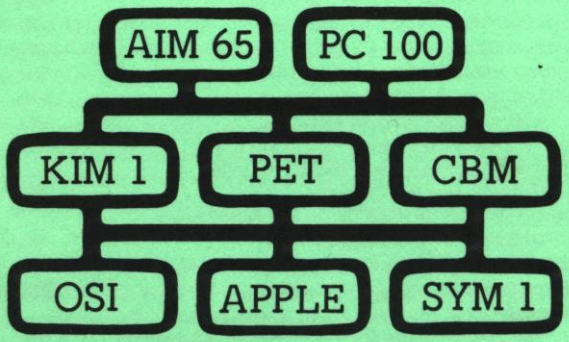
65xx MICRO MAG

COMPUTING · SOFTWARE · HOBBY

DM 8,50 Nr. 18 April 1981



Inhaltsverzeichnis

Einplatinencomputer NICO 69

CPU 6809

On board: 40 Pin Input/Output, 2 k RAM, 8 k EPROM-Fassungen vollständige Buspufferung, dadurch ausbaubar zum System: max. 50 k RAM durch Combo- bzw. RAM-Karten, Diskoperatingsystem bzw. Cassettenoperatingsystem mit Digitalcassettenrecordern, Editor, Assembler, LISP, PASCAL.

CPU-Karte NICO 65 (CPU 6502)	Grundausstattung	390,-- DM
CPU-Karte NICO 65	komplett	450,-- DM
CPU-Karte NICO 69 (CPU 6809)	Grundausstattung	450,-- DM
CPU-Karte NICO 69	komplett	510,-- DM
Videokarte		660,-- DM
Combo-Karte voll bestückt		560,-- DM
Combo-Karte, nur mit RAMs bestückt		470,-- DM
Combo-Karte nur mit VIAs bestückt		140,-- DM
Buskarte mit 14 Steckplätzen		280,-- DM
EPRO-Programmiereinheit für 2758, 2716, 2732 incl. Betriebsprogramm auf Cassette		285,-- DM
Monitorprogramm im EPROM für serielle Schnittstelle mit umschaltbarer Ein- und Ausgabe		90,-- DM
Experimentierkarte		110,-- DM
Analog/Digitalwandler		324,-- DM
Monitorprogramm beinhaltet die Steuerung der Grundfunktionen des Systems		59,-- DM
CRTKEY beinhaltet die Steuerung der Videokarte, der Tastatur und den ladbaren Grundzeichensatz		49,-- DM
SICOS Das Cassettenoperatingsystem ist ein Betriebssystem, welches außer sämtlichen Cassettenfunktionen auch Systemkommandos beinhaltet		290,-- DM
EDASM Dieser Editor ist vor allen Dingen in der Zeileneditierung besonders komfortabel. Der angeschlossene Assembler beinhaltet auch höhere Strukturen wie IF THEN ELSE		480,-- DM
SICAL Diese Rechenroutinen rechnen dezimal mit 14-stelliger Genauigkeit, so daß sie für kaufmännische Zwecke bedenkenlos einsetzbar sind		98,--DM
SICOS Das Diskoperatingsystem gleicht in den Kommandos dem Cassettenoperatingsystem		560,-- DM
AIMCOS gestattet den Anschluß der schnellen Mini-Digitalcassettenrecorder an den AIM 65/PC 100		290,-- DM

Alle Preise zzgl. 13% MWSt.

ELEKTRONISCHE BAUGRUPPEN
MICROPROZESSOR-TECHNIK
Dohmann

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

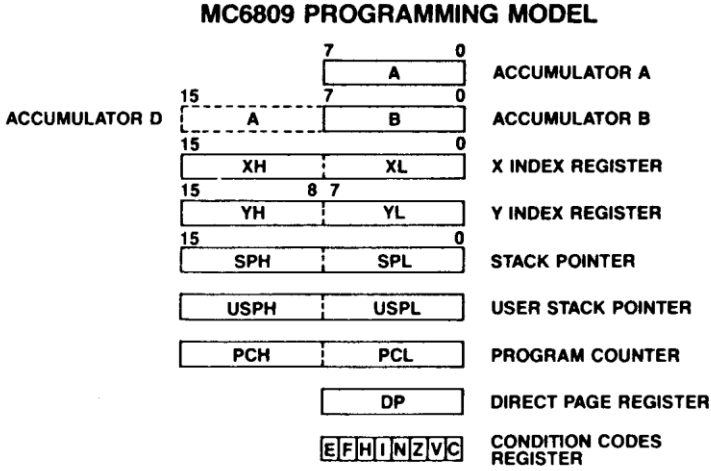
MC 6809: Register, Signale, Befehle

In Heft 15 dieser Zeitschrift wurde der 6809 von Motorola bereits als 'Ein fortschrittlicher Verwandter' des 6502 im Überblick dargestellt. Dieser Aufsatz bemüht sich nun um eine gründlichere und systematische Darstellung, besonders hinsichtlich des Befehlssatzes und der Adressierung. Zum Verständnis eines Prozessors und zur Bewertung seiner Möglichkeiten gehört gleichwohl auch die Kenntnis seiner Register und seiner Ein- und Ausgangssignale. Wir beginnen daher mit den beiden letztgenannten Abschnitten.

Register zur Programmierung des 6809.

Eine MPU, eine Microprocessing Unit, bildet ein hochkomplexes digitales Schaltwerk mit tausenden von Transistorfunktionen. Der größte Teil der die Befehlsausführung steuernden elektronischen Register ist für den Programmierer nicht erreichbar, sie leisten ihren Dienst im Verborgenen. Das gilt insbesondere für das Instruktionsregister (Steuerwerk), das die hereinkommenden Befehle interpretiert und die Aktionen innerhalb der MPU durchschaltet, es gilt gleichermaßen auch für die ALU (Arithmetical and Logical Unit), die die arithmetischen und logischen Verknüpfungen wahrnimmt und ihre Ergebnisse in den erreichbaren Registern ablegt. - Die mit Befehlen des Programmierers ansprechbaren Maschinenregister bilden also eine Untermenge aller MPU-Register.

Der 6809 ist ein 8-Bit Prozessor, gemessen an der Breite seines Datenbusses. Seine internen Register sind vorwiegend jedoch in 16 Bit Breite angelegt und es gibt viele Befehle, die in dieser Arbeitsbreite wirken. Man spricht daher auch von einem Pseudo-16-Bit-Prozessor. Die erreichbaren Register sind in nachfolgendem Schema aufgeführt:



Wir bemerken zwei jeweils 8 Bit breite Akkumulatoren, Akku A und Akku B, die mit einem fast identischen Befehlssatz ausgerüstet sind, der ihnen vor allen anderen Registern das höchste Maß an Rechenhaftigkeit und logischer Verknüpfungsmöglichkeit verleiht.

Es gibt Befehle, mit denen man diese beiden Akkus A und B zugleich als einen Akku D ansprechen kann. Sie sind dann zu einem (nur gedacht vorhandenen) Akku D zusammengekoppelt, wobei das höherwertige Byte in Akku A steht. - Der Befehlssatz für Akku D ist deutlich kleiner als für A oder B allein, er beschränkt sich auf Laden, Abspeichern, Addition, Subtraktion und Vergleich.

Die Indexregister X und Y sind in 16 Bit Breite angelegt. Sie werden in der indizierten Adressierung zur Bildung der effektiv auf den Adreßbus auszusendenden Adresse benutzt. Die in den Registern enthaltene Adresse mag dabei direkt auf die Daten zeigen oder noch abgewandelt werden, und zwar um einen konstanten Offset oder mit dem Inhalt eines Registers. Bei einigen indizierten Adressierungen, die der Bearbeitung von tabellarisch angeordneter Information dienen, wird das Indexregister automatisch nach der Instruktionsausführung erhöht oder erniedrigt, um auf das nächste Datenelement zu zeigen.

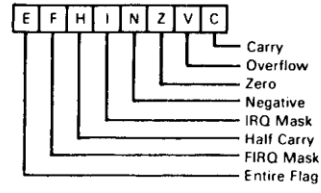
Auch die beiden Stackpointer U und S können wie X und Y als Indexregister benutzt werden. Der Hardware-Stackpointer S ist in die automatische Registerablage und -wiederherstellung während Unterprogrammaufrufen, Interrupts und Rückkehr von dort eingebunden. Der User-Stackpointer U steht dem Programmierer exklusiv zur Verfügung und eignet hervorragend zur Parameterübergabe an andere Programmteile, wobei die Parameter nicht selbst bewegt werden müssen, sondern nur der Zeiger auf die Parameter im Register U. - Beide Stackpointer müssen bei Betriebsbeginn per Programm initialisiert werden, sie können auf jede Adresse im 64 KB Adreßraum weisen. Im Gegensatz zum 6502 liegt der Stack also nicht hardwaremäßig in Speicherseite 1. Und noch ein Unterschied: Die Stackpointer zeigen nicht auf den nächstfreien Platz im Stack, sondern auf das zuletzt abgelegte Byte. - Für die Stacks S und U gibt es außer den indizierten Adressierungsmöglichkeiten PUSH- und PULLbefehle, mit denen stackorientierte höhere Programmiersprachen bequem implementiert werden können und die auch eine modulare Programmierung erlauben.

Der Programmzähler PC zeigt in 16 Bit Breite und auf bekannte Art auf den nächst auszuführenden Maschinenbefehl. In einigen Fällen ist eine Adressierung relativ zum Programmzähler möglich (adreßraumunabhängige Programmierung), wobei der PC wie ein Indexregister benutzt wird.

Der Inhalt des Direct Page Register DP wird bei allen Befehlen mit der verkürzten direkten und beschleunigt ausführenden Adressierungsart automatisch auf den hohen Adreßbus ausgesandt (A8 bis A15). Es erlaubt damit eine 'Zero Page-Adressierung' wie beim 6502, jedoch nach dem Willen des Programmierers in allen Speicherseiten. Durch einen RESET wird das DP auf 00 eingestellt. Im weiteren Verlauf kann es durch Register-Transferbefehl unter Kontrolle und Verantwortlichkeit des Programmierers auf die benötigte Speicherseite eingerichtet werden.

Der Prozessorstatus wird im CC, dem Condition Code Register geloggt. Seine 8 Bitpositionen haben (von rechts nach links) folgende Bedeutung:

CONDITION CODE REGISTER FORMAT



Das Carry-Flag wird in der Position 0 geführt. Beeinflußt wird es von den arithmetischen Operationen Addition, Subtraktion (um Überlauf oder Borgen anzuzeigen), von Komplementbildung und Vergleich, von der Multiplikation mit Ergebnis im Akku D vom Decimal Adjust auf BCD Darstellung im Akku A und von den Verschiebeoperationen in den Akkus oder im Me mory. Näheres bei den Befehlen.

Das V-Flag ist das bekannte Overflow-Flag, das bei arithmetischen Operationen für vorzeichenbehaftete Zahlen einen Wechsel des in Bit7 geführten Vorzeichens anzeigt.

Die Flags Z (Zero) und N (Negative) zeigen in bekannter Weise nach einer Operation an, ob sich im Register oder Memory das Ergebnis Null bzw. ein als negativ zu interpretierendes Ergenbnis (bit7=1) eingestellt hat. Es ist anmerkenswert, daß auch Store-Befehle diese Flags beeinflussen, während das beim 6502 nicht der Fall ist.

Die vorderen vier Statusbits hängen mit Interruptsteuerung und Interruptereignissen zusammen, auf deren Signale im nächsten Abschnitt näher eingegangen wird.

Ein gesetztes I-Flag (IRQ-Flag) bewirkt, daß über den IRQ-Pin hereinkommende Interruptanforderungen nicht akzeptiert werden (maskierter Interrupt). Das I-Flag wird durch folgende Interruptquellen zu 1 gesetzt: NMI, FIRQ, IRQ, RESET und SWI (Software-Interrupt).

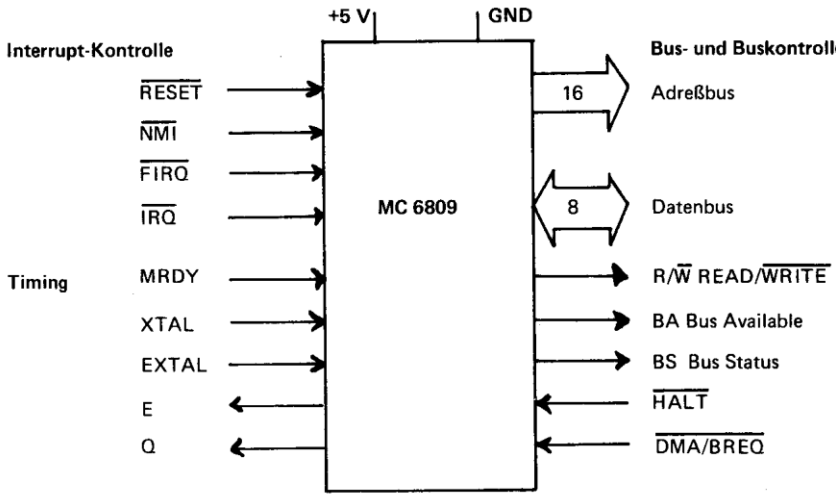
In Bit 5 wird das H-Flag geführt (Half Carry). Es ist nach einer Addition gesetzt, wenn sich aus dem niederen Halbbyte unter BCD-Interpretation ein Übertrag in das höhere Halbbyte ergibt, der beim Decimal Adjust zu berücksichtigen ist.

Bit 6 im Status (F) ist das Maskierungsbit für den mit relativer Priorität versehenen schnellen Interrupt FIRQ. Es wird durch folgende Interruptereignisse zu 1 gesetzt: NMI, FIRQ, SWI und RESET.

Das E-Flag (Entire-Flag) im Status gibt an, ob durch einen Interrupt alle Maschinenregister auf dem Hardwarestack gesichert wurden (E=1 nach IRQ, NMI, SWI, SWI2, SWI3) oder nur der Subset Rückkehradresse und Prozessorstatus (E=0 nach 'schnellem' Interrupt FIRQ).

Signale an der CPU

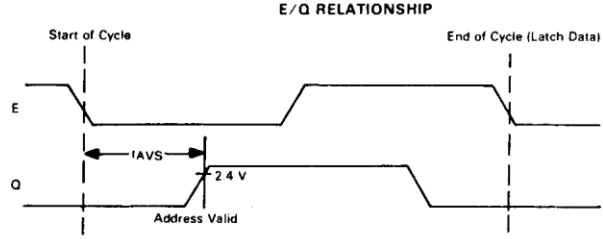
Für die Signale an der CPU ergibt sich folgendes Blockdiagramm:

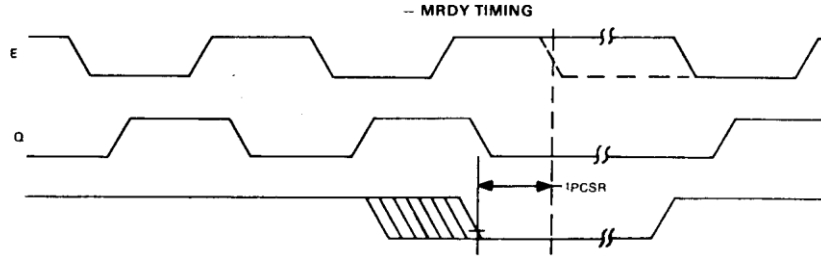


Timing-Signale

Als dynamische Maschine benötigt der Mikroprozessor ein externes Taktsignal, das ihm über die Pins XTAL und EXTAL zugeführt wird. Vom externen Oszialltor (Quarz) wird eine Frequenz verlangt, die das 4-fache der MPU-Nennfrequenz ist. Es gibt Versionen der 6809 mit Nennfrequenzen von 1.0, 1.5 und 2 MHz.

Aus dem angelegten takt erzeugt die MPU die **Ausgangstaktsignale Q und E für das System,** die in ihrer Überlappung vielleicht ungewohnt sind: Ein ausgesandte Adresse wird mit der steigenden Impulsflanke von Q gültig. Die Datenübernahme, lesend oder schreibend, erfolgt mit der abfallenden Impulsflanke des Signales E, das damit dem bekannten 02 entspricht.





Bus-Signale und Buskontrolle

Der Adreßbus hat 16 Pin und kann damit 64 K Speicher adressieren. Dieser Bus kann in Abhängigkeit von anderen Eingangssignalen in den hochohmigen Zustand (Tri-State) gebracht werden. Das Ausgangssignal BA (Bus Available) ist dann =1. Dadurch ist DMA-Fähigkeit gegeben (Direct Memory Access): Eine andere Adreßquelle kann den Bus regieren.

Der Datenbus mit seinen 8 Pin ist wie üblich bidirektional. Das dazugehörige **READ/WRITE-Signal** zeigt mit =1 einen Lesevorgang der MPU und mit =0 einen Schreibvorgang an.

Das Eingangssignal MRDY (Memory Ready) ist für das Zusammenspiel mit langsameren Speichern gedacht, es zögert die abfallende datenübernehmende Impulsflanke des Signales E um bis zu 10 Mikrosekunden hinaus, und zwar in einer 'Feinkörnung' (granularity) von 1/4 Buszyklen.

Die Ausgangssignale BA (Bus Available) und BS (Bus Status) sind im Zusammenhang zu betrachten. Ihre Konbinationen haben folgende Bedeutung:

MPU State		
BA	BS	
0	0	Normal (Running)
0	1	Interrupt Acknowledge
1	0	SYNC Acknowledge
1	1	HALT or Bus Grant

Insbesondere zeigt BA=1 den Tri-State-Zustand des Adreßbusses an. Bei der Verwertung dieses Signales sollte man tote Zyklen beachten.

Die Konbination SYNC Acknowledge in vorstehender Tabelle stellt einen Wartestand der MPU dar, bei der durch einen Befehl des Programmierers (SYNC-Befehl) die weitere Abarbeitung eingestellt wird, bis ein Interruptereignis von außen registriert wird (Synchronisierung). Im Anschluß an dieses Interruptereignis gibt es 4 Möglichkeiten: Zunächst wird der SYNC-Status aufgehoben. Wenn der Interrupt abmaskiert war oder wenn das Interruptsignal kürzer als 3 Maschinenzyklen ansteht, fährt das Programm mit der nächsten Instruktion im Vordergrundprogramm fort, ohne Register zu stacken und ohne in eine Interruptroutine einzutreten (reine Synchronisation). Wenn der Interrupt nicht maskiert war und wenn auch das Interruptsignal 3 Zyklen und länger ansteht, wird der Interrupt abgearbeitet (Register stacken, Interruptprogramm).

Interessant ist auch die Kombination BA=0 und BS=1, Interrupt Acknowledge. Sie tritt in beiden Lesezyklen auf, in denen der Prozessor seinen Programmzähler mit einem Interruptvektor aus dem Top of Memory laden will. In diesem Moment kann man mit weiterer Außenbeschaltung ein hardware vectoring für den Interrupt vollziehen. In Abhängigkeit vom Interruptereignis werden dem Prozessor auf dem Datenbus Vektoren suggeriert, die nicht dem normalen Inhalt der höchsten Speicheradressen entsprechen.

Hinsichtlich der Bussteuerung sind noch zwei Eingangssignale abzuhandeln: **Nach HALT=0 beendet der Prozessor die laufende Instruktion** und hält dann ohne Datenverlust beliebig lange an. Der Adreßbus wird hochohmig gesteuert, die Clocksignale Q und E verlassen weiterhin die MPU, IRQ- und FIRQ-Interrupts werden nicht mehr akzeptiert, NMI- und RESET-Signale werden nur zur späteren Reaktion gespeichert.

Das Eingangssignal DMA/BREQ führt ebenfalls zum Anhalten der MPU nach Ausführung der laufenden Instruktion, Q und E werden weiter erzeugt. Jedoch: Die MPU führt alle 16 Takte einen eigenen Refreshzyklus aus, so daß in dieser Zeit ein DMA-Zugriff unterbrochen werden müßte. Dieses Signal dient daher typisch dem Refresh dynamischer RAMs, wohl weniger einem DMA, das viele Byte erfassen soll.

Die Interrupt-Signale.

Der 6809 kennt drei Befehle zur Erzeugung eines Software-Interrupts, nämlich SWI, SWI2 und SWI3. Auf elektrischem Wege werden Interrupts durch die Signale RESET, NMI, FIRQ und IRQ ausgelöst (alle sind mit LOW wahr). Jeder dieser Interrupts führt nach seiner Annahme zum Laden eines spezifischen Interruptvektors aus den höchsten Speicherzellen gem. folgender Adressierung:

MEMORY MAP FOR INTERRUPT VECTORS

Memory Map For Vector Location		Interrupt Vector Description
MS	LS	
FFFE	FFFF	$\overline{\text{RESET}}$
FFFC	FFFD	$\overline{\text{NMI}}$
FFFA	FFFB	SWI
FFF8	FFF9	$\overline{\text{IRQ}}$
FFF6	FFF7	$\overline{\text{FIRQ}}$
FFF4	FFF5	SWI2
FFF2	FFF3	SWI3
FFF0	FFF1	Reserved

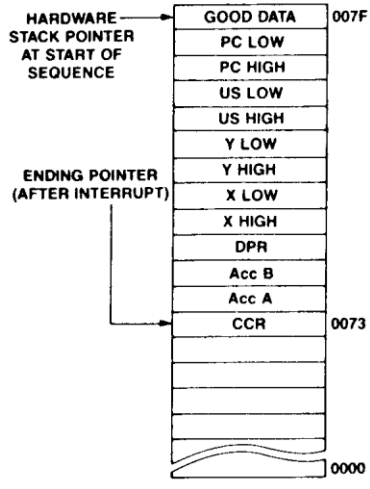
Das RESET-Signal ist wie üblich aktiv LOW. Ein Schmidt-Trigger mit einer erhöhten Schwellwertspannung stellt sicher, daß alle Standard-Peripheriebausteine den RESET-Zustand bereits vor der MPU verlassen.

Der nicht maskierbare Interrupt NMI wird durch die abfallende Impulsflanke am Pin ausgelöst, während der **IRQ** wie üblich durch LOW level ausgelöst wird, der solange anstehen muß, bis die MPU auf das Interruptereignis eingehen kann (also keine Speicherung des IRQ im Prozessor). NMI und IRQ sowie auch die drei Softwareinterrupts führen zum Stacken aller Maschinenregister auf dem Hardware-Stack S und zum Setzen des Entire-Flags E im Status. Mit dieser Prozedur ist einiger Zeitaufwand verbunden. Es wurde daher auch ein **'schneller Interrupt' FIRQ eingerichtet**, der nur Programmzähler und Status stackt. Der FIRQ hat Vorrang vor dem IRQ, er greift jedoch nicht durch, wenn ein NMI, SWI oder RESET zuvor schon automatisch das F-Flag gesetzt hatte.

Die Erweiterung der Interruptstruktur hat nicht nur im FIRQ und den 3 Softwareinterrupts ihren Niederschlag gefunden. Erwähnt wurde schon das Signal Interrupt Acknowledge an BA und BS. Hier ist weiterhin erwähnenswert, daß es eine **Instruktion CWAI** gibt (Wait for Interrupt), die die Menge aller Register bereits vor dem Interruptereignis auf den Hardwarestack rettet, um eine beschleunigte Reaktion zu erzielen. Sie setzt auch das E-Flag zu 1. Das zweite Byte im Befehl CWAI dient dem Maskieren des Statuswortes CC, wodurch das Interrupt-Enable für IRQ oder FIRQ beeinflußt werden mag. Tritt ein FIRQ ein, so werden auch bei seinem Abschluß mit RTI automatisch alle Register wiederhergestellt (normal nur PC und CC mit dem RTI des FIRQ).

Zur Interruptstruktur sei schließlich noch einmal der **SYNC-Befehl** (s.o.) erwähnt, der die Synchronisation der MPU auf ein Interruptereignis zuläßt.

RAM USED FOR STACK STORAGE
RAM ADDRESS $0000–$007F



Dem nebenstehenden Beispiel ist die Reihenfolge der Registerablage auf den Hardwarestack nach einem Interrupt zu entnehmen.

(Die Mehrzahl der Abbildungen wurde der Motorola-Dokumentation zum 6809 entnommen.)

Soviel zu den Registern und Signalen der MPU. Wir können uns damit dem Befehlssatz, den Adressierungsarten und später der Assembler-Syntax zuwenden.

R.L. #

StDir. Peter Rix, 2350 Neumünster

Laufzeitmessung für Programme

Auch für den versierten Programmierer ist das Beurteilen der Zeitbedingungen bei programmtechnischen Alternativen nicht immer einfach, und das Auszählen von Programmbefehlen und Zykluszahlen bei zeitkritischen Routinen ist eine mühsame und fehlerträchtige Arbeit. Aufwendige Entwicklungssysteme bieten deshalb Möglichkeiten, die Zahl der zur Programmabarbeitung erforderlichen Taktzyklen zu zählen. Genau diese Möglichkeit wird mit dem Programm LZEIT für den AIM 65/PC 100 geschaffen.

Programmlaufzeiten bis zu 12,7 Tagen werden auf den Maschinentaktzyklus genau ausgemessen, für die Taktfrequenz 1 MHz stimmen die ermittelten Zeiten damit auf die Mikrosekunde genau.

Das Meßprogramm benutzt zur Taktzählung den Timer 2 des VIA 6522, der nach Auslösung eines Interrupts durch Null dekrementiert, zur Auswertung der Bedienungsverzögerung durch die CPU aber in der Interrupt-Serviceroutine angehalten werden kann.

