34 (Editorial zu Digital Research) erwähnt. - Wenn man diese Bedienerführung sieht, dann überlegt man, ob man noch den zeitgemäßen Computer hat: Commodore, quo vadis?

40/80 Zeichenkarte und 64K RAM-Karte für Commodore VC-20 und C64. Beim Herausgeber konnten vorübergehend zwei Karten der Fa. Martin Roßmöller, (Finkenweg 1, 5309 Meckenheim, T. 02225-14488) getestet werden. Die dynamische RAM-Karte MR 64 ist für den VC-20 bestimmt. Ihre Montage geschieht in Minuten: CPU umstecken, Übergabestecker in den Sockel der CPU. Damit ist man bereits betriebsbereit. BASIC meldet sich jetzt mit 28159 Bytes Free. Durch ein Umschaltregister, das später auch weggeblendet werden kann, können nun weitere RAM-Bereiche den höheren Adreßblöcken zugeteilt werden, bis man schließlich eine reine RAM-Maschine hat, ähnlich dem C 64. Eine Übertragung des Betriebssystems in das RAM ist möglich. - Mit einer sauberen und ruhigen Wiedergabe auf einen Bildschirm-Monitor arbeitet die 40/80 Zeichenkarte, die in den Expansionsschacht gesteckt wird. Sie ermöglicht die Wiedergabe von 40 oder 80 Zeichen in einer Zeile und hat in ihrem Betriebssystem weitere Optionen für die Cursorsteuerung und die Editierung. Weitere Parameter der Darstellung können beim Power Up durch Drücken verschiedener Tasten beeinflußt werden. Diese für VC-20 und C 64 benutzbare Karte ergibt mit dem notwendigen hochauflösenden Monitor ein Bild, das dem auf dem 8032 vergleichbar ist, bei 40 Zeichen in der Zeile reichten die hiesigen Fernseher für ein befriedigendes Bild nicht aus. Die Karten machen in ihrer Verarbeitung einen guten Eindruck und sind ausreichend dokumentiert.

Rockwell International 1984 DATA BOOK der Semiconductor Products Division erschienen. Im Januar erhielt der Herausgeber das o.a. bezeichnete etwa 1000-seite (!) Handbuch (Bestell-Nr. 1, aufgedruckter Preis $5.00). Seine 11 Abschnitte umfassen: R68000 und Peripherie, 6500 Mikros und Peripherie, 6500 Microcomputer, Speicherprodukte, Intelligent Display Controllers, Entwicklungssysteme, AIM 65-Familie, AIM 65/40 Familie, RM 65 Module, Integral Modems und T-1 PCM Produkte. - Für sicher einige hundert Produkte findet man die Datenblätter oder die charakterisierenden Auszüge aus den Datenblättern. Insofern ist das Handbuch eine konzentrierte Informationsquelle für die Auswahl von Komponenten und für ihre Programmierung. Angesichts der Materialfülle können wir nicht auf Einzelheiten eingehen. Festzustellen ist jedoch, daß man von der Existenz vieler Komponenten nicht wußte, z.B. von den FORTH Based Microcomputers R65F11 und R65F12 mit 40 oder 64 (Quip) Pins, die einen erweiterten Befehlssatz haben, RAM und I/O sowie einen FORTH-Kernel. Sie sind damit hochsprachliche Einchipper, besonders auch für Steuerungszwecke geeignet.

Kurse zu Organisation, EDV, Neue Medien führt die IHK München und Oberbayern auch im ersten Halbjahr 1984 in Feldkirchen-Westerham durch. Info: IHK-Bildungszentrum Westerham, Von Adrian-Str. 5, 8152 Feldkirchen-Westerham, T.: 08063 -91-271 und -272.

Kleinanzeigen

Bufferplatine für 6502 & 65C02-Systeme (AIM 65, PC 100 u.ä) zum Aufstecken auf CPU-Fassung, 80x45mm, Adressen über 2x74LS244, Daten über 74LS245 gebuffert. Mit Schaltplan und Beschreibung DM 22,-- (+3,-- Ausland). Bernhard Kokula, Werderplatz 7, 6800 Mannheim 1 (neu!), Tel. 0621 - 41 19 86 (tags: 0621 - 381 - 88 82).