Das Meßprogramm ist im Speicher frei verschieblich, seine Variablen liegen in den Breakpointadressen des Stacks und im Displaybuffer, wo kaum Störungen mit Anwenderprogrammen zu erwarten sind. Außer der Anfangsadresse 'INIT', an der die Messung gestartet wird, müssen keine Programmadressen bekannt sein, denn das Programm setzt nach dem Start einer Messung seine Vektoren für die Interrupt-Routine, für das Ende des Meßvorgangs und für die Rückkehr in ein BASIC-Programm selber, die Lage im Adreßraum kann dabei beliebig sein, die Vektoren können vorher vom Anwenderprogramm anders belegt worden sein. - Die Einbindung des Meßprogrammes in Maschinen- oder BASIC-Programme des Anwenders ist denkbar einfach:

1. Meßprogramm in beliebigen freien Speicherbereich laden. Es hat sich als praktisch erwiesen, auf der Cassette ein Programm zum verschieblichen Laden vorzuschalten, das. z.B. im Stack residieren kann.

2. Ausmessen von Maschinenprogramm-Laufzeiten

```
JSR INIT                    startet die Messung
.....          auszumessender Maschinencode
JSR E89C                    stoppt die Messung
                            Zeitanzeige/Druck
.....          Fortsetzung, z.B. JMP Monitor.
```

   Falls der auszumessende Code allein nicht lauffähig ist und als Teil eines ablaufenden Programmes erfaßt werden soll, müssen die beiden JSR-Befehle in das Benutzerprogramm eingefügt werden. Der dadurch verdrängte Anwendercode kann, ohne in die Messung einzugehen, in die NOP-Adressen am Anfang und Ende des Meßprogrammes ausgelagert werden.

3. Ausmessen von BASIC-Programmlaufzeiten

```
NN POKE 4,INIT-LOW: POKE 5,INIT-HIGH: T=USR(T)
                            startet die Messung
.....          auszumessendes BASIC-Programm
MM T=USR(T)                 stoppt die Messung
                            Zeitanzeige/Druck
.....          Fortsetzung BASIC-Programm.
```

Das stoppende Statement in Zeile MM geht mit ca. 1,6 ms in die Messung ein (Messung in Leerprogramm). Alle BASIC-Zeilen zwischen NN und MM werden zeitlich erfaßt. Wird das BASIC-Programm bei ausgeschaltetem Printer fortgesetzt, sollte die Zeile MM durch :PRINT!"" vervollständigt werden, die Zeit wird dann ausgedruckt.

```
0000          ;--------------------
0000          ;ZEITMESSUNG
0000          ;
0000          ;--------------------
0000          ;DATEINAME: LZEIT
0000          ;
0000          ;--------------------
0000          ;P.RIX, 19.02.81
0000          ;
0000          ;--------------------
0000          ;DEFINITIONEN
0000          ;
0000 INIT=$EA0
0000          ;
0000          ;STARTADRESSE
0000          ;GGFS. ABAENDERN
0000 AUS=$E89C
0000          ;
0000          ;STOPPADRESSE, AENDERT SICH
0000          ;AUCH B. PROGR-VERSCHIEBUNG NICHT
0000 BLANK=$E83E
0000 CLR=$EB44
0000 EQUAL=$E7D8
0000 NOUT=$EA51
0000 NUMA=$EA46
0000 OUTALL=$E9BC
0000 RTS1=$F55A
0000          ;
0000 ZYKZ=$100
0000          ;
0000          ;ZYKLUSZAEHLER, HEX
0000          ;5 BYTE IM STACK
0000 UOUT=$10A
0000          ;
0000          ;STOPPVEKTOR
0000 VIA=$A000
0000          ;
0000          ;BENUTZER-VIA
0000 T2L=VIA+8
0000          ;
0000          ;TIMER 2
0000 T2H=VIA+9
0000 ACR=VIA+$B
0000 IER=VIA+$E
0000 IRQV4=$A400
0000          ;
0000          ;INTERRUPTVEKTOR NACH BREAK
0000 ZEIT=$A450
0000          ;
0000          ;ZEITSPEICHER, DEZIMAL
0000          ;7 BYTE IM DISPLAYBUFFER
0000          ;
0000 DIFF1=IRRUPT-PC-2
0000          ;
0000          ;ADRESSDIFFERENZEN
0000          ;FUER VEKTORINITIALISIERUNG
0000 DIFF2=AUSW-PC-2
0000          ;
```

```
0000            ;--------------------
0000            ;INITIALISIERUNG
0000            ;--------------------
0000
0000                    *=INIT
0000            ;STARTADRESSE FUER ZEITMESSUNG
0EA0 INIT       EA      NOP
0EA1            ;DURCH PROGRAMMAUFRUF JSR INIT
0EA1            ;UBERSCHRIEBENER MASCHINENCODE
0EA1            ;KANN HIER ZWISCHENGESPEICHERT WERDEN
0EA1            EA      NOP
0EA2            EA      NOP
0EA3            EA      NOP
0EA4            EA      NOP
0EA5            08      PHP
0EA6            78      SEI
0EA7            48      PHA
0EA8            8A      TXA
0EA9            48      PHA
0EAA            ;ERMITTELN DER ZUR VEKTORINITIALISIERUNG
0EAA            ;BENOETIGTEN PROGR-ADR. PC+2
0EAA PC         205AF5  JSR RTS1
0EAD            BA      TSX
0EAE            CA      DEX
0EAF            D8      CLD
0EB0            ;BERECHNEN & LADEN
0EB0            ;DES IRQV4-VEKTORS
0EB0            18      CLC
0EB1            BD0001  LDA $100,X
0EB4            695E    ADC #<DIFF1
0EB6            8D00A4  STA IRQV4
0EB9            BD0101  LDA $101,X
0EBC            6900    ADC #>DIFF1
0EBE            8D01A4  STA IRQV4+1
0EC1            ;BERECHNEN & LADEN
0EC1            ;DES UOUT-VEKTORS
0EC1            BD0001  LDA $100,X
0EC4            699A    ADC #<DIFF2
0EC6            8D0A01  STA $10A
0EC9            BD0101  LDA $101,X
0ECC            6900    ADC #>DIFF2
0ECE            8D0B01  STA $10B
0ED1            ;INITIALISIEREN DER STOPPADRESSE
0ED1            ;FUER DEN AUFRUF UBER USR-FUNKTION
0ED1            ;VON BASIC HER
0ED1            A99C    LDA #<AUS
0ED3            8504    STA 4
0ED5            A9E8    LDA #>AUS
0ED7            8505    STA 5
0ED9            ;INITIALISIEREN DES ZYKLUSZAEHLERS
0ED9            ;ANFANGSWERT BERUECKSICHTIGT
0ED9            ;BEREITS B. MESSENDE NOTWENDIGE
0ED9            ;KORREKTUR MIT T2-INHALT
0ED9            A9E7    LDA #$E7
0EDB            8D0401  STA ZYKZ+4
0EDE            A9FE    LDA #$FE
0EE0            8D0301  STA ZYKZ+3
0EE3            A900    LDA #0
0EE5            A202    LDX #2
```

```
ØEE7 RES     9DØØØ1 STA ZYKZ,X
ØEEA         CA     DEX
ØEEB         1ØFA   BPL RES
ØEED         ;T2 F. INTERRUPT IM INTERVALL-
ØEED         ;TIMER-BETRIEB VORBEREITEN
ØEED         8DØBAØ STA ACR
ØEFØ         A97F   LDA #$7F
ØEF2         8DØEAØ STA IER
ØEF5         A91C   LDA #28
ØEF7
ØEF7        ;2 STARTEN. MESSWIRKSAM WIRD
ØEF7         ;DER INHALT
ØEF7         ;STARTWERT -28 = $FFØØ
ØEF7         8DØ8AØ STA T2L
ØEFA         A9FF   LDA #$FF
ØEFC         8DØ9AØ STA T2H
ØEFF         A9AØ   LDA #$AØ
ØFØ1         8DØEAØ STA IER
ØFØ4         ;INTERRUPTFREIGABE T2
ØFØ4         ;RUECKSPEICHERN PROZESSORREGISTER
ØFØ4         68     PLA
ØFØ5         AA     TAX
ØFØ6         68     PLA
ØFØ7         28     PLP
ØFØ8         58     CLI
ØFØ9         6Ø     RTS
ØFØA         ;ZURUECK Z. CALLER
ØFØA
ØFØA         INTERRUPTROUTINE
ØFØA
ØFØA IRRUPT  48     PHA
ØFØB         ;RETTEN AKKU
ØFØB         A92Ø   LDA #$2Ø
ØFØD         8DØBAØ STA ACR
ØF1Ø         ;T2 ANHALTEN
ØF1Ø         D8     CLD
ØF11         ;WIRKSAMEN T2-INHALT $FFØØ UM
ØF11         ;ZYKLENZAHL ZW. AUFTRETEN UND
ØF11         ;PROZESSORREAKTION AUF
ØF11         ;INTERRUPT SOWIE ZYKLENZAHL DER MONI-
ØF11         ,TORINTERRUPTROUTINE KORRIGIEREN
ØF11         38     SEC
ØF12         A9D3   LDA #$D3
ØF14         EDØ8AØ SBC T2L
ØF17         18     CLC
ØF18         6DØ4Ø1 ADC ZYKZ+4
ØF1B         ;ZYKLUSZAEHLER ERHOEHEN
ØF1B         8DØ4Ø1 STA ZYKZ+4
ØF1E         A9FF   LDA #$FF
ØF2Ø         6DØ31 ADC ZYKZ+3
ØF23         8DØ3Ø1 STA ZYKZ+3
ØF26         9ØØD   BCC LDT2
ØF28         EEØ2Ø1 INC ZYKZ+2
ØF2B         DØØ8   BNE LDT2
ØF2D         EEØ1Ø1 INC ZYKZ+1
ØF3Ø         DØØ   BNE LDT2
ØF32         EEØØØ1 INC ZYKZ
```

```
ØF35          ;T2 STARTEN MESSWIRKSAM WIRD
ØF35          ;INHALT STARTWERT -1Ø=$FFØØ
ØF35 LDT2     A9ØØ    LDA #Ø
ØF37          8DØBAØ STA ACR
ØF3A          A9ØA    LDA #1Ø
ØF3C          8DØ8AØ STA T2L
ØF3F          A9FF    LDA #$FF
ØF41          8DØ9AØ STA T2H
ØF44          68      PLA
ØF45          ;AKKU ZURUECK
ØF45          4Ø      RTI
ØF46          ;ZUR. Z. VORDERGRUNDPROGRAMM
ØF46          ----------------------
ØF46          AUSWERTUNG
ØF46
ØF46 AUSW     Ø8      PHP
ØF47          ;REGISTER RETTEN
ØF47          78      SEI
ØF48          48      PHA
ØF49          A92Ø    LDA #$2Ø
ØF4B          8DØBAØ STA ACR          ;T2 ANHALTEN
ØF4E          8A      TXA
ØF4F          48      PHA
ØF5Ø          98      TYA
ØF51          48      PHA
ØF52          D8      CLD
ØF53          38      SEC
ØF54          ;ZYKLUSZAEHLER ENDWERT BEREITSTELLEN
ØF54          AD0401 LDA ZYKZ+4
ØF57          ED08AØ SBC T2L
ØF5A          8D0401 STA ZYKZ+4
ØF5D          AD001 LDA ZYKZ+3
ØF6Ø          ED09AØ SBC T2H
ØF63          8D001 STA ZYKZ+3
ØF66          AD0201 LDA ZYKZ+2
ØF69          E9ØØ    SBC #Ø
ØF6B          8D0201 STA ZYKZ+2
ØF6E          AD0101 LDA ZYKZ+1
ØF71          E9ØØ    SBC #Ø
ØF7          8D0101 STA ZYKZ+1
ØF76          AD0001 LDA ZYKZ
ØF79          E9ØØ    SBC #Ø
ØF7B          8D0001 STA ZYKZ
ØF7E          ----------------------
ØF7E          HEX-BCD-CODIERUNG
ØF7E
ØF7E          ;ZEITWERT ALS GEPACKTE
ØF7E          ;DEZIMALZAHL AUS HEXA-
ØF7E          ;ZYKLUSZAEHLUNG BERECHNEN
ØF7E          F8      SED
ØF7F          A9ØØ    LDA #Ø
ØF81          A2Ø6    LDX #6
ØF8           ;CODIERALGORITHMUS IST DAS
ØF83          ;HORNER-SCHEMA. 5-BYTE ZYKLUS-
ØF83          ;ZAEHLER IN 7-BYTE ZEIT-
ØF83          ;WERT UEBERSETZEN
ØF83 AAA      9D5ØA4 STA ZEIT,X
ØF86          CA      DEX
```

```
ØF87          1ØFA    BPL AAA
ØF89          AØ28    LDY #4Ø
ØF8B BBB      A2Ø4    LDX #4
ØF8D CCC      3EØØØ1  ROL ZYKZ,X
ØF9Ø          CA      DEX
ØF91          1ØFA    BPL CCC
ØF93          A2Ø6    LDX #6
ØF95 DDD      BD5ØA4  LDA ZEIT,X
ØF98          7D5ØA4  ADC ZEIT,X
ØF9B          9D5ØA4  STA ZEIT,X
ØF9E          CA      DEX
ØF9F          1ØF4    BPL DDD
ØFA1          88      DEY
ØFA2          DØE7    BNE BBB
ØFA4          D8      CLD
ØFA5
ØFA5          ----------------------
ØFA5          AUSGABE
ØFA5
ØFA5
ØFA5          ;7-BYTE ZEITWERT WIRD IN DER FORM
ØFA5          ;T=ZZZZZZZX,XXXXXX S
ØFA5          ;AN DIE AKTIVEN AUSGABEEINHEITEN
ØFA5          ;(DISPLAY/DRUCKER) AUSGEGEBEN,
ØFA5          ;POSITIONEN IN Z MIT FUEHRENDEN
ØFA5          ;NULLEN WERDEN FORTGELASSEN
ØFA5          2Ø44EB JSR CLR
ØFA8          2Ø3EE8 JSR BLANK
ØFAB          A954    LDA #'T
ØFAD          2ØBCE9 JSR OUTALL
ØFBØ          2ØD8E7 JSR EQUAL
ØFB3          A2ØØ    LDX #Ø
ØFB5 BYTE     BD5ØA4  LDA ZEIT,X
ØFB8          DØØ6    BNE ZIFF
ØFBA          E8      INX
ØFBB          EØØ4    CPX #4
ØFBD          DØF6    BNE BYTE
ØFBF          CA      DEX
ØFCØ ZIFF     C91Ø    CMP #$1Ø
ØFC2          BØØ6    BCS DZIFF
ØFC4          2Ø51EA JSR NOUT
ØFC7          18      CLC
ØFC8          9ØØ    BCC *+5
ØFCA DZIFF    2Ø46EA JSR NUMA
ØFCD          E8      INX
ØFCE          EØØ4    CPX #4
ØFDØ          FØØ6    BEQ KOMMA
ØFD2          BD5ØA4  LDA ZEIT,X
ØFD5          18      CLC
ØFD6          9ØF2    BCC DZIFF
ØFD8 KOMMA    A92C    LDA #',
ØFDA          2ØBCE9 JSR OUTALL
ØFDD MMS      BD5ØA4  LDA ZEIT,X
ØFEØ          2Ø46EA JSR NUMA
ØFE3          E8      INX
ØFE4          EØØ7    CPX #7
ØFE6          DØF5    BNE MMS
ØFE8          2ØEE8 JSR BLANK
```

```
ØFEB          A953    LDA #'S
ØFED          20ØBCE9 JSR OUTALL
ØFFØ          68      PLA
ØFF1          ;REGISTER WIEDERHSTELLEN
ØFF1          A8      TAY
ØFF2          68      PLA
ØFF3          AA      TAX
ØFF4          68      PLA
ØFF5          28      PLP
ØFF6          ;D. PROGRAMMSTOPP JSR $E89C
ØFF6          ;UEBERSCHRIEBENER MASCHINENCODE
ØFF6          ;KANN HIER ZWISCHENGESPEICHERT WERDEN
ØFF6          EA      NOP
ØFF7          EA      NOP
ØFF8          EA      NOP
ØFF9          EA      NOP
ØFFA          EA      NOP
ØFFB          6Ø      RTS
ØFFC          ;FORTSETZUNG DES AUFRUFENDEN PROGRAMMES
ØFFC                  .END
ØFFC           ERRORS= ØØØØ


0500 A9 LDA #64      BEISPIEL FÜR EIN MASCHINENPROGRAMM
0502 A2 LDX #A5      30-SECOND TIME DELAY SUBROUTINE,
0504 A0 LDY #EA      EXAMPLE 3-6 AUS SCANLON:'6502 SOFTWARE DESIGN'
0506 CA DEX
0507 D0 BNE 0506
0509 88 DEY
050A D0 BNE 0506
050C 38 SEC
050D E9 SBC #01
050F D0 BNE 0502
0511 60 RTS

0200 20 JSR 0EA0     ANWENDUNG VON LZEIT AUF O.A. PROGRAMM, START D. MESSUNG
0203 20 JSR 0500     AUFRUF DER TIME-DELAY SUBROUTINE
0206 20 JSR E89C     STOPP DER MESSUNG
0209 4C JMP E182     SPRUNG ZUM MONITOR

<*>=200
<G>/                 AUSFÜHRUNG MIT AIM 65 'GO'
 T=30,001113 S       AUSDRUCK ZEITVERZOEGERUNG. MIT LDX #A1 WÜRDE SICH
                     IN SCANLON-BEISPIEL DER RICHTIGE WERT ERGEBEN

10 POKE 4,160             BEISPIELANWENDUNG AUF BASIC
20 POKE 5,14              IM AIM 65. USR VEKTOR VORBEREITEN
30 T=USR(T)               START DER MESSUNG
100 FOR I=1 TO 1000       BERECHNUNG VON PI AUS REIHENSUMME
110 S=S+1/(I*I)
120 NEXT I
130 PRINT "PI=";
140 PRINT SQR(6*S)

1000 Ñ=USR(T0             STOPP DER MESSUNG

RUN
PI= 3.14063806
 T=10,330918 S            AUSDRUCK DER ZEIT BEI PRINTER 'EIN'
 T=9,799630 S             AUSDRUCK DER ZEIT BEI PRINTER = 'AUS', ZEILE 1000
                          DANN ABER 1000 T=USR(T):PRINT!""
```

Peter W. Arps, Hamburg

ON ERROR GO TO

Diese für CBM geschriebene Routine nutzt die Eigenart des BASIC-Interpreters, bei jedem PRINT auf den nächsten Interrupt zu warten. In der modifizierten Interrupt-Routine wird der Stack auf die RTS-Adresse C369 (hex) hin untersucht. Von hier aus wird das Fragezeichen bei Fehlermeldungen ausgegeben. Wenn der Ansprung nicht von der Fehlerroutine her erfolgte, wird die Interruptroutine normal durchlaufen. Anderenfalls wird nach dem Sichern der Fehlernummer (Displacement in die Tabelle auf C192 im X-Register) und der Fehler-Zeilen-Nummer (fast immer) zur Zeilennummer für ON-ERROR verzweigt.

Die Einschränkung 'fast immer' muß leider gemacht werden, da diese Routine bei mehreren aufeinanderfolgenden Fehlern nicht richtig arbeitet. Die 'Trefferquote' kann durch den Einbau einer Verzögerungsschleife (FORI=1TO999:NEXT) in der BASIC-ON-ERROR-Routine erhöht werden. Aus zeitlichen Gründen wurde dieses Verhalten noch nicht näher untersucht. Der Autor wäre für Anregungen aus der Leserschaft dankbar.

```
                        0010              .OS
                        0020              .BA $33A
                        0030              ; **********************
                        0040              ; *  ON-ERROR-GOTO   *
                        0050              ; **********************
                        0060              ;
                        0070              ;
                        0080 ;SYNTAX: SYS(826)ZEILEN#   SETZEN
                        0090 ;        SYS(826)          LOESCHEN
                        0100 ;
                        0110 ; NR. DER FEHLERZEILE: PEEK(1016)+PEEK(1017)*256
                        0120 ; FEHLER-NUMMER:       PEEK(1021)
                        0130              ;
                        0140              ;
                        0150 POINT        .DE $11
                        0160 IRQ.VEK      .DE $90
                        0170 FZEIL.NR     .DE 54
                        0180 SAVE.NR      .DE $3F8
                        0190 ZEIL.NR      .DE $3FE
                        0200 FEHL.NR      .DE $3FD
                        0210 CHRGOT       .DE $0076
                        0220 FETCH        .DE $CC8B
                        0230 IRQ.NORMAL   .DE $E62E
                        0240 INTG         .DE $D6D2
                        0250 ERROR        .DE $C369
                        0260 SUCH.ZEIL    .DE $C52C
                        0270 STACK        .DE $10E
                        0280 RTN.ADR      .DE $105
                        0290 GOTO         .DE $C7A0
                        0300 PRINT.FEHL   .DE $C357
                        0310              ;
033A- 20 76 00          0320 INIT         JSR CHRGOT
033D- F0 12             0330              BEQ RESET      ;AUSSCHALTEN
033F- 20 8B CC          0340              JSR FETCH      ;ZEILEN-NR HOLEN
0342- 20 D2 D6          0350              JSR INTG       ;NACH HEX
0345- 8C FE 03          0360              STY ZEIL.NR
0348- 8D FF 03          0370              STA ZEIL.NR+1
034B- A9 5C             0380              LDA #L,ON.ERROR VEKTOR LADEN
034D- A2 03             0390              LDX #H,ON.ERROR
```

```
034F- D0 04      0400              BNE SET.VEK  ;IMMER
                 0410              ;
0351- A9 2E      0420 RESET        LDA #L,IRQ.NORMAL
0353- A2 E6      0430              LDX #H,IRQ.NORMAL
0355- 78         0440 SET.VEK      SEI
0356- 85 90      0450              STA *IRQ.VEK IRQVEKTOR MODIFIZIEREN
0358- 86 91      0460              STX *IRQ.VEK+1
035A- 58         0470              CLI
035B- 60         0480              RTS
                 0490              ;
035C- BA         0500 ON.ERROR     TSX
035D- BD 0E 01   0510              LDA STACK,X  ;ANSPRUNGS-ADR LADEN
0360- C9 69      0520              CMP #L,ERROR VON ERROR-ROUTINE?
0362- D0 0D      0530              BNE RTN      ;NEIN
0364- BD 0F 01   0540              LDA STACK+1,X
0367- C9 C3      0550              CMP #H,ERROR
0369- D0 06      0560              BNE RTN      ;NEIN
                 0570              ;
036B- A5 7B      0580              LDA *CHRGOT+2 DIRECT-MODE?
036D- C9 02      0590              CMP #2
036F- D0 03      0600              BNE WEITER   ;NEIN
0371- 4C 2E E6   0610 RTN          JMP IRQ.NORMAL IRQ-BEENDIGUNG
                 0620              ;
0374- BA         0630 WEITER       TSX
0375- BD 0A 01   0640              LDA $10A,X
0378- 8D FD 03   0650              STA FEHL.NR  ;FEHLER-NR SICHERN
037B- A2 F8      0660              LDX #$F8
037D- 9A         0670              TXS
037E- A5 36      0680              LDA *FZEIL.NR ZEIL# SICHERN
0380- 8D F8 03   0690              STA SAVE.NR
0383- A5 37      0700              LDA *FZEIL.NR+1
0385- 8D F9 03   0710              STA SAVE.NR+1
0388- AD FE 03   0720              LDA ZEIL.NR  ;NEUE ZEILEN-NR LADEN
038B- AE FF 03   0730              LDX ZEIL.NR+1
038E- 85 11      0740              STA *POINT
0390- 86 12      0750              STX *POINT+1
0392- 20 2C C5   0760              JSR SUCH.ZEIL
0395- 90 03      0770              BCC FEHLER   ;ZEILE NICHT DA
0397- 4C B0 C7   0780              JMP GOTO+3
039A- 20 51 03   0790 FEHLER       JSR RESET
039D- AE FD 03   0800              LDX FEHL.NR
03A0- 4C 57 C3   0810              JMP PRINT.FEHL
                 0820              ;
                 0830              .EN
```

```
LABEL FILE:  [ / = EXTERNAL ]

/POINT=0011          /IRQ.VEK=0090        /FZEIL.NR=0036
/SAVE.NR=03F8        /ZEIL.NR=03FE        /FEHL.NR=03FD
/CHRGOT=0076         /FETCH=CC8B          /IRQ.NORMAL=E62E
/INTG=D6D2           /ERROR=C369          /SUCH.ZEIL=C52C
/STACK=010E          /RTN.ADR=0105        /GOTO=C7AD
/PRINT.FEHL=C357     INIT=033A            RESET=0351
SET.VEK=0355         ON.ERROR=035C        RTN=0371
WEITER=0374          FEHLER=039A
//0000,03A3,03A3
```

#

Dipl.-Ing. Uwe Kornnagel, 6096 Raunheim

Pseudo 16-Bit CPU

Im hier vorliegenden Beispiel handelt es sich um die Implementierung einer Pseudo-16-Bit-CPU auf dem 6502. Diese virtuelle CPU hat den Vorteil, daß sie sich in Ihre Maschinenprogramme einbauen läßt, ohne den Aufbau des 6502-Systems zu verändern. Die CPU besitzt drei 16-Bit-Akkus, mit denen sie alle vier Grundrechenarten ausführen kann. Die Argumente stehen in Akku1 und Akku2, das Ergebnis in Akku3. Der Akku3 dient ebenfalls als Indexregister, das dem Anwender Sprünge und Calls über +/- 32K erlaubt. Alle 16-Bit-Operationen werden mittels JSR XXXX angesprungen. Die Unterprogramme lassen sich in 6 Gruppen mit folgender Struktur einteilen:

1. Virtuelle Akku-Befehle	16 Befehle
2. Virtuelle Memory-Befehle	6 Befehle
3. Virtuelle arithmetische Befehle	5 Befehle
4. Virtuelle JMP-JSR-Befehle	2 Befehle
5. Virtueller Transfer-Befehl	1 Befehl
6. Virtuelles SYSRES -Daten und Sprungleiste	
Virtuell gesamt	30 Befehle

Das Übergaberegister für die 16-Bit-CPU setzt sich aus dem Registerpaar Akku und Y-Register der 6502 zusammen, wobei immer das LOW-Byte im Akku und das HIGH-Byte im Y-Register steht.

Als Beispiel soll der Akku1 mit dem Wert $0320 geladen werden:

```
LDA #$20
LDY #$03
JSR $7DBA
```

Das Programm läßt sich problemlos auf den CBM 3001 umschreiben, indem man statt JMP $D722 bei Adresse $7FF6 ein JMP $E775 schreibt. Eine Umsetzung auf den AIM 65 ist ebenso einfach, es muß nur $D722 in $EA46 (NUMA) und $FFD2 in E9BC (OUTALL) geändert werden.

```
LINE# LOC   CODE        LINE

0002  0000              ;*******************************
0003  0000              ;* CBM 8032 PSEUDO 16-BIT-CPU *
0004  0000              ;*******************************

0006  0000              ;VIRTUELLE CPU =CBM=
0007  0000              ;==================
0008  0000
0009  0000                      * =$7D00        ;START FUER 32K-VERSION
0010  7D00
0011  7D00              ;AKKUS LOESCHEN
0012  7D00              ;==============
0013  7D00
0014  7D00  A9 00       CLRAK1 LDA #$00
0015  7D02  8D D9 7F            STA AKK1
0016  7D05  8D DA 7F            STA AKK1+1
0017  7D08  60                  RTS
0018  7D09
```

```
0019  7D09  A9 00     CLRAK2 LDA #$00
0020  7D0B  8D DB 7F          STA AKK2
0021  7D0E  8D DC 7F          STA AKK2+1
0022  7D11  60                RTS
0023  7D12
0024  7D12  20 00 7D  CLRAKS JSR CLRAK1
0025  7D15  20 09 7D          JSR CLRAK2
0026  7D18
0027  7D18  A9 00     CLRAK3 LDA #$00
0028  7D1A  8D DD 7F          STA AKK3
0029  7D1D  8D DE 7F          STA AKK3+1
0030  7D20  60                RTS
0031  7D21
0032  7D21
0033  7D21                    ;AKKUS TRANSFER
0034  7D21                    ;==============
0035  7D21
0036  7D21  AD D9 7F  TRA12  LDA AKK1
0037  7D24  8D DB 7F          STA AKK2
0038  7D27  AD DA 7F          LDA AKK1+1
0039  7D2A  8D DC 7F          STA AKK2+1
0040  7D2D  60                RTS
0041  7D2E
0042  7D2E  AD D9 7F  TRA13  LDA AKK1
0043  7D31  8D DD 7F          STA AKK3
0044  7D34  AD DA 7F          LDA AKK1+1
0045  7D37  8D DE 7F          STA AKK3+1
0046  7D3A  60                RTS
0047  7D3B
0048  7D3B  AD DB 7F  TRA21  LDA AKK2
0049  7D3E  8D D9 7F          STA AKK1
0050  7D41  AD DC 7F          LDA AKK2+1
0051  7D44  8D DA 7F          STA AKK1+1
0052  7D47  60                RTS
0053  7D48
0054  7D48  AD DB 7F  TRA23  LDA AKK2
0055  7D4B  8D DD 7F          STA AKK3
0056  7D4E  AD DC 7F          LDA AKK2+1
0057  7D51  8D DE 7F          STA AKK3+1
0058  7D54  60                RTS
0059  7D55
0060  7D55  AD DD 7F  TRA31  LDA AKK3
0061  7D58  8D D9 7F          STA AKK1
0062  7D5B  AD DE 7F          LDA AKK3+1
0063  7D5E  8D DA 7F          STA AKK1+1
0064  7D61  60                RTS
0065  7D62
0066  7D62  AD DD 7F  TRA32  LDA AKK3
0067  7D65  8D DB 7F          STA AKK2
0068  7D68  AD DE 7F          LDA AKK3+1
0069  7D6B  8D DC 7F          STA AKK2+1
0070  7D6E  60                RTS
0071  7D6F
0072  7D6F
0073  7D6F                    ;AKKU AUSTAUSCH
0074  7D6F                    ;==============
0075  7D6F
0076  7D6F  AD D9 7F  CHA12  LDA AKK1
```

```
0077  7D72  AE DB 7F          LDX AKK2
0078  7D75  8E D9 7F          STX AKK1
0079  7D78  8D DB 7F          STA AKK2
0080  7D7B  AD DA 7F          LDA AKK1+1
0081  7D7E  AE DC 7F          LDX AKK2+1
0082  7D81  8E DA 7F          STX AKK1+1
0083  7D84  8D DC 7F          STA AKK2+1
0084  7D87  60                RTS
0085  7D88
0086  7D88  AD D9 7F    CHA13 LDA AKK1
0087  7D8B  AE DD 7F          LDX AKK3
0088  7D8E  8E D9 7F          STX AKK1
0089  7D91  8D DD 7F          STA AKK3
0090  7D94  AD DA 7F          LDA AKK1+1
0091  7D97  AE DE 7F          LDX AKK3+1
0092  7D9A  8E DA 7F          STX AKK1+1
0093  7D9D  8D DE 7F          STA AKK3+1
0094  7DA0  60                RTS
0095  7DA1
0096  7DA1  AD DB 7F    CHA23 LDA AKK2
0097  7DA4  AE DD 7F          LDX AKK3
0098  7DA7  8E DB 7F          STX AKK2
0099  7DAA  8D DD 7F          STA AKK3
0100  7DAD  AD DC 7F          LDA AKK2+1
0101  7DB0  AE DE 7F          LDX AKK3+1
0102  7DB3  8E DC 7F          STX AKK2+1
0103  7DB6  8D DE 7F          STA AKK3+1
0104  7DB9  60                RTS
0105  7DBA
0106  7DBA
0107  7DBA                    ;LDA AKKU #
0108  7DBA                    ;==========
0109  7DBA
0110  7DBA                    ;DATEN LOW  = R(A)
0111  7DBA                    ;      HIGH = R(Y)
0112  7DBA
0113  7DBA  8D D9 7F    LDAI1 STA AKK1
0114  7DBD  8C DA 7F          STY AKK1+1
0115  7DC0  60                RTS
0116  7DC1
0117  7DC1  8D DB 7F    LDAI2 STA AKK2
0118  7DC4  8C DC 7F          STY AKK2+1
0119  7DC7  60                RTS
0120  7DC8
0121  7DC8  8D DD 7F    LDAI3 STA AKK3
0122  7DCB  8C DE 7F          STY AKK3+1
0123  7DCE  60                RTS
0124  7DCF
0125  7DCF
0127  7DCF                    ;LDA AKKU VON ADRESSE
0128  7DCF                    ;====================
0129  7DCF
0130  7DCF                    ;ADRESSE LOW = R(A)
0131  7DCF                    ;        HIGH= R(Y)
0132  7DCF
0133  7DCF  20 29 7E    LDA1  JSR SETPN
0134  7DD2  B1 1F              LDA (#1F),Y
0135  7DD4  8D D9 7F          STA AKK1
```

```
0136  7DD7  C8              INY
0137  7DD8  B1 1F           LDA (#1F),Y
0138  7DDA  8D DA 7F        STA AKK1+1
0139  7DDD  60              RTS
0140  7DDE
0141  7DDE  20 29 7E     LDA2  JSR SETPN
0142  7DE1  B1 1F           LDA (#1F),Y
0143  7DE3  8D DB 7F        STA AKK2
0144  7DE6  C8              INY
0145  7DE7  B1 1F           LDA (#1F),Y
0146  7DE9  8D DC 7F        STA AKK2+1
0147  7DEC  60              RTS
0148  7DED
0149  7DED  20 29 7E     LDA3  JSR SETPN
0150  7DF0  B1 1F           LDA (#1F),Y
0151  7DF2  8D DD 7F        STA AKK3
0152  7DF5  C8              INY
0153  7DF6  B1 1F           LDA (#1F),Y
0154  7DF8  8D DE 7F        STA AKK3+1
0155  7DFB  60              RTS
0156  7DFC
0157  7DFC
0158  7DFC                 ;SPEICHER AKKU AUF ADRESSE
0159  7DFC                 ;=========================
0160  7DFC
0161  7DFC                 ;ADRESSR LOW = R(A)
0162  7DFC                 ;         HIGH= R(Y)
0163  7DFC
0164  7DFC  20 29 7E     STA1  JSR SETPN
0165  7DFF  AD D9 7F           LDA AKK1
0166  7E02  91 1F           STA (#1F),Y
0167  7E04  C8              INY
0168  7E05  AD DA 7F        LDA AKK1+1
0169  7E08  91 1F           STA (#1F),Y
0170  7E0A  60              RTS
0171  7E0B
0172  7E0B  20 29 7E     STA2  JSR SETPN
0173  7E0E  AD DB 7F           LDA AKK2
0174  7E11  91 1F           STA (#1F),Y
0175  7E13  C8              INY
0176  7E14  AD DC 7F        LDA AKK2+1
0177  7E17  91 1F           STA (#1F),Y
0178  7E19  60              RTS
0179  7E1A
0180  7E1A  20 29 7E     STA3  JSR SETPN
0181  7E1D  AD DD 7F           LDA AKK3
0182  7E20  91 1F           STA (#1F),Y
0183  7E22  C8              INY
0184  7E23  AD DE 7F        LDA AKK3+1
0185  7E26  91 1F           STA (#1F),Y
0186  7E28  60              RTS
0187  7E29
0188  7E29  85 1F        SETPN  STA #1F
0189  7E2B  84 20           STY #20
0190  7E2D  A0 00           LDY ##00
0191  7E2F  60              RTS
0192  7E30
0193  7E30
```

```
0195  7E30                    ;ADDIERE AKK3 = AKK1 + AKK2
0196  7E30                    ;==========================
0197  7E30
0198  7E30  18         ADDR   CLC
0199  7E31  AD D9 7F          LDA AKK1
0200  7E34  6D DB 7F          ADC AKK2
0201  7E37  8D DD 7F          STA AKK3
0202  7E3A  AD DA 7F          LDA AKK1+1
0203  7E3D  6D DC 7F          ADC AKK2+1
0204  7E40  8D DE 7F          STA AKK3+1
0205  7E43  60                RTS
0206  7E44
0207  7E44                    ;SUBTRAIERE AKK3 = AKK1 - AKK2
0208  7E44                    ;=============================
0209  7E44
0210  7E44  38         SUBB   SEC
0211  7E45  AD D9 7F          LDA AKK1
0212  7E48  ED DB 7F          SBC AKK2
0213  7E4B  8D DD 7F          STA AKK3
0214  7E4E  AD DA 7F          LDA AKK1+1
0215  7E51  ED DC 7F          SBC AKK2+1
0216  7E54  8D DE 7F          STA AKK3+1
0217  7E57  60                RTS
0218  7E58
0219  7E58                    ;MULTIPLIZIERE AKK3 = AKK1 * AKK2
0220  7E58                    ;================================
0221  7E58
0222  7E58  20 18 7D   MULT   JSR CLRAK3
0223  7E5B  A2 10             LDX #16
0224  7E5D  4E DC 7F   MLOOP  LSR AKK2+1
0225  7E60  6E DB 7F          ROR AKK2
0226  7E63  90 13             BCC NOADD
0227  7E65  18                CLC
0228  7E66  AD DD 7F          LDA AKK3
0229  7E69  6D D9 7F          ADC AKK1
0230  7E6C  8D DD 7F          STA AKK3
0231  7E6F  AD DE 7F          LDA AKK3+1
0232  7E72  6D DA 7F          ADC AKK1+1
0233  7E75  8D DE 7F          STA AKK3+1
0234  7E78  0E D9 7F   NOADD  ASL AKK1
0235  7E7B  2E DA 7F          ROL AKK1+1
0236  7E7E  CA                DEX
0237  7E7F  D0 DC             BNE MLOOP
0238  7E81  60                RTS
0239  7E82
0240  7E82                    ;VERGLEICHE AKK2 MIT AKK1
0241  7E82                    ;========================
0242  7E82
0243  7E82  A9 00    COMPA   LDA #0
0244  7E84  48                PHA
0245  7E85  28                PLP
0246  7E86  AD DA 7F          LDA AKK1+1
0247  7E89  CD DC 7F          CMP AKK2+1
0248  7E8C  D0 06             BNE CMPEN
0249  7E8E  AD D9 7F          LDA AKK1
0250  7E91  CD DB 7F          CMP AKK2
0251  7E94  60          CMPEN  RTS
0252  7E95
```

```
0253  7E95                    ;DIVIDIERE AKK3 = AKK1 / AKK2
0254  7E95                    ;============================
0256  7E95  20 18 7D  DIVID   JSR CLRAK3
0257  7E98  18                CLC
0258  7E99  AD DB 7F           LDA AKK2
0259  7E9C  6D DC 7F           ADC AKK2+1
0260  7E9F  F0 4B              BEQ DIVER
0261  7EA1  A2 00              LDX #0
0262  7EA3  E8        DVADJ   INX
0263  7EA4  0E DB 7F           ASL AKK2
0264  7EA7  2E DC 7F           ROL AKK2+1
0265  7EAA  20 82 7E           JSR COMPA
0266  7EAD  B0 F4              BCS DVADJ
0267  7EAF  F0 31              BEQ ADJ1
0268  7EB1  CA                 DEX
0269  7EB2  4E DC 7F           LSR AKK2+1
0270  7EB5  6E DB 7F           ROR AKK2
0271  7EB8  20 82 7E  DVLOP   JSR COMPA
0272  7EBB  08                 PHP
0273  7EBC  90 13              BCC NOSUB
0274  7EBE  38                 SEC
0275  7EBF  AD D9 7F           LDA AKK1
0276  7EC2  ED DB 7F           SBC AKK2
0277  7EC5  8D D9 7F           STA AKK1
0278  7EC8  AD DA 7F           LDA AKK1+1
0279  7ECB  ED DC 7F           SBC AKK2+1
0280  7ECE  8D DA 7F           STA AKK1+1
0281  7ED1  28        NOSUB   PLP
0282  7ED2  2E DD 7F           ROL AKK3
0283  7ED5  2E DE 7F           ROL AKK3+1
0284  7ED8  4E DC 7F           LSR AKK2+1
0285  7EDB  6E DB 7F           ROR AKK2
0286  7EDE  CA                 DEX
0287  7EDF  10 D7              BPL DVLOP
0288  7EE1  60                 RTS
0289  7EE2  2E DD 7F  ADJ1     ROL AKK3
0290  7EE5  2E DE 7F           ROL AKK3+1
0291  7EE8  CA                 DEX
0292  7EE9  D0 F7              BNE ADJ1
0293  7EEB  60                 RTS
0294  7EEC  A2 00     DIVER    LDX #0
0295  7EEE  BD E1 7F  ERLOP    LDA ERRT,X
0296  7EF1  F0 06              BEQ EREND
0297  7EF3  20 F9 7F           JSR PRINT
0298  7EF6  E8                 INX
0299  7EF7  D0 F5              BNE ERLOP
0300  7EF9  68        EREND    PLA
0301  7EFA  85 1F              STA $1F
0302  7EFC  68                 PLA
0303  7EFD  85 20              STA $20
0304  7EFF  38                 SEC
0305  7F00  A5 1F              LDA $1F
0306  7F02  E9 02              SBC #2
0307  7F04  85 1F              STA $1F
0308  7F06  A5 20              LDA $20
0309  7F08  E9 00              SBC #0
0310  7F0A  20 F6 7F           JSR WROB
0311  7F0D  A5 1F              LDA $1F
0312  7F0F  20 F6 7F           JSR WROB
```

```
0316  7F12                    ;JMP RELATIV PC + AKK3
0317  7F12                    ;====================
0318  7F12
0319  7F12                    ;OFSET IN AKK3
0320  7F12                    ;AUFRUF JSR RJMP
0321  7F12
0322  7F12  68               RJMP   PLA
0323  7F13  85 1F                    STA $1F
0324  7F15  68                      PLA
0325  7F16  85 20                    STA $20
0326  7F18  E6 1F            RJMP0  INC $1F
0327  7F1A  D0 02                    BNE RJMP1
0328  7F1C  E6 20                    INC $20
0329  7F1E  18               RJMP1  CLC
0330  7F1F  A5 1F                    LDA $1F
0331  7F21  6D DD 7F                ADC AKK3
0332  7F24  85 1F                    STA $1F
0333  7F26  A5 20                    LDA $20
0334  7F28  6D DE 7F                ADC AKK3+1
0335  7F2B  85 20                    STA $20
0336  7F2D  6C 1F 00                JMP ($1F)
0337  7F30
0338  7F30                    ;JSR RELATIV PC + AKK3
0339  7F30                    ;====================
0340  7F30
0341  7F30                    ;OFSET IN AKK3
0342  7F30                    ;AUFRUF JSR RJSR
0343  7F30
0344  7F30  68               RJSR   PLA
0345  7F31  85 1F                    STA $1F
0346  7F33  68                      PLA
0347  7F34  85 20                    STA $20
0348  7F36  48                      PHA
0349  7F37  A5 1F                    LDA $1F
0350  7F39  48                      PHA
0351  7F3A  4C 18 7F                JMP RJMP0
0352  7F3D
0353  7F3D
0355  7F3D                    ;BLOCKTRANSFER
0356  7F3D                    ;=============
0357  7F3D
0358  7F3D                    ;STARTADRESSE IN AKK1
0359  7F3D                    ;ENDADRESSE   IN AKK2
0360  7F3D                    ;NEUE ANFAGSA IN AKK3
0361  7F3D
0362  7F3D  AD DA 7F          TRANSF LDA AKK1+1
0363  7F40  CD DE 7F                 CMP AKK3+1
0364  7F43  D0 06                    BNE TRAN1
0365  7F45  AD D9 7F                 LDA AKK1
0366  7F48  CD DD 7F                 CMP AKK3
0367  7F4B  90 35            TRAN1  BCC TNEG
0368  7F4D  AD D9 7F                 LDA AKK1
0369  7F50  85 5C                    STA $5C
0370  7F52  AD DA 7F                 LDA AKK1+1
0371  7F55  85 5D                    STA $5D
0372  7F57  AD DD 7F                 LDA AKK3
0373  7F5A  85 55                    STA $55
0374  7F5C  AD DE 7F                 LDA AKK3+1
```

```
0375  7F5F  85 56        STA $56
0376  7F61  A0 00  TRL1   LDY #0
0377  7F63  B1 5C        LDA ($5C),Y
0378  7F65  91 55        STA ($55),Y
0379  7F67  E6 55        INC $55
0380  7F69  D0 02        BNE TRC1
0381  7F6B  E6 56        INC $56
0382  7F6D  E6 5C  TRC1  INC $5C
0383  7F6F  D0 02        BNE TRC2
0384  7F71  E6 5D        INC $5D
0385  7F73  A5 5D  TRC2  LDA $5D
0386  7F75  CD DC 7F     CMP AKK2+1
0387  7F78  D0 E7        BNE TRL1
0388  7F7A  A5 5C        LDA $5C
0389  7F7C  CD DB 7F     CMP AKK2
0390  7F7F  D0 E0        BNE TRL1
0391  7F81  60           RTS
0392  7F82  38  TNEG    SEC
0393  7F83  AD DB 7F     LDA AKK2
0394  7F86  ED D9 7F     SBC AKK1
0395  7F89  85 55        STA $55
0396  7F8B  AD DC 7F     LDA AKK2+1
0397  7F8E  ED DA 7F     SBC AKK1+1
0398  7F91  85 56        STA $56
0399  7F93  18           CLC
0400  7F94  AD DD 7F     LDA AKK3
0401  7F97  65 55        ADC $55
0402  7F99  85 55        STA $55
0403  7F9B  AD DE 7F     LDA AKK3+1
0404  7F9E  65 56        ADC $56
0405  7FA0  85 56        STA $56
0406  7FA2  AD DB 7F     LDA AKK2
0407  7FA5  85 5C        STA $5C
0408  7FA7  AD DC 7F     LDA AKK2+1
0409  7FAA  85 5D        STA $5D
0410  7FAC  A0 00  TRL2  LDY #0
0411  7FAE  B1 5C        LDA ($5C),Y
0412  7FB0  91 55        STA ($55),Y
0413  7FB2  38           SEC
0414  7FB3  A5 55        LDA $55
0415  7FB5  E9 01        SBC #1
0416  7FB7  85 55        STA $55
0417  7FB9  A5 56        LDA $56
0418  7FBB  E9 00        SBC #0
0419  7FBD  85 56        STA $56
0420  7FBF  38           SEC
0421  7FC0  A5 5C        LDA $5C
0422  7FC2  E9 01        SBC #1
0423  7FC4  85 5C        STA $5C
0424  7FC6  A5 5D        LDA $5D
0425  7FC8  E9 00        SBC #0
0426  7FCA  85 5D        STA $5D
0427  7FCC  CD DA 7F     CMP AKK1+1
0428  7FCF  D0 DB        BNE TRL2
0429  7FD1  A5 5C        LDA $5C
0430  7FD3  CD D9 7F     CMP AKK1
0431  7FD6  D0 D4        BNE TRL2
0432  7FD8  60           RTS
```

```
0436  7FD9                  ;POINTER
0437  7FD9                  ;=======
0438  7FD9
0439  7FD9  00 00           AKK1    .WORD 0
0440  7FDB  00 00           AKK2    .WORD 0
0441  7FDD  00 00           AKK3    .WORD 0
0442  7FDF  00 00                   .WORD 0
0443  7FE1
0444  7FE1
0445  7FE1                  ;FEHLERMELDUNG
0446  7FE1                  ;=============
0447  7FE1
0448  7FE1  0D              ERRT    .BYT 13,'? DIV. /0 IN ADR. '
0448  7FE2  3F 20
0449  7FE4  00 00                   .WORD 0
0450  7FE6
0451  7FF6
0452  7FF6                  ;CBM 8032 VIRTUELL V1.0
0453  7FF6                  ;======================
0454  7FF6
0455  7FF6  4C 22 D7        WROB    JMP $D722       ;AUSGABE AKKU HEXADEZIMA
0456  7FF9  4C D2 FF        PRINT   JMP $FFD2       ;AUSGABE AKKU ALS ASCII
0457  7FFC                         .END

ERRORS = 0000


SYMBOL TABLE

SYMBOL VALUE
 ADDR     7E30    ADJ1     7EE2    AKK1     7FD9    AKK2     7FDB
 AKK3     7FDD    CHA12    7D6F    CHA13    7D88    CHA23    7DA1
 CLRAK1   7D00    CLRAK2   7D09    CLRAK3   7D18    CLRAKS   7D12
 CMPEN    7E94    COMPA    7E82    DIVER    7EEC    DIVID    7E95
 DVADJ    7EA3    DVLOP    7EB8    EREND    7EF9    ERLOP    7EEE
 ERRT     7FE1    LDA1     7DCF    LDA2     7DDE    LDA3     7DED
 LDAI1    7DBA    LDAI2    7DC1    LDAI3    7DC8    MLOOP    7E5D
 MULT     7E58    NOADD    7E78    NOSUB    7ED1    PRINT    7FF9
 RJMP     7F12    RJMP0    7F18    RJMP1    7F1E    RJSR     7F30
 SETPN    7E29    STA1     7DFC    STA2     7E0B    STA3     7E1A
 SUBB     7E44    TNEG     7F82    TRA12    7D21    TRA13    7D2E
 TRA21    7D3B    TRA23    7D48    TRA31    7D55    TRA32    7D62
 TRAN1    7F4B    TRANSF   7F3D    TRC1     7F6D    TRC2     7F73
 TRL1     7F61    TRL2     7FAC    WROB     7FF6
```

#

Kleinanzeigen

AIM 65, komplette Sätze der 3 Handbücher + Schaltplan englisch, auch in größerer Menge zu verkaufen, DM 30,-/Satz. H.-J. Regge, Bremen, Tel.: 0421 - 71 114.

BASIC-ROMs für AIM 65 zu verkaufen. Paul Rodrigo, Bachemer Str. 68, 5000 Köln 41.

AIM 65 Buffer-Platine in Industriequalität zur Bufferung von Daten- und Adreßleitungen. Nur zwischen AIM 65 und CPU 6502 stecken. Neuer Preis DM 19,60 netto f. Platine ohne Buffer (2x 74LS244, 1x 74LS245) einschl. Schaltunterlagen. Bernhard Kokula, Wredestr. 17, 6700 Ludwigshafen, Tel.: 0621 - 51 82 29.

AIM 65 zu verkaufen. 4K RAM, 2K BASIC-Erweiterung (Funkschau) auf Cassette, Assembler. Heinz-Gerd Paßen, Pappelweg 28, 4422 Ahaus-Wessum, Tel.: 0 25 61 - 79 64.

Dipl.-Math. A. Quint, 6200 Wiesbaden

Garbage Collection Routinen im CBM-BASIC

Für einen String werden folgende Speicherplätze reserviert:

bei einfachen Variablen	7 Bytes pro String
bei Feldern	3 Bytes pro String.

Beispiel: Gibt man über den Bildschirm ein:

AA $="01234"

so wird im Bereich der Variablen folgendes abgelegt:

$41, C1, 05, FB, 7F, 00, 00.

Dabei bedeuten

Byte 1+2:	verschlüsselter Name, Byte 1 erster Buchstabe, Byte 2 zweiter Buchstabe + $80
Byte 3:	Länge des Strings
Byte 4+5:	Startadresse der Zeichenfolge (lo, hi)

Oberhalb des Bereichs der Felder steht dann ab $ 7FFB die Zeichenfolge $30, 31, 32, 33, 34.

Veränderungen an Strings.

Als Beispiel soll folgende Zeile dienen:

```
(ZEILEN-NR.) B2$ = LEFT$(AA$,4) +CHR$(13) + RIGHT$(AA$,1)
```

Für den Beginn der Speicherung der Zeichenfolgen gibt es einen Pointer in $30, 31. Davor wird die neue Zeichenfolge für B2 $, also $30, 31, 32, 33, 0D, 34 abgelegt. Für die Verkettung wird dabei Platz benötigt. Für den Pointer auf den Beginn der Speicherung der Zeichenfolgen ergibt sich damit:

vorher	$7FFB
nachher	$7FEA.