Geld sparen durch Selbstbau: Speichererweiterungen, Ramkarten, Epromkarten u. Programmiergeräte, Rombox etc.; z.B. 80 Zeichenkarte für VC 20, Leerplatine incl. Software, Bauanleitung und Schaltplan 99 DM; 64 kBytes dyn. Ramkarte 99 DM.**Katalog für CBM, VC20 und C64** gegen 2 DM. Roßmöller Datentechnik, Finkenweg 1, 5309 Meckenheim.

Assembler für Commodore 8032/4032 und 65C02 CPU von Rockwell, Editor mit Hilfsfunktionen, Assemblieren mit Offset und Crossreferenz, Disassembler. Alle Assembler laufen mit herkömmlicher CPU. Auf 4040-Diskette, ggfs. 8050/8250 DM 190,--. Peter Probadnigk, Finkenweg 23, 2070 Ahrensburg, T.: 04 102 - 57 051.

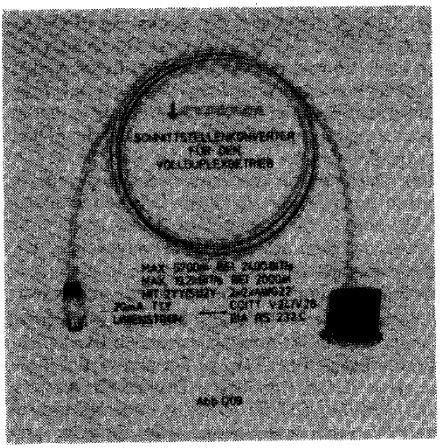
Textverarbeitung mit AIM 65: Groß- und Kleinschreibung, deutsche Tastaturbelegung, Tastenrepeat, schnelles REPLAC, Druckeranschluß. Monitor- und Editorvektoren u.v.a.m.. Zwei EPROMs **REMON 3.5** für 98 DM.

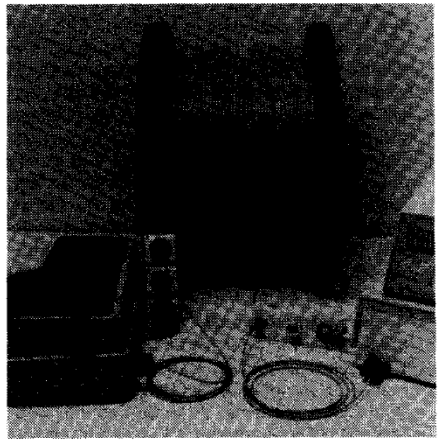
Editorerweiterung und Textformatierung für AIM 65: 9 neue Editorkommandos, z.B. Blockmove, -copy und -erase, 19 Formatierkommandos, z.B. Blocksatz, Seitenumbruch, Seitennumerierung, Zentrierung. **EXEDIT 5.3 und ROFF 3.2-** Software im 2532-EPROM, Sockel C, für 98 DM.

DISKMON 1.0: Commodore Floppystation VC-1541 am User-VIA des AIM 65 anschließbar. Software wie REMON 3.5, jedoch Diskettentreiber im E,F-Bereich integriert. Kein KIM-Tape, TTY, Thermoprinter, dafür ein weiterer Ausgabevektor. Zwei EPROMs für 128,- DM.

ASSMOS 1.0: Geänderter 4KByte-Assembler für AIM 65, Sockel D. Übersetzt alle neuen Maschinenbefehle der 6502-CMOS-CPU, bis auf BBR, BBS, RMB und SMB. Außerdem volle Unterstützung der Kleinschreibung, Anbindung von VC-1541 und 80 Zeichen Ausgabebreite. Software im 2532-EPROM für 58 DM. Paketangebote zu ermäßigten Preisen.

Bestellungen bei Dipl.-Ing. Rüdiger Wollenberg, Stockumer Str. 234, 4000 Dortmund 50, Tel.: 0231 - 75 34 77.





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

Haben Sie weit auseinanderliegende Datengeräte mit Standard V.24/V.28 oder RS 232 C Schnittstelle? ... dann ist 20 mA Current Loop die Lösung für Ihr lokales Netzwerk.

Den Konverter liefern wir:

Technische Daten:

— V.24/V.28 oder RS 232 C — 20 mA Current Loop.

— Galvanische Trennung

— Umschaltbar aktiv/passiv

— Betriebsanzeige: Senden, grüne LED
Empfangen, rote LED

— Maximale Leitungslänge 5700 Meter, Kabel 2YY (ST) 2Y2 x 2 x AWG22 (0,64); 58nF/km; 114 Ohm/km, 2,4 k Bit/sec sind dann noch möglich.