In der Beschreibung der Variablen B2 $ steht dann $ 42, B2, 06, EA, 7F.

Die Speicherung der Zeichenfolgen wird von oben begrenzt durch die höchste RAM-Adresse (Pointer in $34, 35) und von unten begrenzt durch das Ende der Felder (Pointer in $2E,2F). Sobald der Bereich für die Speicherung einer neuen Zeichenfolge eine Schranke unterschreitet, muß der Bereich der gespeicherten Zeichenfolgen zusammengeschoben werden. Gelingt dies nicht, so antwortet der Computer mit einer Meldung OUT OF MEMORY.

Die benötigte Routine beginnt in $ D400 und arbeitet nach folgendem Verfahren: Aus der Menge der Strings wird die Zeichenfolge mit der höchsten Startadresse gesucht. Dazu wird der Bereich der Variablen durchlaufen (Pointer in $ 2A, 2B = Beginn und $ 2E, 2F = Ende). Die gefundene Zeichenfolge wird verschoben und der Pointer auf ihren Beginn korrigiert. Danach wird der Bereich der Variablen erneut durchlaufen. Diesmal wird die Zeichenfolge in der nächsthöheren Adresse gesucht usw..

Die Routine endet, wenn keine Zeichenfolge mit niedrigerer Adresse gefunden werden kann. Dabei werden die im Programm stehenden Zeichenfolgen nicht berücksichtigt. - Man erkennt, daß bei dieser Rotine eine Doppelschleife (ohne Verschieben) programmiert wurde. Dies deckt sich auch mit Testergebnissen, die in Abhängigkeit von der Anzahl der zu untersuchenden Strings einen quadratischen Zeitbedarf der Routine lieferten. Es konnte gelegentlich vorkommen, daß der Computer für 10 Minuten 'tot' war und danach nur für einige Minuten arbeitete, um sich erneut an die 'Stringmüllbeseitigung' zu begeben.

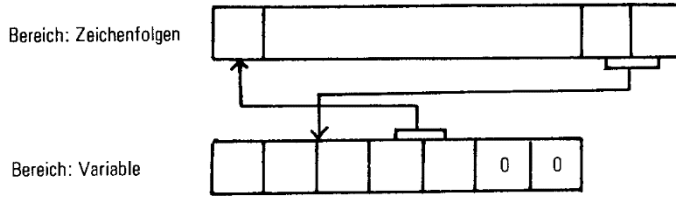
Änderungen bei der 8000er Serie.

Im Bereich der Variablen werden die gleichen Informationen gespeichert. Bei der Speicherung der Zeichenfolge wird ein Pointer ('Link Bytes') angehängt. Dort steht also jetzt $30, 31, 32, 33, 34, 4F, 07. Der Pointer verweist auf die Adresse $074F, die Adresse der Speicherung des Pointers auf den Beginn der Zeichenfolge (genauer: auf die Längenangabe).

Es existieren also 2 Pointer:

Pointer auf den Beginn der Zeichenfolge. Er steht im Bereich der Variablen.

Pointer auf den erstgenannten Pointer. Er steht am Ende der Zeichenfolge.



Diesen zweiten Pointer gibt es natürlich nicht, wenn die Zeichenfolge im Programmtext steht.

Die Veränderung eines Strings führt wie bei der 3000er Serie zu seiner Neuanlage. Zusätzlich wird der Pointer an der alten Zeichenfolge wie folgt geändert:

Lo	in Länge des Strings
Hi	in $FF als Hinweis: 'Müll'.

Bei der notwendigen Garbage-Collection (Routine beginnt in $C66A) wird der Bereich der Zeichenfolgen von oben nach unten durchgesucht. Entweder ist eine Zeichenfolge Müll, dann kann man sie überspringen (Länge bekannt) oder sie wird noch benötigt. Dann kann man sie verschieben und sofort den Pointer auf ihren Beginn korrigieren. Diese Routine durchläuft also den Bereich der eigentlichen Zeichenfolgen, während die Routine in der 3000er Serie den Bereich der Variablen durchlief. Dieses Verfahren ist wesentlich schneller. Es ist kaum möglich, diese Routine länger als 1 Sekunde zu beschäftigen.

Neben diesem großen Vorteil muß man zwei Nachteile akzeptieren:

Jeder String benötigt 2 Bytes mehr.

Eine Anweisung der Form
führt jetzt zu einer Neuanlage der Zeichenfolge, weil sonst der Pointer in die Variablen nicht eindeutig wäre.

Diese Nachteile werden durch die erhöhte Geschwindigkeit mehr als ausgeglichen. Außerdem verschwendet eine Anweisung der Form

```
(Z.-NR.) B2$=LEFT$(AA$,4)+CHR$(13)+RIGHT$(AA$,1)
```

weniger Platz als in der 3000er Seirie.

Nachtrag: Das in Heft 16 des 65xx MICRO MAG beschriebene Sortierprogramm läuft nicht auf der 8000er Serie, weil die Zusatzinformation in den Pointern nicht berücksichtigt wird.

#

Kleinanzeigen

Verkaufe AIM 65 mit BASIC + Assembler im prof. Gehäuse (Sykologic) mit Netzgerät DM 1350,-, 16 K RAM DM 300,-, KTM-2 Video-Interface im Gehäuse DM 500,-, ev. auch PL/65, FORTH. Telefon (nach 19.30 Uhr) 0931 - 28 27 27.

Wir suchen für CBM 3032: 1. Banner-Programm, 2. Softwarepaket für Theaterkasse, Verwaltung von Veranstaltungen, Kartenvorverkauf usw.. Klaus Schuenemann, Köln, Tel.: 0221 - 23 60 47.

Suche RENUMBER-Programm in BASIC für AIM 65. Gerhards, Tel.: 021 71 - 43 051.

Klaus Flesch Badbergstr. 32, 7818 Oberbergen

Assoziative Tabellen (CBM)

Das nachstehende Programm ist in Zusammenarbeit mit BASIC 3.0 ein Verwaltungsprogramm für eine Doppeltabelle, in der zusammengehörige Begriffe (Textketten) aufgenommen und wieder aufgefunden werden können. Typische Anwendungen liegen z.B. in einem Telefonregister, in dem die Begriffspaare Rufnummer und Name verwaltet werden, in einer Mitgliederdatei oder auch in einer Übersetzungshilfe für fremde Sprachen.

Zur Abspeicherungsweise ist zu bemerken, daß eine verdichtete Darstellung für beide assoziative Begriffe gewählt wurde bei der gleiche Wortanfänge nur einmal abgespeichert werden. Eine später hinzukommende Textkette benutzt also den bereits abgespeicherten Wortstamm, für sie wird nur noch die abweichende Folgekette abgespeichert, zusammen mit einer Referenz, ab welcher Speicherstelle im Wortbeginn die Abweichung auftrat. Bei der Eintragung eines neuen Wortes wird dieses also mit den bereits bestehenden Einträgen verglichen. Dabei wird zunächst die Kennzeichnung für einen Wortanfang, eine hex 00 gesucht. Wenn sich dahinter weitere Übereinstimmungen ergeben, wird bis zur ersten Nichtübereinstimmung weiter verglichen. Nach dieser Suche interssieren noch Referenzen im weiteren Speicherbereich auf die Abbruchstelle des Vergleiches, die als Hexbytes mit Adresse Low und Adresse High+hex 80 codiert sind (Bit7=1, zur Markierung). An solchen Stellen wird ebenfalls weitergesucht. Danach steht in den symbolischen Adressen S1 und S1+1 die Endadresse des schon abgelegten Stringrumpfes.

Zu den Inkrementierungsroutinen ist zu bemerken, daß bei Erreichen des Tabellen- oder Speicherendes durch zweimaliges PLA und RTS zum übergeordneten Caller zurückgekehrt wird. Zur Programmbedienung: Mit dem ersten Befehl des Listings bzw. durch ein entsprechendes POKE unter BASIC wird der Vektor der USR-Funktion auf hex 7300, den Beginn des Maschinenprogrammes gerichtet. Danach werden die Funktionen dieses Programmes in BASIC mit dem Befehl A=USR(Argument), A$ B$ aufgerufen. Die übergebenen Argumente lösen dabei folgende Funktionen aus:

Argument	0	Schreibe Wort in A$, B$
	1, 3	Suche Wort und Übersetzung in A$
	2	Suche weiteres Wort in A$

Nach seiner Ausführung liefert das Maschinenprogramm folgende Werte in der Variablen A ab:

Option	0	A=2
	1	A=2, Wort nicht gefunden
		A=3, keine weitere Übersetzung
		A=4, Wort gefunden
	2	wie vor.

Textketten werden ab Adresse hex 1000 abgelegt, so daß Platz für kleinere BASIC Programme im davorliegenden Bereich bleibt. Man kann die Startadresse dieses Textspeichers ggfs. anpassen. Ein Herauslösen einer einmal eingespeicherten Textkette ist nicht möglich, es wurde auch noch kein Verbund zu nicht speicherresidenten Textblöcken hergestellt, der Autor würde sich jedoch über Anregungen zur Erweiterung dieses Speicherverfahrens freuen.

Ein Beispiel: Es sollen folgende Begriffspaare abgespeichert werden:

1. MOON, MOND
2. APPLE, APFEL
3. MONDAY, MONTAG

Dann ergibt sich nachfolgender Aufbau der Tabelle im Textspeicher:

```
$00 M O O N $02 $90 N D $00 A P P L E $0B $90 F E L
$08 $90 A Y $07 $90 T A G
```

```
                        ORG $0000
0000: 4C0073            JMP START
                        ORG $7300
7300: 20D2D6 START      JSR FACC
7303: A511              LDA $11
7305: 2903              AND #03
7307: 8DF875            STA STATUS
730A: A902              LDA #2
730C: 48                PHA
730D: A920              LDA #$20
730F: 85FB              STA S1
7311: A903              LDA #$03
7313: 85FC              STA S1+1
7315: A019              LDY #$19
7317: A920              LDA #$20
7319: 91FB    LOESCH    STA (S1),Y
731B: 88                DEY
731C: 10FB              BPL LOESCH
731E: 207000  INIT      JSR CHGET
7321: 206DCF            JSR VARSUCH
7324: 2090CC            JSR STRTEST
7327: 68                PLA
7328: AA                TAX
7329: A544              LDA $44
732B: 9DFC75            STA WORD1,X
732E: A545              LDA $45
7330: 9DFD75            STA WORD1+1,X
7333: B144              LDA ($44),Y
7335: CA                DEX
7336: CA                DEX
7337: 8A                TXA
7338: 48                PHA
7339: F0E3              BEQ INIT
733B: 68                PLA
733C: A202              LDX #2
733E: 205773            JSR SUB
7341: A900              LDA #0
7343: 8507              STA $7
7345: ADF875            LDA STATUS
7348: F007              BEQ V1
734A: C902              CMP #2
734C: F006              BEQ V2
734E: 4C6D73            JMP UEBERSE
7351: 4CF474  V1        JMP SCHREIB
7354: 4C8573  V2        JMP UEBC
              *
              *SUB      HOHLT ADR(95,96),
              *         LEN(94)  X=0 OR X=2
7357: BDFC75  SUB       LDA WORD1,X
735A: 8561              STA 97
735C: BDFD75            LDA WORD1+1,X
735F: 8562              STA 98
7361: A002              LDY #2
7363: B161    INIT1     LDA (97),Y
7365: 995E00            STA 94,Y
7368: 88                DEY
7369: 10F8              BPL INIT1
736B: C8                INY
736C: 60      ENDE      RTS
                        *
                        *         UEBERSETZER
                        *
736D: 20AA73  UEBERSE   JSR SUCH
7370: 202874            JSR TRANSFER
7373: 9030              BCC NOTFOUND
7375: 206274            JSR UEBINIT
7378: 206D74  UEB1      JSR UEBER
737B: 9023              BCC NOUEBER
737D: 20BA74  FOUND     JSR AUSGABE
7380: A004              LDY #4
7382: 4C7CD2            JMP YFACC
7385: AD0676  UEBC      LDA P2ZW
7388: 8520              STA P2
738A: AD0776            LDA P2ZW+1
738D: F011              BEQ NOUEBER
738F: 8521              STA P2+1
7391: AD0876            LDA P5ZW
7394: 851E              STA P5
7396: AD0976            LDA P5ZW+1
7399: 851F              STA P5+1
739B: 208174            JSR UEBC1
739E: B0DD              BCS FOUND
73A0: A003    NOUEBER   LDY #3
73A2: 4C7CD2            JMP YFACC
73A5: A002    NOTFOUND  LDY #2
73A7: 4C7CD2            JMP YFACC
73AA: A980    SUCH      LDA #$80
73AC: 8DF975            STA FLAG
73AF: AD0076            LDA UEBER1
73B2: 85FB              STA S1
73B4: 851E              STA P5
73B6: AD0176            LDA UEBER1+1
73B9: 85FC              STA S1+1
73BB: 851F              STA P5+1
73BD: 18                CLC
73BE: A55F              LDA 95
73C0: 655E              ADC 94
73C2: 8DFA75            STA ENDZEIGE
73C5: A900    L1        LDA #0
73C7: 201874            JSR SEARCH
73CA: 90A0              BCC ENDE
73CC: 203E74            JSR INCS1
73CF: A000              LDY #0
```

```
73D1: B1FB          LDA (S1),Y
73D3: D15F          CMP (S2),Y
73D5: D0EE          BNE L1
73D7: A900          LDA #0
73D9: 8DF975        STA FLAG
73DC: 202874 L15    JSR TRANSFER
73DF: 203174        JSR INCS1S2
73E2: A000          LDY #0
73E4: B1FB          LDA (S1),Y
73E6: D15F          CMP (S2),Y
73E8: F0F2          BEQ L15
73EA: 205974        JSR DECS1
73ED: A5FC          LDA S1+1
73EF: 0980    L3    ORA #$80
73F1: 201874        JSR SEARCH
73F4: 903A          BCC ENDE1
73F6: 205974        JSR DECS1
73F9: B1FB          LDA (S1),Y
73FB: C51E          CMP P5
73FD: F00B          BEQ WEITER
73FF: 203E74        JSR INCS1
7402: 203E74        JSR INCS1
7405: A51F    L4    LDA P5+1
7407: 4CEF73        JMP L3
740A: 203E74 WEITER JSR INCS1
740D: 203E74        JSR INCS1
7410: B1FB          LDA (S1),Y
7412: D15F          CMP (S2),Y
7414: F0C6          BEQ L15
7416: D0ED          BNE L4
7418: 8DFB75 SEARCH STA SUCHASC
741B: B1FB   SU1    LDA (S1),Y
741D: CDFB75        CMP SUCHASC
7420: F00E          BEQ ENDE1
7422: 203E74        JSR INCS1
7425: 4C1B74        JMP SU1
7428: A5FB   TRANSFER LDA S1
742A: 851E          STA P5
742C: A5FC          LDA S1+1
742E: 851F          STA P5+1
7430: 60      ENDE1 RTS
7431: E65F    INCS1S2 INC S2
7433: D002          BNE *+4
7435: E660          INC S2+1
7437: A55F          LDA S2
7439: CDFA75        CMP ENDZEIGE
743C: F018          BEQ READY
743E: 18      INCS1 CLC
743F: A5FB          LDA S1
7441: 6901          ADC #1
7443: 85FB          STA S1
7445: 9002          BCC *+4
7447: E6FC          INC S1+1
7449: CD0276        CMP UEBEREND
744C: D0E2          BNE ENDE1
744E: A5FC          LDA S1+1
7450: CD0376        CMP UEBEREND+1
7453: D0DB          BNE ENDE1
7455: 18            CLC
7456: 68      READY PLA
7457: 68            PLA
7458: 60            RTS
7459: A5FB    DECS1 LDA S1
745B: D002          BNE *+4
745D: C6FC          DEC S1+1
745F: C6FB          DEC S1
7461: 60            RTS
7462: AD0476 UEBINIT LDA TABANF
7465: 8520          STA P2
7467: AD0576        LDA TABANF+1
746A: 8521          STA P2+1
746C: 60            RTS
746D: A003    UEBER LDY #3
746F: B120    LOOP1 LDA (P2),Y
7471: F01B          BEQ TABEND
7473: 88            DEY
7474: C51F          CMP P5+1
7476: D006          BNE WEITERU
7478: B120          LDA (P2),Y
747A: C51E          CMP P5
747C: F015          BEQ ENDEE1
747E: 88    WEITERU DEY
747F: 10EE          BPL LOOP1
7481: 18    UEBC1   CLC
7482: A520          LDA P2
7484: 6904          ADC #4
7486: 8520          STA P2
7488: 90E3          BCC UEBER
748A: E621          INC P2+1
748C: B0DF          BCS UEBER
748E: 18    TABEND  CLC
748F: 8D0776        STA P2ZW+1
7492: 60            RTS
7493: 98    ENDEE1  TYA
7494: 6902          ADC #2
7496: 2903          AND #3
7498: A8            TAY
7499: B120          LDA (P2),Y
749B: 48            PHA
749C: 88            DEY
749D: A520          LDA P2
749F: 8D0676        STA P2ZW
74A2: A521          LDA P2+1
74A4: 8D0776        STA P2ZW+1
74A7: A51E          LDA P5
74A9: 8D0876        STA P5ZW
74AC: A51F          LDA P5+1
74AE: 8D0976        STA P5ZW+1
74B1: B120          LDA (P2),Y
74B3: 8520          STA P2
74B5: 68            PLA
74B6: 8521          STA P2+1
74B8: 38            SEC
74B9: 60            RTS
```

```
74BA: A939    SUCHWORT LDA #$39
74BC: 85FD             STA C0
74BE: A903             LDA #$03
74C0: 85FE             STA C0+1
74C2: A000             LDY #0
74C4: B120    LOOP     LDA (P2),Y
74C6: F02B             BEQ END
74C8: 91FD    LOOPP    STA (C0),Y
74CA: 20E374           JSR DECREMEN
74CD: B120             LDA (P2),Y
74CF: F022             BEQ END
74D1: 10F5             BPL LOOPP
74D3: 297F             AND #$7F
74D5: 48               PHA
74D6: 20EB74           JSR DEC1
74D9: B120             LDA (P2),Y
74DB: 8520             STA P2
74DD: 68               PLA
74DE: 8521             STA P2+1
74E0: 4CC474           JMP LOOP
74E3: A5FD    DECREMEN LDA C0
74E5: D002             BNE *+4
74E7: C6FE             DEC C0+1
74E9: C6FD             DEC C0
74EB: A520    DEC1     LDA P2
74ED: D002             BNE *+4
74EF: C621             DEC P2+1
74F1: C620             DEC P2
74F3: 60      END      RTS

74F4: 203E75 SCHREIB   JSR SC1
74F7: A5FB             LDA S1
74F9: 8D0A76           STA ADRESSE
74FC: A5FC             LDA S1+1
74FE: 8D0B76           STA ADRESSE+1
7501: A200             LDX #0
7503: 205773           JSR SUB
7506: 203E75           JSR SC1
7509: A5FB             LDA S1
750B: 8D0C76           STA ADRESSE+2
750E: A5FC             LDA S1+1
7510: 8D0D76           STA ADRESSE+3
7513: A003             LDY #3
7515: AD0476           LDA TABANF
7518: 851E             STA P5
751A: AD0576           LDA TABANF+1
751D: 851F             STA P5+1
751F: B11E    LO1      LDA (P5),Y
7521: F00D             BEQ E1
7523: A51E             LDA P5
7525: 18               CLC
7526: 6904             ADC #4
7528: 851E             STA P5
752A: 90F3             BCC LO1
752C: E61F             INC P5+1
752E: B0EF             BCS LO1
7530: A007    E1       LDY #7
7532: B90A76  E2       LDA ADRESSE,Y
7535: 911E             STA (P5),Y
7537: 88               DEY
7538: 10F8             BPL E2
753A: 4CA573           JMP NOTFOUND
753D: 60      RTS      RTS
753E: 20AA73  SC1      JSR SUCH
7541: 9003             BCC SC2
7543: 4C2874           JMP TRANSFER
7546: 207C75  SC2      JSR TRANSEND
7549: A000             LDY #0
754B: ADF975           LDA FLAG
754E: F004             BEQ SS4
7550: A900             LDA #0
7552: F019             BEQ SS3
7554: A51F    SS4      LDA P5+1
7556: CD0376           CMP UEBEREND+1
7559: D007             BNE SS1
755B: A51E             LDA P5
755D: CD0276           CMP UEBEREND
7560: F010             BEQ SS2  POINT
7562: A51E    SS1      LDA P5   KANN
7564: 91FB             STA (S1),Y
7566: 209475           JSR INCS1S  SPA
7569: A51F             LDA P5+1  WERD
756B: 0980             ORA #$80
756D: 91FB    SS3      STA (S1),Y
756F: 209475           JSR INCS1S
7572: B15F    SS2      LDA (S2),Y
7574: 91FB             STA (S1),Y
7576: 208775           JSR INCS2S
7579: 4C7275           JMP SS2
757C: AD0276  TRANSEND LDA UEBEREND
757F: 85FB             STA S1
7581: AD0376           LDA UEBEREND+1
7584: 85FC             STA S1+1
7586: 60               RTS
7587: E65F    INCS2S   INC S2
7589: D002             BNE *+4
758B: E660             INC S2+1
758D: A55F             LDA S2
758F: CDFA75           CMP ENDZEIGE
7592: F018             BEQ READYY
7594: E6FB    INCS1S   INC S1
7596: D002             BNE *+4
7598: E6FC             INC S1+1
759A: A5FB             LDA S1
759C: CD1276           CMP PEND
759F: D00A             BNE E
75A1: A5FC             LDA S1+1
75A3: CD1376           CMP PEND+1
75A6: D003             BNE *+5
75A8: 4C03CE           JMP ERROR
75AB: 60      E        RTS
75AC: 68      READYY   PLA
75AD: 68               PLA
75AE: 209475           JSR INCS1S
75B1: A5FB             LDA S1
75B3: 8D0276           STA UEBEREND
```

```
75B6: A5FC            LDA S1+1
75B8: 8D0376          STA UEBEREND+1
75BB: 4C5974          JMP DECS1
75BE: A020    ALLINIT LDY #TABLEN
75C0: A9F8            LDA #STATUS
75C2: 8511            STA $11
75C4: A975            LDA #STATUS/256
75C6: 8512            STA $12
75C8: A900            LDA #0
75CA: 9111    LOOP11  STA($11),Y
75CC: 88              DEY
75CD: 10FB            BPL LOOP11
75CF: A900            LDA #TAB1
75D1: 8D0076          STA UEBER1
75D4: A910            LDA #TAB1/256
75D6: 8D0176          STA UEBER1+1
75D9: A901            LDA #TAB2
75DB: 8D0276          STA UEBEREND
75DE: A910            LDA #TAB2/256
75E0: 8D0376          STA UEBEREND+1
75E3: A914            LDA #TAB3
75E5: 8D0476          STA TABANF
75E8: A976            LDA #TAB3/256
75EA: 8D0576          STA TABANF+1
75ED: A9FF            LDA #$6FFF
75EF: 8D1276          STA PEND
75F2: A96F            LDA #$6FFF/256
75F4: 8D1376          STA PEND+1
75F7: 60              RTS

75F8: 00      STATUS  DFB $00
75F9: 00      FLAG    DFB $00
75FA: 00      ENDZEIGE DFB $00
75FB: 00      SUCHASC DFB $00
75FC: 000000
```

```
75FF: 00      WORD1   DFW $0000,$0000
7600: 0010    UEBER1  DFW TAB1 $0000
7602: 0110    UEBEREND DFW TAB2 $0000
7604: 1476    TABANF  DFW TAB3 $0000
7606: 0000    P2ZW    DFW $0000
7608: 0000    P5ZW    DFW $0000
760A: 000000
760D: 00      ADRESSE DFW $0000,$0000
760E: 000000
7611: 00              DFW $0000,$0000
7612: FF6F    PEND    DFW $6FFF
7614: 000000
7617: 00      TAB3    DFW $0000,$0000

              STRTEST EQU $CC90
              VARSUCH EQU $CF6D
              FACC    EQU $D6D2
              YFACC   EQU $D27C
              ERROR   EQU $CE03
              CHGOT   EQU $0076
              CHGET   EQU $0070
              POINT   EPZ $77
              P5      EPZ $1E
              S1      EPZ $FB
              S2      EPZ 95
              P1      EPZ S1
              P2      EPZ $20
              CO      EPZ $FD

              AUSGABE EQU SUCHWORT
              TAB1    EQU $1000
              TAB2    EQU TAB1+1
              TABLEN  EQU TAB3+4-STATU
              PROGEND EQU *
```

START	$7300	ENDE1	$7430	SC1	$753E	ADRESSE	$760A
LOESCH	$7319	INCS1S2	$7431	SC2	$7546	PEND	$7612
INIT	$731E	INCS1	$743E	SS4	$7554	TAB3	$7614
V1	$7351	READY	$7456	SS1	$7562	STRTEST	$CC90
V2	$7354	DECS1	$7459	SS3	$756D	VARSUCH	$CF6D
SUB	$7357	UEBINIT	$7462	SS2	$7572	FACC	$D6D2
INIT1	$7363	UEBER	$746D	TRANSEND	$757C	YFACC	$D27C
ENDE	$736C	LOOP1	$746F	INCS2S	$7587	ERROR	$CE03
UEBERSE	$736D	WEITERU	$747E	INCS1S	$7594	CHGOT	$76
UEB1	$7378 ¤	UEBC1	$7481	E	$75AB	CHGET	$70
FOUND	$737D	TABEND	$748E	READYY	$75AC	POINT	$77
UEBC	$7385	ENDEE1	$7493	ALLINIT	$75BE ¤	P5	$1E
NOUEBER	$73A0	SUCHWORT	$74BA	LOOP11	$75CA	S1	$FB
NOTFOUND	$73A5	LOOP	$74C4	STATUS	$75F8	S2	$5F
SUCH	$73AA	LOOPP	$74C8	FLAG	$75F9	P1	$FB
L1	$73C5	DECREMEN	$74E3	ENDZEIGE	$75FA	P2	$20
L15	$73DC	DEC1	$74EB	SUCHASC	$75FB	CO	$FD
L3	$73EF	END	$74F3	WORD1	$75FC	AUSGABE	$74BA
L4	$7405	SCHREIB	$74F4	UEBER1	$7600	TAB1	$1000
WEITER	$740A	LO1	$751F	UEBEREND	$7602	TAB2	$1001
SEARCH	$7418	S1	$7530	TABANF	$7604	TABLEN	$20
SU1	$741B	S2	$7532	P2ZW	$7606	PROGEND	$761E
TRANSFER	$7428	RTS	$753D ¤	P5ZW	$7608	¤= UNUSED	

Dirk Sanders, 6100 Darmstadt

SEARCH (AIM)

Suchprogramm mit Hex- und ASCII-Eingabe von bis zu 20 Zeichen

In manchen Monitorprogrammen, z.B. UETBUG2 für 6809, ist eine nützliche FIND-BYTE-Routine vorgesehen. Meist nur auf 1 Byte beschränkt, bietet sie trotzdem eine große Hilfe bei der Untersuchung von komplizierten Programmen und deren Zugriff auf bestimmte Speicherstellen, z.B. BASIC-Interpreter des AIM und seine Zero-Page. SEARCH bietet diese Funktion etwas erweitert für den AIM.

Bedienung des Programmes: Nach Aufruf mit der Taste F1 erscheint die Abfrage "SRCH =" und erwartet die Eingabe des zu untersuchenden Speicherbereiches wie beim Dump. Nachdem auf "TO=" die Endadresse eingegeben und mit Return abgeschlossen wurde, können bis zu 20 Hexzahlen und/oder ASCII-Zeichen eingegeben werden. Um eine Eingabe als ASCII-Zeichen zu kennzeichnen, ist pro Byte als erstes SPACE und dann das Zeichen einzugeben. Return schließt die Eingabe ab.

Nun erscheinen auf dem Display/Drucker die Adressen, in denen Übereinstimmung vorliegt (siehe Beispiele). Das Programm kehrt über RTS (weil Aufruf mit F1) in den Monitor zurück. Der Editor und sein Inhalt werden von SEARCH trotz gleicher Speicherstellen nicht beeinflußt. Ein Hinweis: Im Bereich $A000 *bis* $AFFF *kann es vorkommen, daß der Programmablauf durch Einwirkung der Interfacebausteine zum Stocken kommt. Soweit möglich ist dieser Bereich aus der Suche auszuklammern. Die Laufzeit (ohne Drucken) beträgt ca. 4 Sekunden für den Bereich* $0000 *bis* $FFFF .

```
0000 4C JMP 0C00    BELEGUNG DER FUNKTIONSTASTE F1

0C00 20 JSR EB44    CLEAR D/P POINTER
0C03 A0 LDY #00
0C05 B9 LDA E049,Y
0C08 20 JSR E97A    PRINT "SRCH"
0C0B C8 INY
0C0C C0 CPY #05
0C0E 90 BCC 0C05
0C10 20 JSR E7A3    FROM
0C13 B0 BCS 0C10    FEHLER?
0C15 20 JSR F8D0    ADDR IN NOWLN
0C18 20 JSR E83E    BLANK
0C1B 20 JSR E7A7    TO
0C1E B0 BCS 0C1B    FEHLER
0C20 20 JSR EA13    CR/LF
0C23 A2 LDX #01
0C25 20 JSR E973    INPUT ASCII
0C28 C9 CMP #0D     RETURN? = ENDE DER EINGABE
0C2A F0 BEQ 0C48
0C2C C9 CMP #20     SPACE?, DANN FOLGT ASCII
0C2E D0 BNE 0C35    FÜR HEX-EINGABE ÜBERSPRINGEN
0C30 20 JSR E973    INPUT ASCII
0C33 B0 BCS 0C3E    JMP
0C35 20 JSR EA84    1. HEX IN MSD VON A
0C38 20 JSR E973    INPUT ASCII
0C3B 20 JSR EA84    2. HEX IN LSD VON A
0C3E 95 STA EA,X    INSTRING-SPEICHER DES EDITORS
0C40 20 JSR E83E    BLANK
0C43 86 STX E9      ANZAHL DER BYTE
```

```
ØC45 E8 INX
ØC46 DØ BNE ØC25     JMP NÄCHSTES BYTE
ØC48 2Ø JSR EA13     CR/LF
ØC4B BØ BCS ØC5Ø     JMP, ERSTES MAL NOWLN NICHT ERHÖHEN
ØC4D 2Ø JSR F928     NOWLN=NOWLN+1
ØC5Ø AØ LDY #ØØ
ØC52 B1 LDA (DF),Y   LADE BYTE
ØC54 D9 CMP ØØEB,Y   VERGLEICHE MIT STRING
ØC57 FØ BEQ ØC68     BEI GLEICH:AUSDRUCK DER ADRESSE
ØC59 A6 LDX DF
ØC5B EC CPX A41C     OBERES ENDE ERREICHT?
ØC5E DØ BNE ØC4D
ØC6Ø A6 LDX EØ
ØC62 EC CPX A41D
ØC65 9Ø BCC ØC4D
ØC67 6Ø RTS          JA, ZURÜCK ZUM MONITOR
ØC68 C8 INY          NÄCHSTES BYTE PRÜFEN
ØC69 C4 CPY E9       IST DANN NOCH EINS ZU ÜBERPRÜFEN?
OC6B 9Ø BCC ØC52
ØC6D A5 LDA DF
ØC6F C9 CMP #EB      PROGRAMM FINDET NATÜRLICH AUCH DIE
ØC71 DØ BNE ØC77     ADRESSE DES SUCHSTRINGS,
ØC73 A5 LDA EØ       DIESE ° ÜBERSPRINGEN
ØC75 FØ BEQ ØC59
ØC77 AE LDX A415     ANZEIGEZEIGER
ØC7A EØ CPX #14      SIND SCHON 4 ADRESSEN IM DISPLAY?
ØC7C 9Ø BCC ØC81
ØC7E 2Ø JSR EA13     JA, FOLGT CR/LF
ØC81 2Ø JSR E83E     BLANK
ØC84 A5 LDA EØ
ØC86 2Ø JSR EA46     ADRESSE INS DISPLAY
ØC89 A5 LDA DF
ØC8B 2Ø JSR EA46
ØC8E 1Ø BPL ØC59     JMP

<[>SRCH FROM=EØØØ TO=FFFF      BEISPIEL:SUCHEN ASCII-STRING
 R  O  C  K  W  E  L  L        JEWEILS SPACE VOR DEN BUCHSTABEN
 FF8A                          ANZEIGE
<[>SRCH FROM=EØØØ TO=FFFF
 L  D  A                       SUCHEN DES ASCII-STRINGS LDA
 EFDØ                          ANZEIGE
<[>SRCH FROM=EØØØ TO=FFFF
2Ø Ø5 EF                       SUCHEN EINES SUBROUTINE-AUFRUFES
                               KEINE REFERENZ
<[>SRCH FROM=EØØØ TO=FFFF
4C Ø5 EF                       SUCHEN EINES SPRUNGBEFEHLES
 E7D1 EEF8                     ANZEIGE DER REFERENZEN
<[>SRCH FROM=EØØØ TO=FFFF
2Ø BC F8                       SUCHEN EINES SUBROUTINE-AUFRUFES
 F668 F69B F6D2                ANZEIGE DER REFERENZEN
```

#

Kleinanzeigen

Wegen Systemumstellung zahlreiche **AIM65-Erweiterungen** wie 80-stelliger Normalpapierdrucker, 9"-Monitor, ASCII-Tastatur, Kunststoff- und Metallgehäuse, CRT-Controller, 32k Speicher, BASIC und anderes zu verkaufen. Liste gegen Rückumschlag von P. Rott, Wernhüterstraße 7, 8900 Augsburg 1, oder Telefon 0821 - 71 29 23 nach 18 Uhr.

Floppy-Interface für DAIM zu DM 600,- zu verkaufen. Telefon (nach 19.30 Uhr) 0931 -

Bernhard Kokula, 6700 Ludwigshafen

Assembler Object Deplacer (AIM)

Programm zur Ablage des vom Assembler erzeugten Objectcodes an beliebiger Stelle ins RAM, unabhängig vom Speicherzielbereich, für den das Programm geschrieben ist. Das Programm ist damit eine utility für das EPROM-Programmieren und erspart spätere Verschiebungen wie z.B. mit RELOCAT oder LMR (Heft 17). Die Benutzung der Routine SADDR erspart Zeropage und testet automatisch, ob am Ziel der Ablage RAM ist (MEMERR). Das Programm startet unter F3-Taste den Assembler direkt. Es ist leicht verschiebbar (JSR PACKO).

Folgende Erklärungen zum Programm: Der AIM-Assembler liefert bei 'OBJ='Y, 'OBJ.OUT='U (User) den Objectcode über den anwendervektorierten Ausgabekanal UOUT. Die Bytes liegen bei dieser Ausgabe auf dem Stack und müssen von dort hervorgeholt werden. Die Ausgabe erfolgt in ASCII in einem Format, das demjenigen für Lochstreifen entspricht:

;	Anzahl Daten	1. Adresse	Daten	Checksumme	0D	0A
3A	2 Byte ASCII	4 Byte ASCII	n Byte ASCII	4 Byte ASCII	1 Byte CR	1 Byte LF

Der Beginn ist also immer das ';' (3A). Danach müssen 6 Zeichen in 3 Hexzeichen gepackt werden und kommen in einen eigenen Speicher. Die erste der drei Hexzahlen enthält dabei die Anzahl der zu erwartenden 'Nutz'-Daten (ASCII). Die darauf folgenden Datenkommen in den mit 'TO=' erfragten Speicherbereich, und zwar in der zuvor gemeldeten Menge. Was darauf folgt, wird weggeworfen'. Dann warten wir wieder auf ';', und es beginnt von vorn.

```
0000                                    ;V5 * 16.11.80/KOKULA
0000
0000          ASSEMBLER OBJEKT-DEPLACER
0000          ;PROGRAMM ZUR ABLAGE DES OBJEKTCODES
0000          ;AN BELIEBIGER STELLE IM RAM
0000          ;DIREKT BEIM ASSEMBLIEREN OHNE UMRECHNUNG
0000          ;BEDIENUNG:
0000          ;  START MIT F3-USER
0000          ;  ASSEMBLER-FRAGEN NORMAL BEANTWORTEN
0000          ;        "   -"OBJ."=Y(ES)
0000          ;        "-"OBJ.OUT"=U(SER)
0000
0000          AIM 65 EQUATES
0000
0000 ADDR             =$A41C          ;"TO"-ZIEL ADRESSE
0000 ASSEMB           =$D000          ;ASSEMBLER START
0000 MEMERR           =$EB33          ;MEMORY FAIL
0000 PACK             =$EA84          ;ASCII TO HEX
0000 PHXY             =$EB9E
0000 PLXY             =$EBAC
0000 SADDR            =$EB78          ;STA (ADDR),Y U.CMP(ADDR),Y
0000 TO               =$E7A7          ;STORE ZIELADRESSE
0000 UOUT             =$10A           ;USER-OUT VEKTOR
0000
0000          ZEROPAGE
0000                  *=$EB           ;EDITOR STRINGBUFFER
00EB FLAG             *=*+1           ;"1" NACH 1.PACK
00EC HALFPK           *=*+1           ;TEMP.STORE FUER 1.DATE
00ED ADCNT            *=*+1           ;ZAEHLER FUER 1. 3 ZEICHEN
00EE ADBUFF           *=*+2           ;SPEICHER FUER 1. 3 ZEICHEN
00F0 ADBUF            *=*+1           ;OBJ.DATEN COUNTER
00F1
```

```
00F1            ;-------------------
00F1            ;PROGRAMM ADRESSEN
00F1                  *=$112        ;USER-F3
0112            4C0002 JMP INIT
0115                  *=$200        ;*** PROGRAMMSTART
0200            ;-------------------
0200            ;INIT PER START F3
0200 INIT
0200            A90D   LDA #<INI    ;OBJ.DEPLACER
0202            8D0A01 STA UOUT
0205            A902   LDA #>INI
0207            8D0B01 STA UOUT+1
020A            4C00D0 JMP ASSEMB   ;ASSEMBLER DIREKTSTART
020D            ;-------------------
020D            ;INIT PER ASSEMBLER
020D INI                            ;"OBJ.OUT"=U(SER)
020D            B004   BCS OBJPRT
020F            20A7E7 JSR TO       ;HOLE RAM-ANF.ADRESSE
0212            60     RTS
0213            ;-------------------
0213            ;HAUPTPROGRAMM
0213 OBJPRT
0213            68     PLA          ;DATE WIRD IM STACK GEBRACHT
0214            209EEB JSR PHXY     ;SAVE <X>,<Y>
0217            C93B   CMP #';'     ;ANFANG JEDER 'LINE'
0219            D00A   BNE BU1
021B            A200   LDX #0
021D            86EB   STX FLAG     ;CLEAR FLAG
021F            A203   LDX #3       ;ADRESSBUFFER LAENGE
0221            86ED   STX ADCNT    ;BUFFER FUER 'OBJ.HEX ZAHL'
0223            D00F   BNE HOME     ;ADRESSE NICHT AUSGEWERTET
0225 BU1                            ;1. 3 DATEN ZUM ADRESSBUFFER
0225            A6ED   LDX ADCNT
0227            F00F   BEQ BU2      ;NUN KOMMEN DIE OBJ.DATEN
0229            205702 JSR PACKO    ;WANDLE 2X ASCII TO HEX
022C            B006   BCS HOME     ;NACH 1.DATE
022E            C6ED   DEC ADCNT    ;WENN 2. DATE AUCH DA IST
0230            A6ED   LDX ADCNT    ;POINTER
0232            95EE   STA ADBUFF,X ;SCHAFF DATE IN BUFFER
0234 HOME                           ;RUECKKEHR ZUM ASSEMBLER
0234            20ACEB JSR PLXY     ;RESTORE <X>,<Y>
0237            60     RTS
0238 BU2                            ;SPEICHERE OBJ.DATEN/
0238            A6F0   LDX ADBUF    ;  /SOLANGE ADBUF >0
023A            F0F8   BEQ HOME     ;NUR NOCH CHECKSUMME, CR/LF
023C OBJBUF                         ;SPEICHERN DER OBJ.DATEN
023C            205702 JSR PACKO
023F            B0F3   BCS HOME
0241            A000   LDY #0
0243            2078EB JSR SADDR    ;'STA (ADDR),Y'
0246            F003   BEQ *+5
0248            4C33EB JMP MEMERR   ;MEMORY FAIL
024B            C6F0   DEC ADBUF    ;OBJ.DATEN COUNTER/
024D            EE1CA4 INC ADDR     ;  / VOM ASSEMBLER GELADEN
0250            D0E2   BNE HOME
0252            EE1DA4 INC ADDR+1
0255            D0DD   BNE HOME
0257
```

```
0257            ;--------------------
0257            SUBROUTINE
0257 PACKO                              ;PACKT 2 ASCII TO HEX
0257            A6EB    LDX FLAG
0259            D006    BNE PA1
025B            85EC    STA HALFPK      ;NUR SAVEN
025D            E6EB    INC FLAG        ;KENNUNG: 1.DATE
025F            38      SEC             ;KENNUNG: KEINE DATE
0260            60      RTS
0261 PA1
0261            46EB    LSR FLAG        ;CLEAR FLAG
0263            A8      TAY             ;SAVE 2. CHAR.
0264            A5EC    LDA HALFPK
0266            2084EA  JSR PACK        ;PACK 1 CHAR.
0269            98      TYA
026A            4C84EA  JMP PACK        ;PACK 2. CHAR.
026D END                .END
026D             ERRORS= 0000
```

#

CHANGE To End (AIM)

Dieses kleine Dienstprogramm dient der Editierung von Texten im Textspeicher des AIM 65. Im Editor-Modus haben wir dort den C-Befehl (=CHANGE), der sich auch hinsichtlich des zu verwendenden Suchstrings von Hand repetieren läßt. Es bleibt aber die Mühe, für jeden zu ändernden String das TO=, den neuen String einzutippen und die Änderungsfunktion auszulösen. Dieses Programm automatisiert den Vorgang. Mit den zeilenorientierten Rangierbefehlen des Editors stellt man auf die Zeile ein, ab der geändert werden soll. Dann verläßt man den Editormodus mit QUIT oder ESCAPE und löst die F1-Taste aus. Der angezeigte Cursor fragt nun nach dem Suchstring, der abgeändert werden soll. Nach Return wird nun die erste Zeile angezeigt, in der der Suchstring vorkommt. Wie unter C-Befehl betätigt man nochmals Return und erhält die Abfrage 'TO='. Man gibt den neuen String ein, löst mit Return aus. Von nun ab werden alle Folgestrings im Textspeicher entsprechend abgeändert, wenn sie den Suchstring enthalten. Das untenstehende Beispiel (Auschnitt) wurde zum Spaß einmal auf den Quelltext dieses Programmes angewendet.

```
0000            ;--------------------
0000            CHANGE TO END
0000
0000           ;THIS PROGRAM WILL CHANGE
0000            ;ALL CHARACTER STRINGS DOWNWARDS FROM NOWLN
0000            ;TO NEW STRING DEFINED BY 'TO='
0000            ;ON THE FIRST INSTANCE
0000            ;WORKS LIKE C-COMMAND, BUT CONINUOUSLY
0000
0000            ;--------SYMBOLE---------
0000 CR=$D
0000 REPLAC=$F93F
0000 FC2=$F82E
0000 CURPO2=$A415
0000 CFLG=$F8B2
0000 FCHAR=$F80C
0000 READ=$E93C
0000 FC2=$F82E
0000 FCHA1=$F80F
0000 STIY=$A427
```

```
0000 OLDLEN=$E9
0000 ADDA=$F92A
0000 CLR=$EB44
0000 KEP=$E7AF
0000 INO2=$F77A
0000 SUB=$F91D
0000 PLNE=$F727
0000
0000          ---------------------
0000          MAIN
0000                *=$10C
010C          4C0006 JMP CHNG
010F          ;INSTALL F1-KEY
010F          ---------------------
010F
010F                *=$600
0600 CHNG
0600          20B2F8 JSR CFLG
0603          200CF8 JSR FCHAR
0606 CHN1     203CE9 JSR READ
0609          C90D   CMP #CR
060B          F009   BEQ CHN2
060D          202EF8 JSR FC2
0610          200FF8 JSR FCHA1
0613          4C0606 JMP CHN1
0616 CHN2
0616          AD29A4 LDA STIY+2
0619          85E9   STA OLDLEN
061B          AD15A4 LDA CURPO2
061E          48     PHA
061F          202AF9 JSR ADDA
0622          2044EB JSR CLR
0625          A005   LDY #5
0627          20AFE7 JSR KEP
062A          A000   LDY #0
062C          207AF7 JSR INO2
062F CHN2A
062F          68     PLA
0630          AA     TAX
0631          F006   BEQ CHN4
0633 CHN3     201DF9 JSR SUB
0636          CA     DEX
0637          D0FA   BNE CHN3
0639 CHN4
0639          202EF8 JSR FC2
063C          AD29A4 LDA STIY+2
063F          85E9   STA OLDLEN
0641          AD15A4 LDA CURPO2
0644          48     PHA
0645          202AF9 JSR ADDA
0648          203FF9 JSR REPLAC
064B          4C2F06 JMP CHN2A
064E                 .END
064E           ERRORS= 0000
```

Abo-Aktion Gewinne

Bei der am 4. April unter Ausschluß des Rechtsweges abgehaltenen Verlosung zu der in Heft 17 ausgeschriebenen Abo-Aktion wurden folgende Hauptgewinner ermittelt:

1. Preis: 1 SHARP BASIC Taschencomputer PC 1210 an Herrn Wolfgang Seer, 4230 Wesel.
2. bis 5. Preis: Je 1 Buch von L.J. Scanlon '6502 Software Design' an:
Prof. Tammo Glißmann, 4358 Haltern 5,
Martin Lang, Köln 50,
Renate Schulz, 6050 Offenbach/M,
Diego Galli, CH-6600 Locarno.

Die Gewinne wurden auch an die weiteren Gewinner versandt. Der Herausgeber dankt allen Beteiligten.
Die den erfolgreich geworben habenden übrigen Lesern zugesagte Null-Ziehkraft IC-Sockel kann wegen Lieferschwierigkeiten erst in Kürze zugesandt werden.

Kleinanzeigen in dieser Zeitschrift werden aufmerksam studiert und dienen der Kontaktaufnahme mit einer engagierten Leserschaft und vermitteln Angebote für den An- und Verkauf von Hard- und Software.

Kleinanzeigen kosten DM 10,- für 2 Zeilen. Betrag bitte bei Auftragserteilung überweisen oder in gängigen Briefmarken beilegen!

Kleinanzeigen

EPROM-Schießer für CBM komplett mit Steckern, Software auf Cassette oder Disk Anleitung, o. Gehäuse, DM 300,--. Spezial UV-Löschlampe z. EPROM-Löschen, niedr. Wärmeentwicklung, k. weiteres Zubehör erforderlich DM 120,-- +MWSt. David Long, Kl. Pfahlstr. 19a, 3000 Hannover, Tel.: 0511 - 31 190 19.

Eprom-Programmiergerät für CBM Anschlußfertig im Gehäuse. Mit eigenem Netzteil und dazugehörigem Kabel mit Steckern. Software für 2516 u. 2532 oder äquivalent - in bel. Bereich, auch in EPROM, kompl.DM 500,--. Heinrich Krohn, Am Berge 3d, 33 Braunschweig, T.: 05307-5413.

```
<[>JSR                  F1-Taste und Suchstring
JSR CFLG                Anzeige erstes Vorkommen
TO=JUMP SUBROUTINE      Vorgabe neuer String
END=<T>                 Abarbeiten bis Textende
.PAG 'CHANGE TO END
=<F>
JUMP
JUMP SUBROUTINE CFLG
=<L>
/
OUT=                    Listung eines Textabschnittes
JUMP SUBROUTINE CFLG
JUMP SUBROUTINE FCHAR
CHN1 JUMP SUBROUTINE READ
CMP #CR
BEQ CHN2
JUMP SUBROUTINE FC2
JUMP SUBROUTINE FCHA1
JMP CHN1
```

#

Tape-Dupe for AIM

Häufig tritt der Wunsch auf, ein Programm- oder Textfile von einer Tonbandcassette auf eine andere zu überspielen. Oft legt man aus Gründen der Datensicherung Zwischenversionen seiner Entwicklungen auf einer Arbeitscassette an, von der man später auf ein 'sauberes' Band überspielen will. Das nachstehende Programm kopiert automatisch von Tonbandkanal 1 nach Kanal 2. Zu Beginn wird der Bediener nach dem Namen des Files gefragt. Danach erfolgt das Aufsuchen des Files und das Kopieren automatisch. Voraussetzung ist allerdings, daß beide Laufwerke mit Fernsteuerung für das Ein- und Auschalten versehen sind.

```
0000          ;-----------------
0000          ;TAPE-DUPE FOR AIM
0000          ;-----------------
0000          ;BY ROLAND LOEHR
0000          ;
0000          ;V26.09.80-1
0000          ;
0000          ;-----------------
0000          ;THIS PGM COPIES
0000          ;A FILE, WHICH IS CALLED BY NAME
0000          ;FROM TAPE1 TO TAPE 2
0000          ;& WILL WORK ON BOTH TYPES OF FILES
0000          ;
0000          ;-----------------
0000          ;DECLARATION OF SYMBOLS
0000 CR=$D                          ;CARRIAGE RETURN
0000 WHEREI=$E848
0000 FNAM=$E8A2
0000 INFLG=$A412
0000                                ;INPUT DEVICE
0000 TAPOUT=$A435
0000                                ;FLAG FOR TAPE DEVICE-#
0000 LOADTA=$E32F
0000                GET FIRST BLOCK BY NAME
0000 TIBY1=$ED53
0000                                ;LOAD A PHYSICAL BLOCK
```

```
0000 TABUFF=$116
0000                                ;TAPE-BUFFER
0000 TAOSET=$F21D
0000                                ;SET TAPE FOR OUTPUT
0000 OUTTAP=$F24A
0000                                ;WRITE A BYTE TO TAPE
0000 DRB=$A800
0000                                ;PORT TO INTERFACE
0000 ACR=$A80B
0000                                ;VIA-REGISTER TO INFLUENCE
0000         ;TIMER 1 FREE RUNNING ON PB7
0000 PHXY=$EB9E
0000                                ;SAVE X,Y
0000PLXY=$EBAC
0000                                ;RESTORE X,Y
0000
0000
0000        ____________________
0000        KEYFUNCTION
0000              *=$10C          ;FOR KF1
010C         4C0002 JMP DUPE
010F        ____________________
010F        MAIN PGM TO BE CALLED SUBROUTINE
010F              *=$200
0200 DUPE
0200         2048E8 JSR WHEREI
0203         202102 JSRDU MP        ;COPY BLOCK
0206         ____________________
0206         AND NOW CHECK FOR LAST BLOCK
0206 CHECK
0206         A24F   LDX #$4F       ;TO CHECK AT END
0208         ;OF TAPE BUFFER
0208        BD1601LDA TABUFF,X
020B NULL
020B        C900   CMP #0          ;A FILLER?
020D         D009   BNE NXTBLK
020F         CA     DEX             ;THE BYTE AHEAD
0210         BD1601 LDA TABUFF,X
0213         C90D   CMP #CR         ;THE DELIMITER?
0215         D0F4   BNE NULL
0217         60     RTS             ;RETURN TO CALLER
0218         ____________________
0218         LOAD NEXT BLOCK
0218 NXTBLK
0218         2053ED JSR TIBY1
021B         202102 JSR DUMP
021E         4C0602 JMP CHECK
0221         ____________________
0221         SUBROUTINE DIVISION
0221 DUMP
0221         ;RESEMBLES TOBYTE IN MONITOR
0221         A901   LDA #1          ;TAPE2 FOR OUTPUT
0223         8D35A4 STA TAPOUT
0226         201DF2 JSR TAOSET
0229         A923   LDA #'# ;CHAR FOR BEGINNING
022B         204AF2 JSR OUTTAP
022E TABY2
022E         BD1601LDA TABUFF,X
0231         204AF2 JSR OUTTAP
```

```
0234            E8      INX
0235            E053    CPX #83         ;END OF BLOCK?
0237            D0F5    BNE TABY2
0239            AD00A8 LDA DRB
023C            29CF    AND #$CF        ;TURN BOTH TAPES OFF
023E            8D00A8 STA DRB
0241            58      CLI
0242            A900    LDA #0
0244            8D0BA8 STA ACR          ;TURN TIMER 1 OFF
0247            20ACEB JSR PLXY
024A TAPE1
024A            A900    LDA #0
024C            8D35A4 STA TAPOUT       ;RESTORE TAPE 1
024F            60      RTS
0250                    .END
0250             ERRORS= 0000                                    #
```

Ein Editor-Assembler für 6809

Angesichts der Vielzahl von Befehlen des 6809 und angesichts ihrer etwa 1400 Kombinationsmöglichkeiten in den Adressierungsarten, und weil einzelne Befehle bis zu 5 Byte lang sein können, ist die Benutzung eines Assemblers und Editors für die maschinensprachliche Programmierung sehr zu empfehlen.