— Maximale Übertragungsrate bei gleichem Kabel: 19,2 k Bit/s bis 2000 m aktiv senden
und bis 290 m passiv senden.

— Einspeisung: +/— 12 V aus angeschlossenem Datengerät oder über Schalterdosennetzgerät SKNT 20, das zusätzlich in die Wandsteckdose der fest verlegten 20 mA Datenleitung eingebaut wird.

— Stromverbrauch: + 12 V ca. 60 mA max.
—12 V ca. 30 mA max.

— Abmessungen: 49 x 54 x 15 mm
AMP Subminiatur D Stecker 25polig, lieferbar mit Stiften oder Buchsen.

— Auf Wunsch mit flexibler Leitung 7 x 0,14 und Rundstecker Amphenol Tuchel 7polig.
Vier Adern Daten, drei für externe Stromversorgung.

Preise incl. MWSt

Konverter ohne Kabel ..... DM 342,00
Konverter mit 2,5 m
7 x 0,14 wie Abb. DO9: ..... DM 438,90
Netzgerät SKNT 20 für
Einbau in Installationsdose: ..... DM 171,00

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

STECKER
INGENIEURBÜRO

5000 Köln 60 (Niehl)
Postfach 60 07 66
Delmenhorster Straße 20
Telefon (02 21) 7 12 40 18
Neue Telefon-Nummer

Assembler

R65C02-Assembler

für AIM 65 und kompatible Systeme. 2-Pass-Assembler für den kompletten (!) Befehlssatz mit zusätzlichem Komfort. Assemblerliste formatiert, Assemblierung mit Offset für virtuelle Speicherbereiche. Der Assembler läuft auf der herkömmlichen 6502. - 2 EPROMs für Speicherbereiche Ihrer Wahl DM 195,--

6805/68705 Cross-Assembler

für AIM 65 und kompatible Systeme. 2-Pass-Assembler mit allem gewohnten Komfort und 6502-Syntax. Erzeugt aus den Mnemonics von Motorola 6805-Code. 2 EPROMs wie vor DM 280,--

6805/68705-Assembler unter FORTH

wie in Heft 35 beschrieben: 1-Pass-Assembler mit Verarbeitung von Symbolen und Labeln. Codeablage für virtuelle Speicherräume. 2 EPROMs für AIM 65 DM 100,--

Mathe-ROM für 6502

Implementierung nach Peter Rix (s. Hefte 28/29). Fließkommaarithmetik und höhere mathematische Funktionen wie in Microsoft-BASIC für AIM-65-FORTH und für jedes 6502-Assemblerprogramm (20 S. Dokumentation mit Einsprungspunkten und Argumenten), für Sockel $D000 DM 124,30

Floppy-Schnittstelle w. Beschaffungsschwierigkeiten nicht lieferbar.

Commodore Serie 700 und 600 ROM-Listing, ca. 300 S., geb.

Ausführliche Dokumentation mit original CBM-Labeln, Belegung des RAM und des I/O, Cross-Referenzliste. Assemblierfähiger Quelltext. Erklärung, wie die Banks benutzt werden. DM 74,--

LASER 110 Heimcomputer mit Microsoft-BASIC

CPU Z80A, 4 KB RAM. S. Besprechung in Heft 32 DM 198,--

Vorstehende Preis inkl. MWSt, zuzügl. DM 2,50 je Sendung im Inland

Kurse:

Assembler-Programmierung 6502/65C02 nebst Motorola 6805 und Interfacing in Ahrensburg. 16./17.März 1964

Einführung in FORTH, Ahrensburg 23./24. März 1984

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

DUO Plott Interface

für MX80 F/T und
ITOH 8510
und COMMODORE-Rechner 30/40/80

Paralleles IEEE 488 - Interface, genormter IEEE-Stecker und Kabel
kompletter Zeichensatz des CBM-Computers
zwei Geräteadressen für Groß-/Grafikmodus und Textmodus
alle Funktionen der Drucker bleiben erhalten, Floppy-Kompatibilität
Deutsche Umlaute, ß und Paragraph
Problemloser Einbau, inklusive deutsches EPSON-Handbuch
Komplettpreis einschl. Kabel, CBM-Grafiksatz und Handbuch incl. MWSt
für EPSON DM 298,--
für ITOH DM 298,--