Ein entsprechendes Programmpaket wurde auf dem EUROCOM2 der Firma Eltec unter Mitbenutzung von Routinen des Betriebssystems entwickelt. Eine Anpassung an andere Systeme dürfte ohne große Schwierigkeiten möglich sein. Der Texteditor erlaubt die Editierung mit Delete in der offenen Zeile, das Einfügen einer numerierten Zeile an der richtigen Stelle, das Löschen einer oder mehrerer Zeilen und ein Renumber in 5er Inkrementen. Quelltext kann auf Digital- oder auch Audiocassetten abgespeichert und auch modular zurückgeladen werden.

Der Assembler richtet sich weitgehend nach den Regeln des 'MC 6809 Preliminary Programming Manual' von Motorola. Er ist so konzipiert, daß er Syntaxfehler schon weitgehend bei der Abarbeitung erkennt, anhält und vom Benutzer eine korrigierte Zeile erwartet, die er dann abarbeitet. Auch eine Assemblierung im Einzelschrittmodus (jeweils eine Zeile) ist möglich. Es werden binäre, hexadezimale und dezimale Zahlen verarbeitet, im Immediate Mode auch ASCII-Zeichen als Operanden. Label und Symbolbezeichnungen dürfen bis zu 6 Zeichen lang sein und beanspruchen einen Platz von 10 Byte in der Symboltafel.

Bei ausreichendem Interesse kann dieser Editor-Assembler in dieser Zeitschrift veröffentlicht werden. Interessenten wenden sich bitte an den Herausgeber, der zwischenzeitlich auch einen direkten Kontakt zum Autor dieser Programmentwicklung, Herrn Claus Birkner herstellen wird. Nachstehend ein von diesem Assembler erzeugtes Listing für ein Verschiebeprogramm im Speicher.

```
0005 0000                       START   =4096
0010 0000                       ENDE    =5121
0015 0000                       LOC     =$B000
0020 0000                               *=%10010:10100000
0025 12A0 9E 00                 BEGIN   LDX <START
0030 12A2                               ;ANFANGSADRESSE DES BLOCKS HOLEN,DIRECT-
0035 12A2 10 BE B0 00                   LDY LOC
0040 12A6                               ;ANFANGSADRESSE DES NEUEN BEREICHS HOLEN
0045 12A6 A6 80                 LOOP    LDA ,X+   ;X-INDIZIERT,X=X+1
0050 12A8 A7 A0                         STA ,Y+   ;Y-INDIZIERT,Y=Y+1
0055 12AA BC 14 01                      CMPX ENDE
0060 12AD                               ;MIT ENDADRESSE DES BLOCKS VERGLEICHEN
0065 12AD                               ;EXTENDED-MODE
0070 12AD 25 F7                         BLO LOOP
0075 12AF                               .END
```


#

Ein 6809-System

Beim Herausgeber wird seit Oktober 1980 ein von der Firma Dohmann in Güterlsoh geliefertes 6809-System betrieben, und zwar in folgender hardwaremäßiger Konstellation: CPU—Karte NICO 69, Mutterplatine, 2 RAM/VIA Combokarten, Video-Karte, ASCII-Tastatur, Bildschirm, 2 Philips Digitalcassettenlaufwerke, Matrixdrucker. Alle Karten sind Standard-Serienprodukte im Europaformat mit 64-poliger VG-Leiste und Signalbelegung gem. MCS-Bus. Der Betrieb dieses Systems wird gestützt durch das diskettenorientierte SICOS-Betriebssystem, den SIMON-Monitor und durch einen leistungsfähigen Editor/Assembler. Zu den Platinen im einzelnen:

CPU-Karte. *Sie trägt die 6809-CPU (wahlweise 1 oder 2 MHz), 2 KB RAM-Speicher, 8 KB ROM, PROM oder EPROM (Typ durch Lötbrücken festlegbar), 2 VIAs 6522 sowie Treiber für alle Busse. Bei der hier betriebenen Version enthalten die Festwertspeicher in 6 KB das Betriebssystem. Eine VIA ist als System-VIA eingesetzt und dient dem künftigen Anschluß eines Frontpanels. Die andere VIA dient dem Betrieb eines Matrixdruckers.*

Combokarten. *Diese Karten mit Bustreibern an der Eingangsseite können wahlweise mit 8KB statischen RAMs oder mit 2 VIAs 6522 oder mit beidem bestückt werden. Jede dieser Funktionsgruppen hat einen unabhängigen Adreß-select in Form von DIP-Schaltern. Die Kombokarten sind damit vielseitig verwendbar und können an die jeweiligen Bedürfnisse des Betreibers angepaßt werden. Beim hier betriebenen System dienen 2 VIAs dem Interfacing für Tastatur und Digitalcassettenrekordern, die beiden VIAs einer weiteren Combokarte dienen Anwenderzwecken.*

Die Videokarte *ist eingangsseitig ebenfalls busgepuffert. Sie trägt den 6845 CRT-Controller, die Takterzeugung, 2 KB Video-RAM, 2 KB RAM für die Erzeugung eigener Zeichensätze sowie 2 K EPROM für das Video-Betriebssystem einschl. Standard-Zeichensatz. Diese Karte wurde früher schon am AIM 65 betrieben und trägt daher noch das AIM-bezogene Betriebssystem, während dasjenige für 6809 in den EPROMs der CPU-Karte untergebracht ist.*

Bei Auswahl und Konfiguration des 6809-Entwicklungssystems sprach hier ganz deutlich der Wunsch mit, die Combo- und Videozusatzkarten ohne doppelten Investitionsaufwand sowohl am AIM 65 wie auch am 6809 betreiben zu können.

Das SICOS-Betriebssystem *stellt die höchste Befehlsebene beim Einschalten des Gerätes dar. Durch Schalter am Frontpanel ist allerdings auch eine anwenderspezifische Vektorierung des Betriebsbeginns möglich. Das SICOS meldet sich zunächst mit der interaktiven Datumsabfrage. Danach kann man durch die reservierten Worte SIMON bzw. GO (Adresse) in den Monitor eintreten oder ein residentes Anwenderprogramm zur Ausführung bringen. Daneben stehen unter SICOS verschiedene Befehle zur Vergügung, die dem Verkehr mit den Cassettenlaufwerken regeln, nämlich Laden oder Speichern eines Files (auch mit Vorgabe einer Ladeadresse), Listen der Inhaltsverzeichnisse, Löschen eines Files, Laden und Starten eines Programmes. Daneben sind Datumsanzeige und -änderung sowie über ein Statuswort das Setzen der aktiven E/A-Systemeinheiten möglich. Das SICOS ist bereits auf Diskettenbetrieb zugeschnitten.*

Daher ein kurzes Wort zu den Digital-Cassettenlaufwerken. Die Aufzeichnung läuft auf Mini-Cassetten, die vor der ersten Inbetriebnahme formatiert werden müssen. Dabei erhält das Magnetband einen Kopf mit Namen, Datum und Inhaltsverzeichnis, und es werden die nachfolgenden Blöcke bereits als Leerblöcke angelegt. Jede namentliche Ansprache eines Files bedingt daher zunächst das Aufschlagen des Inhaltsverzeichnisses am Bandbeginn, danach den schnellen Vorlauf zum Beginn des Files und anschließend das Lesen oder das Schreiben in die Posititon des nächstfreien Blocks. Die Blockverwaltung ist daher wie bei einer Diskette eindeutig gelöst. Da auch eine formatierte Cassette im Prinzip noch immer ein sequentieller Datenträger ist, erge-

ben sich durch die Spulvorgänge gewisse Wartezeiten. Auch werden je Cassettenseite effektiv nur einige 40.000 Bytes gespeichert. Der Vorteil dieses Trägers liegt natürlich im Preis und auch in der Hantierungs- und Datensicherheit z.B. gegenüber einem üblichen Audio-Cassettenrecorder. Und man darf die schnelleren Floppies ja wohl auch in einiger Zeit erwarten. - Nach den Beobachtungen des Autors sollten die Magnetköpfe der Cassettendrives nicht in der unmittelbaren Nähe eines Bildschirm-Monitors betrieben werden.

Der SIMON-Monitor *enthält (mit Tastenrepeat) u.a. folgende Dienstleistungen:Aufschlagen und Veränderung von Speicherzelleninhalten unter Adresse im Vorwärts- und Rückwärtslauf, formatierten Memory-Dump, Registeranzeige GO (Programmausführung ab Adresse), TRACE und STEP.*

Der Text-Editor. *Dieses Dienstprogramm wird zusammen mit dem Assembler ins RAM geladen (ca. 6 KB) und stellt das komfortable Herzstück des Softwaresystems dar. Die Leistungen des Editors sind überdurchschnittlich und eignen nicht nur zur bequemen Abfassung von Programmquelltexten für den Assembler, sondern auch für die Textbearbeitung allgemein, zumal eine bequeme Umschaltmöglichkeit für gemischte Groß- und Kleinschreibung unter Zuhilfenahme der Shift-Taste gegeben ist. Der Bediener wird interaktiv um die Definition eines Speicherbereiches für den Textspeicher und für ein scratch pad (Notizblock) gebeten. Außerdem steht ihm mit einfachem Tastendruck immer ein Statusverzeichnis des Text-Editors zur Verfügung.*

Textzeilen werden unter Angabe einer virtuellen Zeilennummer angezeigt, die von außen her verwaltet wird und die nicht in die Abspeicherung eingeht. Unter Angabe einer Zeilennummer kann Text an beliebiger Stelle um neue Zeilen ergänzt, bearbeitet, entfernt oder umrangiert werden, letzteres einzelzeilenweise oder auch zeilenblockweise.

Es stehen die üblichen zeilenorientierten Rangierbefehle für den Textspeicher zur Verfügung: Kopfzeile anzeigen (TOP), letzte Zeile (BOTTOM), nächste Zeile (NEXT), Vorzeile (UP). Die zeilenorientierten Befehle N und U können mit erweiterten Suchfunktionen (FIND) gekoppelt und repetiert (wiederholt) werden: Schlage die Folge- oder Vorzeile auf, die den Suchstring irgendwo als Instring enthält oder den Suchstring definiert am Zeilenanfang oder am Zeilenende. Vorstehende Funktionen auch limitiert auf eine vorgegebene Menge von Folge- oder Vorzeilen.

Ähnliche Dienste finden wir auch bei den Textbearbeitungsbefehlen Lösche (DELETE) Einzelzeile, eine Menge Folgezeilen, alle Folgezeilen bis vor das Auftreten eines Suchstrings (Stichwortes), der auch so definiert sein kann, daß er am Anfang oder am Ende einer Zeile auftreten muß. Der Stringveränderungsbefehl (CHANGE) kann auf die offene Zeile, eine Menge von Folgezeilen oder bis zu einer Zeile (Nummer) angewendet werden. Einzelzeilen oder Zeilenblöcke können auf den Notziblock mit oder ohne Löschung der Quellzeile(n) kopiert und von dort an beliebiger Stelle wieder eingefügt werden, natürlich mit der sich ergebenden laufenden Numerierung.

Der Inline-Editor. *Jede bereits abgeschlossene Zeile kann zur komfortablen Neubearbeitung wieder eröffnet werden. Der Schreibcursor kann definiert an den Zeilenanfang, an das Zeilenende oder auf ein bestimmtes Zeichen gesetzt werden. Unter Repeat-Funktion ist ferner ein Rangieren nach rechts oder links möglich. Zeichen und Zeichenketten können eingefügt oder gelöscht werden, auch ein erase to end of line ist möglich, ferner die Verschmelzung mit der Folgezeile.*

Auf der Befehlsebene des Editors stehen ebenfalls die Ein- und Ausgabebefehle des Betriebssystems SICOS zur Verfügung. - Die besondere Leistungsfähigkeit des Editors wird hier bereits in verschiedenen Textbearbeitungsaufgaben genutzt.

Der Assembler. *Die 6809-Assemblerprogrammierung stellt in dieser Zeitschrift einen besonderen Abschnitt dar. Zu dem von der Firma Dohmann gelieferten Assembler ist zu bemerken, daß er fast überall die Motorola-Syntax benutzt. Wo Abweichungen bestehen, dienen sie dem größeren Programmierkomfort, insbesondere für das Setzen und Löschen von Flags im Statusriegister. La-*

bels können eine beliebige Länge haben. Somit ist es möglich, im Label eine Satzaussage für die Tätigkeit unter dem Label (Symbol) zu machen. Der Assembler läßt ferner eine strukturierte Programmierung in Zähl- und Bedingungschleifen unter Benutzung lokaler Label zu, die die Symboltafel nicht belasten und deren Aufbau dem DO WHILE oder DO UNTIL entspricht. Mit der hier benutzten ersten Version des Assemblers konnte einwandfrei gearbeitet werden, inzwischen sind auch bedingte Assemblierung und Makros implementiert.

An anderer Stelle wurde bereits darauf hingewiesen, daß Instruktionen in der Maschinensprache des 6809 bis zu 5 Byte lang sein können. Angesichts dessen ist ein guter Assembler ein Muß, mit der byteweisen hexadezimalen Befehlseingabe in 'Hackermanier' ist nichts mehr auszurichten.

Dokumentation: *Der Autor hat der erste System in der genannten Hardware-Softwarekombination bezogen, als Dokumentation noch sehr knapp war. Die Systemkarten sind mit Layout und Signalen sicher ausreichend beschrieben. Die Softwaredokumentation umfaßte die Programme des SIMON und CRTKEY (Keyboard und Monitorausgabe) sowie die Einsprungsadressen des Lineeditors, ferner Bedienerhinweise für SICOS und den Editor/Assembler. Man darf hoffen, daß demnächst weitere Dokumentation und die Implementierungsfähigkeit für höhere Sprachen zur Verfügung stehen werden.*

Zusammenfassung. *Das besprochene System ist modular aus Serienprodukten aufgebaut worden, auch unter dem Gesichtspunkt, die Module wahlweise als Erweiterung am AIM 65 betreiben zu können. Zwischenzeitlich stehen weitere Systemkarten für den Ausbau zu Entwicklungs- und Anwendersystemen zur Verfügung, ebenso eine noch leistungsfähigere Assemblerversion. Gelegentlich soll hier entsprechend nachgerüstet werden. Auch ohne diese Abrundungen hat der 6809-Computer die Erwartung erfüllt, ein preiswertes, leistungsfähiges und in seinen Ausbaumöglichkeiten wohldurchdachtes Entwicklungssystem zu erhalten, das die Einarbeitung in die Programmierung ermöglicht und das mit seinem komfortablen Text-Editor auch Anwendungen im kaufmännischen Betrieb zuläßt. In Betrieben, wo ein solches Entwicklungssystem der Vorbereitung von Computern in festen Anwendungen dienen soll, kann dasselbe Modulsystem im Anwendungscomputer weiterbenutzt werden. Schon die CPU-Karte stellt mit ihrem RAM, den Steckplätzen für Festwertspeicher und den 2 Interfacebausteinen 6522 einen möglichen selbständigen Computer dar.*

R. L.

Hans-Georg Lange, 1000 Berlin 30

Direktzugriff mit CBM

Bekanntlich bietet das BASIC 4.0 des CBM, wie es z.B. im 8032 implementiert ist, in Verbindung mit dem DOS 2.0 der Floppy 4040 komfortable Möglichkeiten des Direktzugriffs auf einzelne Sätze einer relativen Datei.
Auch wenn man das BASIC 4.0 nicht hat, so kann man mit dem hier beschriebenen Verfahren den Direktzugriff trotzdem benutzen. Voraussetzung ist allerdings, das das DOS 2.0 in der Floppy implementiert ist (für die 3040 gibt es Aufrüstsätze).

Das erste der beiden abgedruckten Programme eröffnet eine relative Datei, erweitert sie auf 511 Sätze und füllt diese mit Inhalt. Das zweite Programm akzeptiert eine Satz- und Positionsnummer und liest den betreffenden Satz aus der Datei.

Wie man sieht, erfolgt die Verwaltung von relativen Dateien ausschließlich durch Übertragen von Zeichenketten über den Kommandokanal bzw. Sekundäradresse. Das Zugriffsverfahren ist damit nicht an BASIC gebunden. Allerdings ist das Zusammensetzen dieser Kommandostrings in Assembler oder TCL-PASCAL (siehe Heft 17) etwas aufwendiger.

```
 100 C$="ABCDEF":REM TESTSTRING
 110 OPEN1,8,15
 120 RL=10:REM RECORDLAENGE 10
 130 PO=1:REM POSITION 1 IM RECORD
 140 X$="1:TEST,L,"+CHR$(RL)+",W":REM DATEINAME: TEST, RECORDLAENGE RL
 150 OPEN2,8,2,X$:REM OEFFNE RELATIVE DATEI MIT SEK.ADR 2
 160 PRINT#1,"P"CHR$(2);CHR$(255);CHR$( 1 );CHR$(1):PRINT#2:REM
 170 REM ERWEITERE DATEI AUF 511 RECORDS. NACH DIESEM KOMMANDO ERFOLGT DIE
 180 REM FEHLERMELDUNG: RECORD NOT PRESENT,SIE DARF IGNORIERT WERDEN
 190 FORI=1TO511
 200 T$=C$+STR$(I)
 215 PRINT#1,"P"CHR$(2);CHR$(IAND255);CHR$(I/256);CHR$(PO):REM NAECHSTER RE
 220 REM POSITION 1
 230 REM BEI SEQUENTIELLEM SCHREIBEN KANN DIESES KOMMANDO ENTFALLEN
 240 PRINT#2,T$;:REM SCHREIBE RECORD
 250 PRINTI;
 260 NEXT
 270 CLOSE2:REM SCHLIESSE RELATIVE DATEI
READY.
```

PROGRAMM2

```
 100 OPEN1,8,15
 230 OPEN2,8,2,"1:TEST,L":REM OEFFNE RELATIVE DATEI ZUM LESEN
 400 INPUT"RECORD# ";I
 405 IFI>511THENPRINT"ILLEGALE RECORDNR.":GOTO400
 410 INPUT "POSITION ";PO
 420 IFPO>8THENPRINT"ILLEGALE.POSITION":GOTO410
 450 PRINT#1,"P"CHR$(2);CHR$(IAND255);CHR$(I/256);CHR$(PO):REM ANWAEHLEN
 455 INPUT#2,A$
 460 PRINTA$
 470 IFI=0 THEN CLOSE2 : END
 500 GOTO400
READY.
```

Dr. Olaf Haeggquist, 8700 Würzburg

DAIM Floppy Disk-System für den AIM 65

Der folgende Beitrag beschreibt das von der Firma Compass 1978 herausgebrachte System. Besprochen wird nur die softwaremäßig neueste Version (2.2), die auch File-Linkage in PL/65 erlaubt.

Geliefert wird entweder nur das Conrollerboard allein, oder auch mit max. zwei Shugart SA Laufwerken. Die Controllerkarte ist pinkompatibel zum 84-poligen AIM Motherboard, der Datenbus ist also invertiert. Will man den Kauf des Motherboards umgehen, dann muß ein 8-Bit invertierender bidirketionaler Bus-Buffer (z.B. SN 74LS640) dazwischengeschaltet werden. Der Controller ist für heutige Verhältnisse als veraltet anzusehen (NEC 372). Die Software befindet sich in vier 2708-EPROMs und hat einen Umfang von knapp 4K Byte. Maximal sind (softwarebedingt) zwei Minifloppy-Laufwerke anschließbar, singlesided und single density, also auch z.B. BASF 6106 oder Shugart SA 400. Die Speicherkapazität beträgt (formatiert) 78 K Byte. Im Objektcode lassen sich nur 30 K Byte abspeichern, bedingt durch das AIM 65/System 65-Aufzeichnungsformat. Dieser Nachteil wird nur zum Teil aufgewogen, daß SYSTEM 65-Kompatibilität besteht.

Der Befehlsvorrat des Disk Operating Systems ist ebenso auch mit dem des SYSTEM 65 identisch:

C = Compress Disk	R = Recover File
L = List File	1 = Disk Functions
X = Rename File	2 = Display Directory
I = Init Disk	3 = Delete File
Z = Zero Directory	4 = List Room Left.

Der Datenverkehr mit dem AIM erfolgt über den User IN/OUT und wird durch *= 9009 aktiviert. F1 springt zu den Disk-Functions, mit F2 können die Floppys gestoppt werden. Alle Ein- und Ausgabemöglichkeiten des AIM bleiben voll erhalten, jedoch muß am Ende immer zum Monitor zurückgekehrt werden und mit F2 die Disk gestoppt werden. Im Falle von Dump ist es zum Schließen des Files obligat. Dieser Nachteil liegt im AIM–Monitor begründet, jedoch läßt sich der Monitor für Eprommer-Besitzer leicht so verändern, daß das Abschalten der Floppys überflüssig wird und man immer im Editor oder im BASIC verbleiben kann. Das Abspeichern von 4K Objektcode oder von 8K Text braucht 8 Sekunden. Diese Zeit ist sehr lang und im wesentlichen durch das Aufzeichnungsformat bzw. durch den Controller selbst begründet. Hierin liegt sicher der größte Nachteil des Systems, das sonst kaum Wünsche offen läßt.

Im Assembler ist Disk File Linkage nur mit dem von Compass angebotenen Assembler ($90) möglich, der dann auch ein brauchbares Listing und die Symboltabelle herausgibt. Daneben ist sogar noch Assemblieren mit Offset möglich. Außerdem bietet Compass noch für $10 das Assemblerlisting des Floppy-Betriebssystems an und gibt auch Hinweise für direktes Lesen und Schreiben der Disk per Programm. Daneben wird neuerdings zu jedem Controllerboard eine Distribution Disk mit nützlichen Programmen geliefert.

Für weitere Auskünfte steht der Verfasser gern zur Verfügung (Tel. nach 19.30 Uhr: 0931-28 27 27). Ebenso gibt er gern Hinweise für spezielle Änderungen des AIM-Monitors bzw. hilft bei der Beschaffung von EPROMs.

Maschinensprache-Monitor für ACORN ATOM. Für den in Heft 16 dieser Zeitschrift angekündigten ACORN ATOM ist mittlerweile ein komfortabler Maschinensprache-Monitor alsEPROM oder auf Cassette erhältlich. Neben den Grundfunktionen eines Monitors, wie Anzeige und Änderung von Register- und Speicherinhalten, ermöglicht ein Simulator Einzelschrittbetrieb und kontinuierliche Abarbeitung eines Maschinenprogrammes. Ebenso lassen sich Speicherinhalte verschieben und vergleichen. Einfache Hexadezimal-Arithmetik ist möglich. Als Besonderheit sind die residenten Assembler- und Disassemblerroutinen zu erwähnen, deren Ein- und Ausgabeformate zueinander kompatibel sind. Großer Wert wurde auf übersichtliche Darstellung und einfache Bedienung gelegt. Der Monitor erweitert den ACORN ATOM damit zu einem vollwertigen und preiswerten Entwicklungssystem für Assemblerprogramme. Kontakt: M. Nowak, Gustav-Müller-Str. 15, 1000 Berlin 62. (H G. Lange)

****

Hannover-Messe 1981

Nachfolgend ein kurzer Messebericht: Nach dem persönlichen Eindruck war der Besucherandrang an den ersten Messetagen nicht so groß wie in früheren Jahren. Das Angebot im Bereich Bürotechnik und in den zur Mikroprozessortechnik gehörenden Sparten ist noch größer geworden. In der Tendenz kann man feststellen, daß der Prozessor in mehr und mehr Anwendungen hineinschlüpft, daß die Zahl der Anwendercomputer und der Platinencomputer bereits unübersehbar groß geworden ist und daß Bildschirmmonitore in den verschiedensten Anwendungen zur Selbstverständlichkeit geworden sind. Der farbige Bildschirm ist dabei noch relativ selten. In hervorragender Bildqualität war er z.B. bei Commodore (hochauflösender Farbcomputer als Proto-

typ), bei APPLE, bei HP und auf einigen anderen Ständen zu beobachten, auch bei einigen Japanern. - Die Diskussion um die an Bildschirmarbeitsplätzen entstehenden Belastungen ist bekannt. Nach Ansicht des Autors sollte doch einmal geprüft werden, ob mit der heute durchaus schon preiswerten Farbwiedergabe dort Erleichterungen geschaffen werden können, indem Mitarbeiter die Farben von Schrift und Hintergrund auf ihr Sehvermögen und die Beleuchtungsverhältnisse am Arbeitsplatz einstellen und zur Vermeidung von Monotonie auch verändern.

Commodore hatte neben den schon bekannten Geräten der Serien 4000 und 8000 nebst Peripherie (Drucker, Floppies) vor allem den Volkscomputer zu DM 898,- groß herausgestellt. Der VC 20 ist mit Schreibmaschinentastatur, 8 Tasten für Sonderfunktionen, vier programmierbaren Tasten und Farbumschalttasten sowie einem Ausgang für den Anschluß eines PAL- oder s/w-Fernsehgerätes versehen. In seiner Grundausstattung ist der VC 20 also ein bildschirmorientierter Computer mit Betriebssystem und BASIC in 20 KB Festwertspeicher und 5 KB RAM, davon 3,5 KB für den Benutzer. Gegenüber dem BASIC 3 ergeben sich gewisse Einschränkungen. Für die Farbausgabe stehen 8 Farben für die Zeichen und 16 für den Hintergrund zur Verfügung. 23 Zeilen werden mit je 22 Zeichen abgebildet, das sind 64 ASCII-Zeichen (Umschaltung groß/klein) und grafische Zeichen. Das erzeugte Farbbild machte auf den benutzten handelsüblichen FS-Empfängern mit ihrer beschränkten Bandbreite einen sehr guten Eindruck. Integriert ist ferner ein Tongenerator.

Für diesen ab etwa Juni 1981 lieferbaren Computer sind zahlreiche Erweiterungs- und Anschlußmöglichkeiten vorgesehen, nicht nur für RAM und ROM, Spiele, Speichern auf Datasette, sondern auch RS 232C serielles Interface, IEEE 488-Interface (Datenübertragung, bzw. Anschluß von CBM-Peripherie).

Commodore stellte auch Prototypen aus, den schon erwähnten hochauflösenden Farbcomputer, ein 5 MB Floppy-System, das auch die Herstellung von Sicherheitskopien erlaubt und eine 'MICRO MAINFRAME. Zu diesem interessanten Gerät sind im Anschluß an ein Gespräch mit R. John Feagans (Designer) folgende Details zu vermitteln: Unter der Haube findet man 2 CPUs, nämlich die bekannte 6502 des CBM 8032 und die 6809 von Motorola, die durch Umschaltung im Wechsel betrieben werden können. Dieser Computer wurde für die Waterloo-Universität entwickelt, um den dortigen Großrechner von den Programmentwicklungen der vielen Studierenden zu entlasten. Es wurden daher die Sprachen Waterloo-BASIC, FORTRAN, APL, PASCAL und Assembler implementiert, COBOL ist in Vorbereitung.

Zur Hardware gehört eine RS 232C-Schnittstelle, die den Terminalbetrieb gestattet, der CBM 8032 als Basis, 64 K virtuelles RAM und 20 K zusätzliches ROM. Jeder Prozessor kann die ersten 32 K des gemeinsamen RAM und den virtuellen Speicher adressieren und natürlich die ihm zugeordneten ROMs. - Warum wurde nun diese Kombination gewählt? Es sollte Aufwärtskompatibiltität für vorhandene 8032-Programme mit 6502 CPU gewährleistet werden (also nicht nur eine Speichererweiterung wie beim 8096). Die CPU 6809 leiht sich besser für die Implementierung der höheren Sprachen und kann in dieser Umgebung die meiste vorhandene Hardware mitbenutzen. Alle bisherigen kompatiblen Drucker und Floppies können weiterbenutzt werden. Der spezielle Zeichensatz für das APL wird durch ein keyboard overlay und eine Erweiterung des Zeichengenerators erzeugt. - Das Waterloo-BASIC kennt keine Beschränkung der Länge von Strings, es läßt längere Variablennamen zu, hat Strukturierungsmerkmale und kann Prozeduren mit stellvertretenden Parametern erklären. - Die MICRO MAINFRAME ist für den Herbst 1981 angekündigt.

Auf dem Stand von **APPLE COMPUTER** war kaum ein Fuß auf die Erde zu setzen. Besonders wurde der APPLE III mit zahlreichen Softwareanwendungen für den professionellen Einsatz herausgestellt.

CANON Computer präsentierte seinen CX-1, einen Tischcomputer auf der Basis des 6809 mit Bildschirmmonitor (31 cm, 24 Zeilen mit 80 Zeichen/Z), Keyboard mit besonderem Zahlenfeld und 2 Floppies 5,25" mit je 320 KB Speicherkapazität, alles in ein Gehäuse integriert und 25 kg schwer. Dem Anwender stehen 32 KB RAM zur Verfügung (erweiterbar auf 96 KB), ferner Be-

triebssystem, BASIC und die Verwaltung von ISAM-Dateien (Indexed Sequential Access Method), Assembler und später auch PASCAL und COBOL. Weitere Optionen umfassen diverse Schnittstellen und Peripherie. Preis oberhalb DM 12.000 (o. Gewähr).

Die GWK-Elektronik aus Herzogenrath zeigte ihre Computersysteme und das vielfältige modulare Kartenangebot für 6502 und 6809 im Europaformat und mit VG 64-Leiste. Neu ausgestellt wurden die Karten für 6809, CPU-Karte mit 10 KB RAM oder ROM (pinkompatible Typen), alle Busse gepuffert und DMA-fähig, ausgelegt für memory banking und multiuser/multitasking-Betrieb, mit seriellem I/O über UART, Baudrate einstellbar, ferner quasistatische RAM-Karte mit 32 KB, Serielle I/O-Karte, parallele I/O-Karte für Centronix-Interface, Controllerkarte für 1-4 Floppy Disk Laufwerke mit single oder double density oder sides (5,25"), EPROM-Board und Buskarte mit Belegung 64/96.

#

Rockwell's Zeitschrift INTERACTIVE für den AIM 65 soll zentral von Anaheim in Kalifornien aus vertrieben werden. Entgegen früheren Ankündigungen ist daher ein Bezug über den Herausgeber der Zeitschrift 65xx MICRO MAG nicht möglich. Die hier vorliegenden Abonnementsbestellungen wurden Anfang Januar 1981 an Rockwell in München-Martinsried zur Erledigung weitergeleitet. Wie es scheint, haben die Lieferungen auf Grund dieser Bestellungen jedoch noch nicht begonnen.

KIM-1 wird nicht mehr geliefert. Dem Vernehmen nach ist die Produktion dieses einst preiswerten und beliebten Computers ausgelaufen. Wegen des vielen für dieses Gerät leicht ereichbaren Schrifttums ist das sicher schade. Andererseits ist darauf hinzuweisen, daß am Markt inzwischen eine Fülle von Einplatinencomputern in Europakartenformat für alle Einsatzzwecke erhältlich ist.

#

Nützliche Dokumentation für PET und CBM

ROM-Listen für die Commodore-Rechner wurden schon verschiedentlich veröffentlicht, so auch erstmals in Heft 11 dieser Zeitschrift. Von der Firma Hans-Joachim Koch in Garbsen sind jetzt zwei sehr nützliche weitergehende Dokumentationen zu beziehen: Zunächst eine 14-seitige DIN A4-Liste mit über 400 Adreßanganben für die Computer der Serien 2000 (BASIC 1.0), 3000 (BASIC 3.0) und 8000 (BASIC 4.0) in paralleler Darstellung. Auf etwa 4,5 Seiten sind dabei die Belegungen der RAM-Speicherseiten 0 bis 3 dargestellt (in deutscher Sprache), auf den übrigen Seiten folgen die Einsprungspunkte der ROM-Routinen mit Kommentar bis zu 4 Zeilen je Eintrag. Als Beispiel ein kurzer Ausschnitt aus dieser ROM-Liste:

```
2001  3001  8001  Beschreibung / Entry-point
-------------------------------------------------
D702  D718  C963  WAIT-Kommando
D71E  D72C  C97F  führt Addition und Subtraktion aus
D728  D736  C989  FAC #2 - FAC #1
D73F  D77C  C9A8  FAC #1 + FAC #2
D7AC  D7E3  CA8D  normalisiert FAC #1
D891  D8C8  CAF2  reelle Zahl: 1.8
```

Eine zweite 36-seitige Liste enthält ausführliche Beschreibungen der wichtigsten ROM-Routinen. Für den Assembler-Programmierer dürften sie mit ihren bisher einmaligen Darstellungen in 10 Kapiteln unerläßlich sein: 1. Beschreibung der internen Darstellung für Adressen, Zahlen, Strings. 2. Eingaben von der Tastatur (Byte, String, Hex-Byte, Hex-Adresse). 3. Ausgabe auf Bildschirm für dito. 4. Umwandlungsroutinen ASCII zu den versch. Zahlendarstellungen und umgekehrt. 5. Arithmetik-Routinen mit Laden und Abspeichern. 6. Logische Operationen. 7. Verbindung zu BASIC (RUN, Suche nach Variablen, nach BASIC-Zeilen, Auswertung von Ausdrücken, Parameterübergabe an Assemblerprogramme). 8. Ein- und Ausgabe über den IEEE-488-Bus. 9. Cassetten E/A. 10. Verschiedenes.

Die Listen sind bei Hans-Joachim Koch, Liegnitzer Straße 8 in 3008 Garbsen 8 zu beziehen, 05131-53 510, und zwar zum Preise von 20 bzw. 24 DM, beide Listen zusammen 35 DM (o.G.). Abschließend ein kleiner Auszug aus der Beschreibung der ROM-Routinen:

8.4 Eingabe vom IEEE-488-Bus

Soll die gesamte Dateiverwaltung aus dem Assemblerprogramm erfolgen, so ist nach dem öffnen der Datei die Eingabe folgendermaßen vorzunehmen:

```
LDA #0      ;STATUS-FLAG
STA STATUS  ; LOESCHEN
JSR TALK    ;AKTIVIERT DEVICE ALS TALKER
LDA SA      ;SEKUNDAERADRESSE
JSR SECND   ; ZUM BUS SENDEN
JSR ACPTR   ;ZEICHEN VOM BUS HOLEN
STA CHAR    ; UND ABSPEICHERN
JSR UNTLK   ;DEVICE INAKTIV MACHEN
LDA STATUS  ;STATUS-FLAG
BNE ERROR   ; PRUEFEN
```

Es werden die Angaben LF, DN und SA benötigt.

Peter W. Arps, 2000 Hamburg 73

SUPER LIST CBM/Centronics

SUPER-LIST ist ein Programm zum Listen von BASIC-Programmen mit einem Centronix-Drukker, der ja die Bildschirmsteuerzeichen des CBM nicht korrekt ausgibt, um eine einwandfreie Programmliste zu erhalten. Die Cursoranweisungen, Pi und Reverse ON/OFF werden daher umgesetzt. Weiter wird nach 64 Druckzeilen ein Seitenvorschub ausgeführt und der Titel des auszudruckenden Programms ausgegeben.

Um die Übersichtlichkeit des Ausdruckes zu erhöhen, wird zwischen den einzelnen Befehlen eine Leerstelle eingefügt. Ist eine Programmzeile länger als 66 Zeichen, so wird zwischen dem 66. und 70. Zeichen nach einer Leerstelle ein Zeilenvorschub gemacht. Dadurch wird erreicht, daß Befehle möglichst nicht getrennt werden.

Die BASIC-Befehle 'OPEN 4,4', 'CMD4' und 'CLOSE4' werden in Maschinensprache ausgeführt. Es ist nicht erforderlich, wie nach OPEN 4,4:CMD4:LIST einen Syntax-Error zu erzeugen, um die Ausgabe wieder auf den Bildschirm zu legen. Mit STOP kann der Ausdruck jederzeit unterbrochen werden.

```
                    0010              ; S U P E R L I S T
                    0020              ;
                    0030              ;
                    0040              .BA $7E00     ;SET ADR-COUNTER
                    0050              ;
                    0060 FLAG         .DE 9
                    0070 FELD1        .DE 251
                    0080 FELD2        .DE 252
                    0090 ZEILEN       .DE 253
                    0100 ZEICHEN      .DE 254
                    0110 YSAVE        .DE 70
                    0120 FILE         .DE 210
                    0130 DEV          .DE 212
                    0140 ADR          .DE 92
                    0150 I/O          .DE 14
                    0160 BASIC.STRT   .DE 40
                    0170 AREA         .DE $33A
                    0180 READ         .DE $FFCF
                    0190 WRITE        .DE $FFD2
                    0200 PRTEXT       .DE $CA1C
                    0210 OPEN         .DE $F524
                    0220 CLOSE        .DE $F2AC
                    0230 CMD          .DE $FFC9
                    0240 RESET        .DE $CAB7
                    0250 BASIC        .DE $C389
                    0260 CRLF         .DE $C9E2
                    0270 STOP         .DE $F301
                    0280 NR           .DE $DCD9
                    0290 KLRTXT       .DE 49298
                    0300 ;
                    0310 ;
7E00- A9 D6         0320 ENTRY        LDA #L,TEXT
7E02- A0 7F         0330              LDY #H,TEXT
7E04- 20 1C CA      0340              JSR PRTEXT    ;'TITLE?'
7E07- A2 02         0350              LDX #2
7E09- 20 CF FF      0360 INPUT        JSR READ      ;INPUT UEBERSCHRIFT
7E0C- C9 0D         0370              CMP #13       ;ENDE?
7E0E- F0 06         0380              BEQ TXTEND    ;JA
7E10- 9D 3A 03      0390              STA AREA,X
```

```
7E13- E8          0400              INX
7E14- D0 F3       0410              BNE INPUT
                  0420              ;
7E16- A9 1B       0430 TXTEND       LDA #27        ;ESC
7E18- A0 0E       0440              LDY #14        ;BREITSCHRIFT
7E1A- 8D 3A 03    0450              STA AREA
7E1D- 8C 3B 03    0460              STY AREA+1
7E20- A9 00       0470              LDA #0
7E22- 9D 3A 03    0480              STA AREA,X     ;TEXTENDE-KENNUNG
7E25- 20 4B 7E    0490              JSR SETP
                  0500              ;WENN OPEN4,X SELBST ERFOLGEN SOLL
                  0510              ;'JSR OPEN' ENTFERNEN
7E28- 20 24 F5    0520              JSR OPEN
7E2B- A2 04       0530              LDX #4
7E2D- 20 C9 FF    0540              JSR CMD
7E30- 8E 0E 00    0550              STX I/O
7E33- 20 80 7E    0560              JSR UEB        ;UEBERSCHRIFT AUGEBEN
7E36- AE 28 00    0570              LDX BASIC.STRT
7E39- AD 29 00    0580              LDA BASIC.STRT+1
7E3C- 20 B4 7E    0590              JSR WORK
7E3F- 20 4B 7E    0600              JSR SETP
7E42- 20 AC F2    0610              JSR CLOSE

7E45- 20 B7 CA    0620              JSR RESET
7E48- 4C 89 C3    0630              JMP BASIC
                  0640              ;
                  0650              ;
7E4B- A9 FF       0660 SETP         LDA #255       ;POINTER FUER OPEN/CLOSE
7E4D- A2 04       0670              LDX #4
7E4F- 85 D3       0680              STA *211
7E51- 86 D2       0690              STX *FILE
7E53- 86 D4       0700              STX *DEV
7E55- 60          0710              RTS
                  0720              ;
                  0730              ;
7E56- 20 D2 FF    0740 PUT          JSR WRITE
7E59- E6 FE       0750              INC *ZEICHEN ZEICHENZAEHLER ERHOEHEN
7E5B- 10 48       0760              BPL RTN1
7E5D- 24 09       0770              BIT *FLAG      ;'"'-FLAG AN?
7E5F- 30 04       0780              BMI P1         ;JA
7E61- C9 20       0790              CMP #' '       ;LEERSTELLE?
7E63- F0 06       0800              BEQ P2         ;JA - NEUE ZEILE
7E65- A5 FE       0810 P1           LDA *ZEICHEN
7E67- C9 85       0820              CMP #133 BEI 70 ZEICHEN IMMER NEUE ZEIL
7E69- D0 3D       0830              BNE RTN1
7E6B- 20 71 7E    0840 P2           JSR OUT        ;NEUE ZEILE
7E6E- 4C 9A 7E    0850              JMP TAB5
                  0860              ;
7E71- 20 E2 C9    0870 OUT          JSR CRLF AUSGEBEN
7E74- C6 FD       0880              DEC *ZEILEN -ZAEHLER VERMINDERN
7E76- D0 22       0890              BNE TAB5       ;KEIN BLATTWECHSEL
7E78- A2 08       0900              LDX #8
7E7A- 20 E2 C9    0910 O1           JSR CRLF       ;8 x VORSCHUB
7E7D- CA          0920              DEX
7E7E- D0 FA       0930              BNE O1
                  0940              ;
7E80- B4 FC       0950 UEB          STY *FELD2
7E82- A2 0A       0960              LDX #10
```

```
7E84- 20 9C 7E  0970            JSR T10         ;TAB(10)
7E87- A9 3A     0980            LDA #L,AREA     ;UEBERSCHRIFT AUGEBEN
7E89- A0 03     0990            LDY #H,AREA
7E8B- 20 1C CA  1000            JSR PRTEXT
7E8E- 20 E2 C9  1010            JSR CRLF
7E91- 20 E2 C9  1020            JSR CRLF
7E94- A9 3E     1030            LDA #62
7E96- 85 FD     1040            STA *ZEILEN     ;GRUNDSTELLUNG
7E98- A4 FC     1050            LDY *FELD2
                1060            ;
7E9A- A2 05     1070 TAB5       LDX #5
7E9C- A9 20     1080 T10        LDA #' '
7E9E- 20 D2 FF  1090 T1         JSR WRITE
7EA1- CA        1100            DEX
7EA2- D0 FA     1110            BNE T1
7EA4- A9 3F     1120            LDA #63         ;MAX.133-63=70 ZEILEN
7EA6- 85 FE     1130            STA *ZEICHEN
7EA8- 60        1140 RTN1       RTS
                1150            ;
                1160            ;
7EA9- 20 71 7E  1170 ZEND       JSR OUT         ;ENDE PGM-ZEILE
7EAC- A0 00     1180            LDY #0
7EAE- B1 5C     1190            LDA (ADR),Y
7EB0- AA        1200            TAX
7EB1- C8        1210            INY
7EB2- B1 5C     1220            LDA (ADR),Y     ;ADR NAECHSTE ZEILE
                1250 WORK       ;AUSGABE DER BASIC-ZEILEN
7EB4- 86 5C     1260            STX *ADR        ;POINTER INS PM
7EB6- 85 5D     1270            STA *ADR+1
7EB8- A0 01     1280            LDY #1
7EBA- 84 09     1290            STY *FLAG       ;GRUNDSTELLUNG
7EBC- B1 5C     1300            LDA (ADR),Y
7EBE- F0 E8     1310            BEQ RTN1        ;PGM-ENDE
7EC0- 20 01 F3  1320            JSR STOP GEDRUECKT?
7EC3- F0 E3     1330            BEQ RTN1 - JA - ENDE
7EC5- C8        1340            INY
7EC6- B1 5C     1350            LDA (ADR),Y     ;ZEILEN # LOW
7EC8- AA        1360            TAX
7EC9- C8        1370            INY
7ECA- B1 5C     1380            LDA (ADR),Y     ;ZEILEN # HIGH
7ECC- 20 D9 DC  1390            JSR NR          ;UMWANDELN U. AUSGEBEN
7ECF- A0 03     1400            LDY #3
                1410            ;
7ED1- A9 20     1420 LOOP3      LDA #' '
7ED3- 20 56 7E  1430 LOOP2      JSR PUT         ;PRINT BYTE
7ED6- C8        1440 LOOP1      INY
7ED7- B1 5C     1450            LDA (ADR),Y     ;LADE PGM-BYTE
7ED9- F0 CE     1460            BEQ ZEND        ;ZEILEN-ENDE
7EDB- 30 0E     1470            BMI PI BEI ;PI,BEFEHLEN UND GRAPH.ZEICHEN
7EDD- C9 22     1480            CMP #'"'
7EDF- D0 41     1490            BNE DPUNKT
7EE1- A5 09     1500            LDA *FLAG
7EE3- 49 80     1510            EOR #128        ;FLAG SETZEN/LOESCHEN
7EE5- 85 09     1520            STA *FLAG
7EE7- A9 22     1530            LDA #'"'
7EE9- D0 E8     1540            BNE LOOP2       ;IMMER
                1550            ;
7EEB- C9 FF     1560 PI         CMP #255        ;PI?
```

```
7EED- F0 04     1570            BEQ UMSETZ    ;JA
7EEF- 24 09     1580            BIT *FLAG     ;GRAPH.ZEICHEN?
7EF1- 10 48     1590            BPL BEFEHL    ;NEIN - IST BEFEHL
                1600            ;
7EF3- 84 46     1610 UMSETZ     STY *YSAVE
7EF5- A2 FF     1620            LDX #255
7EF7- E8        1630 UMS1       INX
7EF8- DD 90 7F  1640            CMP TAB1,X    ;ZEICHEN IN TAB?
7EFB- 90 D6     1650            BCC LOOP2     ;NEIN - UNVERAEN.AUSGEBEN
7EFD- D0 F8     1660            BNE UMS1      ;WEITERSUCHEN
7EFF- 8A        1670            TXA
7F00- 0A        1680            ASL A         ;MAL 4
7F01- 0A        1690            ASL A
7F02- 85 FB     1700            STA *FELD1    ;SAVE OFFSET IN TAB2
7F04- A0 04     1710            LDY #4
7F06- A6 FB     1720 UMS2       LDX *FELD1    ;INDEX LADEN
7F08- BD 99 7F  1730            LDA TAB2,X    ;LADE ZEICHEN
7F0B- F0 08     1740            BEQ UMS3      ;LETZTES ZEICHEN
7F0D- 20 56 7E  1750            JSR PUT       ;AUSGEBEN
7F10- E6 FB     1760            INC *FELD1    ;INDEX + 1
7F12- 88        1770            DEY
7F13- D0 F1     1780            BNE UMS2
7F15- C6 FE     1790 UMS3       DEC *ZEICHEN KORREKTUR
7F17- A9 1B     1800            LDA #27       ;ESC
7F19- 20 D2 FF  1810            JSR WRITE
7F1C- A9 0F     1820            LDA #15       ;BREITSCHRIFT AUS
7F1E- A4 46     1830            LDY *YSAVE

7F20- D0 B1     1840            BNE LOOP2     ;IMMER
                1850            ;
                1860            ;
7F22- 24 09     1870 DPUNKT     BIT *FLAG
7F24- 30 CD     1880            BMI UMSETZ    ;IST IN STRING
7F26- C9 3A     1890            CMP #':
7F28- D0 05     1900            BNE AEND
7F2A- 20 56 7E  1910            JSR PUT       ;AUSGEBEN
7F2D- D0 A2     1920            BNE LOOP3     ;NACH ':' SPACE
                1930            ;
7F2F- C9 41     1940 AEND       CMP #'A
7F31- 90 A0     1950            BCC LOOP2
7F33- C9 5B     1960            CMP #$5B
7F35- B0 9C     1970            BCS LOOP2
7F37- 09 80     1980            ORA #128 GROSSBUCHSTABEN AUSGEBEN
7F39- D0 98     1990            BNE LOOP2
                2000            ;
                2010            ;
7F3B- 38        2020 BEFEHL     SEC
7F3C- E9 7F     2030            SBC #127      ;OP-CODE ./. 127
7F3E- 85 FB     2040            STA *FELD1    ;SAVE
7F40- 84 46     2050            STY *YSAVE
7F42- 24 09     2060            BIT *FLAG     ;LETZTER BEFEHL = ON?
7F44- 50 07     2070            BVC BEF1      ;NEIN
7F46- 20 8B 7F  2080            JSR B1        ;SPACE EINFUEGEN
7F49- A5 FB     2090            LDA *FELD1
7F4B- 85 09     2100            STA *FLAG     ;FLAG AUS
7F4D- A0 12     2110 BEF1       LDY #18       ;OFFSET TAB4
7F4F- 20 82 7F  2120            JSR BINSRT    ;GGF. SPACE EINFUEGEN
7F52- A6 FB     2130            LDX *FELD1
7F54- E0 12     2140            CPX #18       ;IST ES ON?
```

```
7F56- D0 04      2150             BNE BEF2      ;NEIN
7F58- A9 40      2160             LDA #64
7F5A- 85 09      2170             STA *FLAG     ;'ON'-FLAG
7F5C- A0 FF      2180 BEF2        LDY #255      ;BASIC-BEFEHL ...
7F5E- CA         2190 TXT1        DEX  ;DECODIEREN U. AUSGEBEN
7F5F- F0 08      2200             BEQ TXT3
7F61- C8         2210 TXT2        INY
7F62- B9 92 C0   2220             LDA KLRTXT,Y
7F65- 10 FA      2230             BPL TXT2
7F67- 30 F5      2240             BMI TXT1
7F69- C8         2250 TXT3        INY
7F6A- B9 92 C0   2260             LDA KLRTXT,Y LADE KLARTEXT
7F6D- 08         2270             PHP  ;SAVE STATUS
7F6E- 09 80      2280             ORA #128 GROSSBUCHSTABEN AUSGEBEN
7F70- 20 56 7E   2290             JSR PUT
7F73- 28         2300             PLP  ;RESTORE STATUS
7F74- 10 F3      2310             BPL TXT3      ;WEITER
7F76- A5 FB      2320             LDA *FELD1
7F78- A0 FF      2330             LDY #255
7F7A- 20 82 7F   2340             JSR BINSRT
7F7D- A4 46      2350             LDY *YSAVE
7F7F- 4C D6 7E   2360             JMP LOOP1     ;NAECHSTES ZEICHEN
                 2370             ;
                 2380             ;
7F82- C8         2390 BINSRT      INY
7F83- D9 BD 7F   2400             CMP TAB4,Y    ;SPACE EINFUEGEN?
7F86- F0 03      2410             BEQ B1        ;JA
7F88- B0 F8      2420             BCS BINSRT    ;WEITERSUCHEN
7F8A- 60         2430             RTS  ;OHNE SPACE
```

```
7F8B- A9 20     2450 B1         LDA #'
7F8D- 4C 56 7E  2460            JMP PUT
                2470            ;
                2480            ;
7F90- 11 12 13  2490 TAB1       .BY $11 $12 $13 $1D $91 $92
7F93- 1D 91 92
7F96- 93 9D FF  2500            .BY $93 $9D $FF
                2510 ;
                2520 ;TABELLE ZUR UMSETZUNG DER GRAPH.
                2530 ;ZEICHEN AUS TAB1
                2540 ;JE ELEMENT 4 BYTES
                2550 ;27 14: FUER CENTRONICS 730 BREITSCHRIFT
7F99- 1B 0E 40  2560 TAB2       .BY 27 14 64 0
7F9C- 00
7F9D- 72 76 73  2570            .BY 'rvs' 0
7FA0- 00
7FA1- 68 6F 6D  2580            .BY 'home'
7FA4- 65
7FA5- 1B 0E 5E  2590            .BY 27 14 94 0
7FA8- 00
7FA9- 1B 0E DE  2600            .BY 27 14 222 0
7FAC- 00
7FAD- 6F 66 66  2610            .BY 'off' 0
7FB0- 00
7FB1- 63 6C 72  2620            .BY 'clr' 0
7FB4- 00
7FB5- 1B 0E DF  2630            .BY 27 14 223 0
7FB8- 00
7FB9- 70 69 00  2640            .BY 'pi' 0 0
7FBC- 00
                2650 ;
                2660 ;TABELLE DER TOKEN-128 FUER LEERSTELLE EINFUEGEN
7FBD- 02 03 06  2670 TAB4       .BY 2 3 6 7 8 10 11 12 14 18
7FC0- 07 08 0A
7FC3- 0B 0C 0E
7FC6- 12
7FC7- 13 17 18  2680            .BY 19 23 24 26 30 31 32 33
7FCA- 1A 1E 1F
7FCD- 20 21
7FCF- 22 25 28  2690            .BY 34 37 40 41 42 48 49
7FD2- 29 2A 30
7FD5- 31
7FD6- 0D 0D 54  2700 TEXT       .BY 13 13 'TITLE? ' 0
7FD9- 49 54 4C
7FDC- 45 3F 20
7FDF- 00
                2710            .EN
```