Umrüstsatz MX80 F/T auf MX80 Typ 3

Aus Ihrem EPSON MX-80 F/T wird der MX-80 Typ 3
Einzelnadelansteuerung, erweiterter Befehlssatz, Elite-Schrift
Preis inkl. MWSt DM 98,--
Komplettpreis Interface, Kabel, Handbuch und Umrüstsatz inkl. MWSt DM 450,--

Für COMMODORE

Deutsche Tastatur mit Original-Tastensätzen (keine Aufkleber)
inkl. deutschem Zeichengenerator
für CBM 8032 DM 98,--
für CBM 8032 und 8096 DM 98,--
nur 8096, ohne Tasten DM 98,--

Speichererweiterung auf 96 K

LOS-kompatibel, inkl. MWSt DM 798,--

ISAM-Routinen
Datenhaltungsprogramm, Indexed Sequential Access Method
siehe Besprechung in Heft 23 des 65xx MICRO MAG, **DM 298,--**

Drucker:

RX 80 DM 998,-- + MWSt

ITOH 8510 DM 1495,-- + MWSt

EPSON-Druck-Computer FX 80 DM 1295,-- + MWSt

Stellberg Computer-Systeme

COMMODORE EPSON C. ITOH SOFTWARE INTERFACE

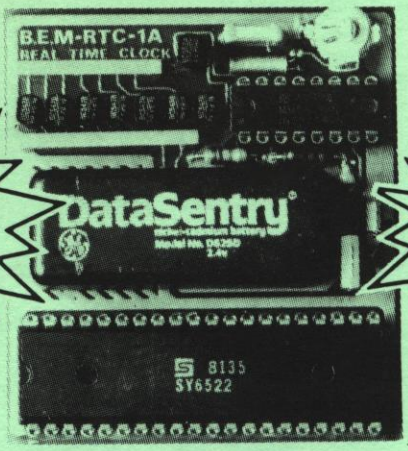
Blindenaaf 36 5063 Overath Tel.: 022 06 - 66 44

B.E.M-RTC-1A

Echtzeituhr/Kalender Module

Geeignet für alle 6502 und 6809

Computersysteme mit einer freiverfügbaren PIA (6520) oder VIA (6522)



NEU!

NEU!

PREIS: DM 165,– ohne MwSt.

Inkl. Batteriepufferung
Inkl. Software-Beispielen

Bezahlg.: Euroscheck in DM. Keine Portokosten.

BRUTECH ELECTRONICS

Postbus 58, 3645 ZK. Vinkeveen-Holland
Tel. 0 29 72/39 65, Telex 1 8 576 BEMIN-NL

65xx MICRO MAG

COMPUTING · SOFTWARE · HOBBY

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

65xx MICRO MAG erscheint zweimonatlich, jeweils Mitte Februar, April usw..- Offsetdruck: Druckartist Gerhard M. Meier, Hamburg 70.

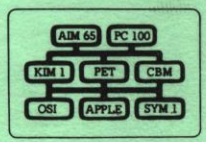
Bezugsbedingungen: Abonnement ab laufender Ausgabe für 6 Hefte DM 54,-- (Inlandsendpreis). Ausland/foreign via surface mail DM 59,--, USA air 26 Dollar. Abonnements laufen bis auf Widerruf mit Kündigungsmöglichkeit bis zu zwei Wochen vor deren Ablauf.

Nachliefermöglichkeiten: Hefte 1-6 und 7-13 sind als Buch nachlieferbar (s. Anzeige unten). Hefte 14-34 können alle als Einzelhefte zu DM 7,80/St. + DM 2,50 je Sendung geliefert werden. Einzelpreis bei 10 und mehr Heften je Sendung DM 6,--.

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

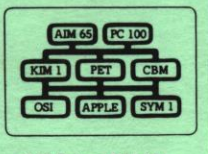
Leser-Service des Herausgebers

Das Buch 1-6 des 65.. MICRO MAG



230 Seiten, DM 26,--

Das Buch 7-13 des 65.. MICRO MAG



340 Seiten, DM 42,--

FORTH User's Manual
Rockwell-Handbuch für das FIG-FORTH des AIM 65. Mit der Erläuterung des Befehlssatzes auch für andere FORTH-Betreiber geeignet. Ca. 300 S., engl. DM 30,--

Mathe-ROM nach P. Rix für FORTH des AIM + Assemblerprog. m. Doku DM 124,30

Vorstehende Preise inkl. MWSt, zuzüglich DM 2,50/Sendung + ggfs. NN + DM 2,--

Assembler für R65C02, MC6805 und 6809: Siehe im Heftinneren.