LABEL FILE: [/ = EXTERNAL]

```
/FLAG=0009          /FELD1=00FB         /FELD2=00FC
/ZEILEN=00FD        /ZEICHEN=00FE       /YSAVE=0046
/FILE=00D2          /DEV=00D4           /ADR=005C
/I/O=000E           /BASIC.STRT=0028    /AREA=033A
/READ=FFCF          /WRITE=FFD2         /PRTEXT=CA1C
/OPEN=F524          /CLOSE=F2AC         /CMD=FFC9
/RESET=CAB7         /BASIC=C389         /CRLF=C9E2
/STOP=F301          /NR=DCD9            /KLRTXT=C092
ENTRY=7E00          INPUT=7E09          TXTEND=7E16
```

```
SETP=7E4B       PUT=7E56        P1=7E65
P2=7E6B         OUT=7E71        O1=7E7A
UEB=7E80        TAB5=7E9A       T10=7E9C
T1=7E9E         RTN1=7EA8       ZEND=7EA9
WORK=7EB4       LOOP3=7ED1      LOOP2=7ED3
LOOP1=7ED6      PI=7EEB         UMSETZ=7EF3
UMS1=7EF7       UMS2=7F06       UMS3=7F15
DPUNKT=7F22     AEND=7F2F       BEFEHL=7F3B
BEF1=7F4D       BEF2=7F5C       TXT1=7F5E
TXT2=7F61       TXT3=7F69       BINSRT=7F82
B1=7F8B         TAB1=7F90       TAB2=7F99
TAB4=7F8D       TEXT=7FD6
//0000,7FE0,7FE0
```

#

Editorial

Mit diesem Heft 18 ist der dritte Jahrgang des 65xx MICRO MAG nun komplett. Rein äußerlich bedeuten 18 Hefte an die 800 veröffentlichte reine Textseiten. Wichtiger als diese Zählung ist der vermittelte Inhalt, die vielen Programme zur Systementwicklung besonders des KIM-1, des AIM 65 und des PET/CBM, die vielen zu den Systemen gegebenen Erklärungen und die grundsätzlichen Übersichten zu besonderen Programmierungs- und Betriebsfragen. Viele Artikel wurden wegen ihres Gehaltes mit Genehmigung an anderer Stelle nachgedruckt oder wurden Vorbild für andere (manchmal ohne Nennung der Quelle!).

In diesen drei Jahren ergaben sich viele hundert beratende Gespräche mit Systembetreibern am Telefon und sicher auch einige hundert persönliche Gespräche im Zusammenhang mit Messen, Vorträgen und den durchgeführten Workshops. Es war und ist eine schöne Aufgabe, den Lesern die für die Nutzung der Systeme notwendigen Kenntnisse auf diesen Wegen zu vermitteln. Heute ist mit Freude festzustellen, daß es die anfängliche Unsicherheit nicht mehr gibt. Der Computer und seine Programmierung sind selbstverständlich geworden, weil entsprechende Information zur Verfügung steht.

Die Innovation schreitet weiter. Viele der großen Hersteller bringen in kurzer zeitlicher Folge neue Systeme auf den Markt. Hinzu tritt ein zunehmendes Angebot von Systemen aus den mittelständischen Betrieben. Und auch neue CPUs und Erweiterungsmöglichkeiten wollen behandelt sein. Der darzustellende Stoff nimmt damit zu. Trotz der Innovation sollte man aber nicht übersehen, daß für die schon etwas älteren beliebten Systeme wie AIM 65 und CBM 3032 noch keineswegs alle geschickten Programmier- und Nutzungsmöglichkeiten dargestellt sind. Und sie nehmen ja auch weiter an der Implementierung und Verbesserung der höheren Sprachen teil. In vielen Fällen wird es auch das Rationellste sein, auf einen bisher schon benutzten und bekannten Computer weiter aufzubauen.

Trotz allen Bemühens ist es von hier aus schier unmöglich, auf alle interessanten neuen Computer einzugehen. Man kann ein solches Gerät nicht nur für Besprechungszwecke anschaffen. Aus diesem Grunde sind Artikel und Programme aus der Leserschaft weiterhin sehr willkommen, zumal sie immer die Thematik bereichern. Zur Vermeidung unnötiger Arbeit stimme man sich im Vorwege kurz telefonisch ab. Jedem Programm sollte eine Cassette oder Diskette beigefügt sein, damit mit den hier vorhandenen Geräten ein reprofähiger Ausdruck des Listings angefertigt werden kann.

#

English Summary

The first article in this issue opens up a series on **Motorola's MC6809.** First of all its registers are detailed, especially the status. Next the signals of the MCU are grouped to bus control, timing and interrupt. - **Laufzeitmessung für Programme** allows to measure the execution time of a program to the accuracy of 1 cycle. Although written for the AIM 65 it may be rewritten for other systems. - **ON ERROR GO TO** displays the erroneous statement to the CBM screen. - **Pseudo 16 Bit CPU** simulates a CPU of this kind on any 6502-system and contains a multitude of most commonly used basic instructions. - **Garbage Collection** is a treatment of how this job is done in BASIC 3.0. and 4.0. - **Assoziative Tabellen** introduces a threaded method of string storage and retrieval for pairs of items, may be used as a translater. - **SEARCH** allows to scan the whole AIM memory for strings of ASCII and/or hex bytes and to list all references. - **Assembler Object Deplacer** puts object output from the AIM assembler to any desired location to prepare EPROM burning from RAM. - **Tape Dupe** for AIM copies one desired file from tape 1 to tape 2. - **CHANGE To End** repeats the AIM editor Change command from NOWLN to the bottom of text. - **Ein 6809-System** is a users report on a German system with a very good editor/assembler - **Direktzugriff mit CBM** installs relative files in CBM BASIC 3.0 together with DOS 2. - **DAIM Floppy** is a users report. - **Super List** adapts a Centronics printer to the CBM and formats the output of BASIC listings.

#

1/4-Grafik für CBM 3001

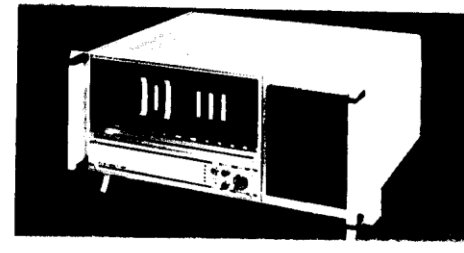
In Heft 17 wurde für das Programm mit obigem Namen (Seite 15) leider die Tabelle nicht mit abgedruckt. Sie lautet in hexadezimaler Notierung für die Speicherplätze ab 03EB:

7E 7C E2 7B 61 FF EC 6C 7F E1 FB 62 FC FE A0

#

Nach dem Auslaufen des KIM-1 dürfte der in mancher Weise ähnliche aber leistungsfähigere **ESCO-Computer** auf vermehrtes Interesse stoßen. Es handelt sich um ein modulares Computersystem auf Europakarten mit 96-poligem VG-Stecker aus deutscher Fertigung. Die CPU-Karte kann dabei schon allein als Einplatinencomputer betrieben werden. Sie trägt die 6502 CPU, vier 24-polige Sockel für Festwertspeicher und bis zu 2 KB RAM, ferner 2 Interfacebausteine PIA 6520 (6522 in Vorbereitung) und Bustreiber. Zusammen mit der Karte ESCO-2, die eine Hexatastatur, LED-Anzeigen, Cassetten- und TTY-Interface sowie einen Interfacebaustein 6520 für deren Bedienung trägt, und mit dem für die CPU Karte lieferbaren Monitorprogramm und zahlreichen Zusatzkarten wird der ESCO zum leistungsfähigen Entwicklungssystem, besonders natürlich mit seinem Editor mit den üblichen zeilen- und stringorientierten Befehlen und dem 2-Pass Assembler. Die Karte ESCO 2 ist aber auch T-förmig auf die CPU-Karte aufsteckbar und bildet dann das Bedienpanel für den Einschubcomputer.

Das 'große' ebenfalls von der Firma Neumüller vertriebene Entwicklungssystem ist der **BEM IMPACT 1000 von Brutech** auf Europakarten im 19-Zoll–Gehäuse und wahlweise für 6502 oder 6809 ausgelegt. Dazu gehören 48 KB RAM und 14 KB ROM-Software (Debug-Monitor, DOS) und 2 Floppies mit Macro-Assembler/Editor, die üblichen Schnittstellen und Erweiterungsmöglichkeiten. Zur vorhandenen und angekündigten Software gehören Extended BASIC, 6809-Crossassembler und PASCAL. Dieses System wurde kürzlich hier vorgeführt und bestach nicht nur durch seinen professionellen Aufbau, sondern auch durch die schnelle Zusammenarbeit mit den Disketten und den leistungsfähigen Assembler/Editor. - Die Brutech Floppies sind auch an KIM, AIM und SYM betreibbar. - Info: Neumüller GmbH, Eschenstr. 2, 8028 Taufkirchen, 089 - 61 18 - 216.



#

Viel nützliches Zubehör für PET und cbm-Geräte findet man im Angebot von phs/SLS. Durch unterschiedliche Schwerpunktsetzung in den einzelnen Firmen in Hannover, Frankfurt und Kaufbeuren ist es gelungen, ein überaus vielfältiges Angebot für alle PET/cbm-Besitzer zu erstellen. Natürlich kann in einer Anzeige nicht alles wiedergegeben werden, hier jedoch einige besondere Leckerbissen:
ALPHA-Monitor nennt sich ein äußerst komfortables Monitor-Programm für die PETs der 2000er-Serie, das einen sehr komfortablen residenten Monitor für den alten PET darstellt. Neben allen Befehlen, die auch der TIM des cbm kennt, verfügt ALPHA zusätzlich über die Befehle FIND,HUNT,PRINT,TRACE und DISASSEMBLE. Alle Vorgaben sind mit Bildschirmmasken dokumentiert (DM 230,-). Der Monitor wird im EPROM geliefert und befindet sich auf einem Adapter, der die 4k-ROM-Fassungen in zwei 2k-Bereiche aufteilt. Somit sind Toolkit ($ B000-B7FF) und ALPHA ($ B800 - BFFF) gleichzeitig verfügbar und belegen nur einen EPROM-Steckplatz. Die fehlenden ROM-Steckplätze werden durch ein entsprechendes Interface zur Verfügung gestellt (DM 120,-), das einfach auf den Memory-Expansion-Bus aufgesteckt wird und seine Stromversorgung aus dem Anschluß des 2.Recorders entnehmen kann. Sollte der ME-Anschluß durch die dynamische Speichererweiterung (16k - DM 760,- , 32k - DM 960,-) belegt sein, so stehen zwei Anschlußalternativen zur Verfügung: das Interface INT 36 mit den drei ROM-Steckplätzen wird auf Wunsch auch kabelkonfektioniert geliefert oder Sie verwenden einen Dreifachverteiler (INT 34 - DM 89,-) für den ME- Bus.
Das ständig wachsende Angebot an Software im ROM/EPROM überfordert schon bald die drei Steckplätze des cbm. Hier konnte Abhilfe geschaffen werden :
mit den Adaptern INT 50 (zweimal 2k je Fassung nutzbar , DM 58,-), INT 51 (wahlweise eines von zwei 2k-ROMs, schaltbar, DM 78,-), INT 54 (4 umschaltbare 4k-Plätze für eine Fassung, DM 89,-) und INT 55 (3 mal 4 umschaltbare 4k-Plätze für 3 Fassungen, DM 258,-). Alle schaltbarenPlatten können demnächst auch softwaremäßig umgeschaltet werden (Steuerplatte i.V.).
Bequeme Benutzung Ihres PET/cbm wird bei phs/SLS großgeschrieben :
Besitzer "alter" PETs können unter der Bezeichnung S03 eine Platine zur Aufnahme von zwei weiteren Betriebssystemen beziehen und haben somit die Möglichkeit, alle drei Betriebssysteme (2000,3000 und 4000) in Ihrem Rechner zu verwenden (DM 350,- ohne Betriebssysteme).
Auch um die Lesesicherheit bei den eingebauten Kassettenrecordern hat man sich erfolgreich Gedanken gemacht: SPEEDY-TAPE heißt ein in das RAM ladbares Programm, das die Lesesicherheit der Recorder um ein vielfaches steigert; die Geschwindigkeit steigt übrigens auch : von 300 Baud bisher auf 2500 Baud. Somit laden und schreiben Sie Programme von 8k Umfang in 25 Sekunden sicher auf Band (DM 90,-). Dieses Programm wurde oft kopiert, leider dabei aber auch in der Leistungsfähigkeit vermindert. Schade für alle, die sich eine "billige" Kopie besorgt haben..
PASCAL ist zweifellos in Mode. Mehrere Anbieter versuchen, sogenannte "Compiler" an den Verbraucher zu bringen. Was muß ein solcher Compiler leisten ?
phs/SLS versteht unter Compiler ein Programmpaket, das PASCAL Quellsource in direkt ladefähige Maschinenprogramme übersetzt. Wenn die Übersetzung einmal gemacht wurde, dann ist der Compiler im RAM des cbm überflüssig. Compilierte Programme können also maximal 31763 Bytes lang sein und arbeiten gänzlich auf Maschinenebene. Dabei sollten aber alle Möglichkeiten des PET/cbm erhalten bleiben, wie z.B. alle Ein-Ausgabe-Routinen von und zur Peripherie, wie diese nun auch aussehen mag. phs/SLS addiert deshalb einen Identifier zum Standard-PASCAL nach Jenssen/Wirth: "CALLPET" ermöglicht den Aufruf von BASIC-Subroutinen mit max. 2.5 kB Länge. Das nennen wir einen Compiler, und deshalb sind wir auch teurer als andere Anbieter von " Compilern".
Ein vorläufiges Handbuch (DM 15,-) kann einzeln bezogen werden, der Compiler kostet 998,- DM.

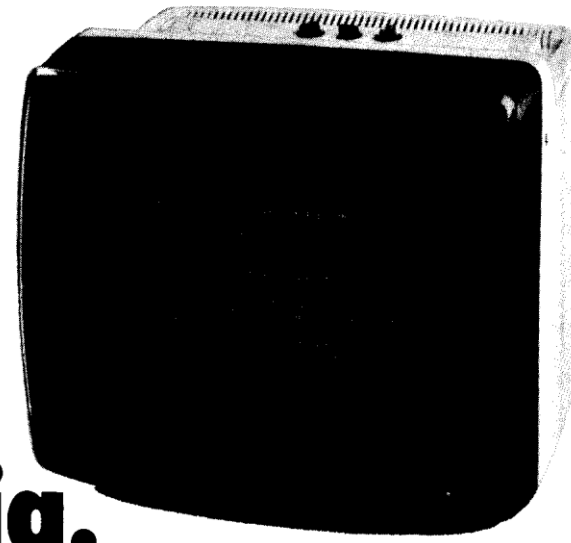
Die sportliche Betätigung bei langen Programmierabenden(-nächten) soll nicht zu kurz kommen. phs/SLS bietet deshalb an: Trimmtrainer-Programm komplett mit einem Heimtrainer zur Muskellockerung. Ein medizinisch abgestimmtes Programm, das Ihnen auch genug Pausen beim simulierten Waldlauf lässt. Der Trimmtrainer wird einfach an der Decke oder in einem Türrahmen befestigt, Sie steigen mit allen vieren ein und werden staunen, wie fit Sie nach einigen Trimmtouren sind !
Dies ist nur ein kleiner Ausschnitt aus unserem Gesamtangebot, fordern Sie deshalb unseren Katalog an, in dem fast 1000 Programmangebote und ca. 200 Interface verzeichnet sind. Wegen des umfangreichen Materials (ca. 300 g Papier !) verschicken wir den Katalog nur gegen DM 5,- Schutzgebühr, die allerdings bei einem Einkauf von über 50,-DM Nettowarenwert vergütet werden. Ab DM 150,- versenden wir frei Haus, ansonsten werden 5,-DM Porto und Verpackung pauschal berechnet.
Alle Preisangaben verstehen sich zzgl. 13 % Mehrwertsteuer.
phs-EDV Beratung M.Penzkofer , Teichstr. 9 ,3000 Hannover 91 , Tel: 0511 453817
SLS Udo Engelhardt , Wagengasse3, 6230 Ffm 80 , Tel. 0611 303212
phs-EDV Veratung K.Schneider , Eichenmähderweg 74, 8950 Kaufbeuren 2
Tel. 08341 62447

Unsere Leistungen in Stichworten:
Interface
Zweite Kassette mit Billig-Recorder ; Schaltinterface 220 Volt ; Video-Interface für den Anschluß eines externen TVs ; Stromversorgung 5 Volt (positiv oder negativ) ; User-Port-Kontroller mit LEDs ; Joysticks ; CW- und RTTY- Interface ; Telefon-Interface ; elektronisches Bandzählwerk ; Speichererweiterungen statisch und dynamisch ; Stromversorgung +5V, +12V, -5V ; A/D-Wandler ; Stand-alone System 64P ; BASIC-Switch ; Schaltverstärker ; Eprommer ; Multi-Control-Connector für User-Port ; Tongenerator CB2 ohne externe Versorgung ; Interface für Video-Recorder-Steuerung ; u.a.m.
Software
Utilities ; Textverarbeitung ; Buchhaltung ; Verwaltungsprogramme ; Demos ; Astrologie ; Mathematik ; Programme nach Kundenwunsch
Firmware
Monitore ; Basicerweiterungen ; PIC-Chip (Graphik) ; Stapelverarbeitung mit cbm ; u.a.m.
Zubehör
Disketten ; Kassetten ; Joysticks ; Stecker-Steckkontakte ; Ersatzteile ; Sortimentskästen ; Spezialbauteile ; Literatur ; Drucker mx 80 und es 100 ;
Weitere Leistungen:
Vermittlung von Kontakten ; praktische Tips mit jedem Katalog ; An- und Verkauf von Systemen (z.B.: IBM-Kugelkopf, gebraucht - DM 1200,- + 13% MWSt.)

WAS KANN MAN MEHR BIETEN ?

p h s
E D V - Beratung
Michael Penzkofer
Teichstraße 9
3000 Hannover 91
Tel.: 05 11 / 45 38 17

Wer beruflich »in die Röhre schaut« sieht bei uns grün. Und dies nicht nur scharf, sondern vielmehr auch ruhig.



Für alle, die mit Bildschirmdaten leben, entwickelte Sanyo den augenfreundlichen Typ DM5912CX. Denn EDV-Informationen lesen, kann weitaus deutlicher und erholsamer sein. So schrieb uns ein anwendungserfahrener Experte: „Der von Ihnen gelieferte Monitor ist ein wirklich angenehmes Display für den Mikroprozessor. Das grüne Bild steht scharf und ruhig und beansprucht mein leider krankes Augenlicht in keiner Weise.".
Das alles wurde von uns in ein formschönes, elfenbeinfarbiges Kunststoffgehäuse mit Rauchglas-Frontblende eingebaut.

Technische Hauptmerkmale:

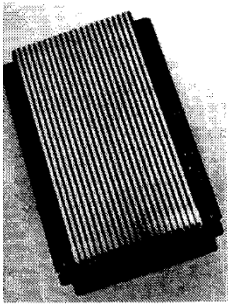
Bildschirmdiagonale:	12" (31 cm)
Videobandbreite:	18 MHz
Geometriefehler:	kleiner als 1%
Displayformat:	80 Charakter, 24 Zeilen
Eingangssignal:	Standard Video 1 V_{ss}

SANYO
VIDEO-SYSTEME

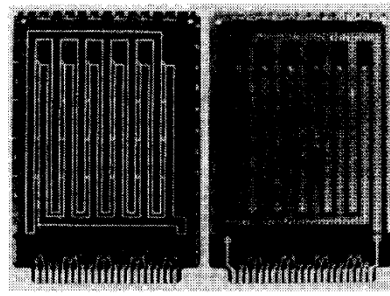
Ausführliche Informationen mit Fachhändler-Nachweis:
SANYO Video Vertrieb GmbH. & Co., Lange Reihe 29, 2000 Hamburg 1, Telefon (040) 24 62 66

Verkauf von Leiterplatten, Steckadapter, Relaisplatinen

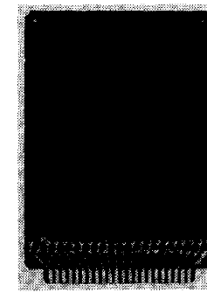
1



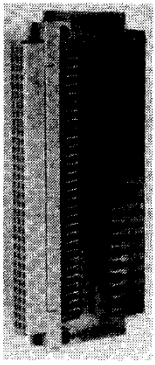
2



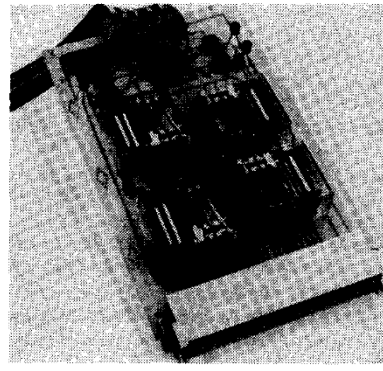
3



4



5



6

	p.St.	bei Abnahme von 3-5	von 6-10
1. 3690 Card Extender	DM 67,95	DM 57,75	DM 50,95
2. 3682-2 DIP Plugboard	DM 34,05	DM 28,95	DM 25,55
3. 3662 DIP Plugboard	DM 31,10	DM 26,45	DM 23,35
4. C 991 Steckadapter	DM 59,50	DM 50,55	DM 44,65
5. E 991 Relaisplatine	DM 210,60	DM 179,05	DM 157,95
6. E 941 dito m. Halter	DM 261,80	DM 221,85	DM 201,35

Alle Preise zuzüglich MWSt und Versandspesen. Lieferung ab Lager Köln, Zwischenverkauf vorbehalten. Weitere techn. Beschreibung auf Anfrage.

STECKER
INGENIEURBURO

5000 KÖLN 60 (Niehl)
Postfach 60 07 66
Delmenhorster Str. 20
Telefon (02 21) 74 51 12

Systemerweiterung für AIM 65 / PC 100

auf Europakarten

Einheitlches Format 100x160 mm, einheitlich nur 5 Volt.
Vollständig kompatibel zum AIM-Systembus und zur AIM-Software. Alle Karten mit 44-pol. und 64-pol. a/c-DIN-Stecker lieferbar. Anschlußfertig und ausführlich getestet. 1 Jahr Garantie.

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

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

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

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

EPROM-Karte 32 kB für 8x 2532 oder 2716 oder pinkompatible CMOS-RAMs. ICs einzeln beliebig adressierbar und einzeln selektierbar (für Umschaltung zwischen BASIC, PI.65, FORTH, DOS etc.) ohne Eproms:
180,- + MWST = 203,40 DM

BUS-EXPANSION Daten- und Adreßleitungen gepuffert, Schreibschutz für 16x4 kB, 5 Steckplätze, kann auf ca. 12 Stecker ausgebaut werden, lieferbar für 64- (vorzugsweise)und 44-polige karten.
240,- + MWSt = 271,20 DM

BUS-TERMINATOR-Karte aktiver Busabschluß für voll- und teilbestückten Bus bis 1m. 60,- + MWSt = 67,80 DM

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

```
2114L 450 ns =  9,--    2114L 200 ns = 10,--    2114 CMOS   = 18,--
4116  200 ns =  9,--    4116  150 ns = 12,--    4164 200 ns = 90,--
2716  450 ns = 23,--    2532  450 ns = 42,--    6116 200 ns = 65,--
2732  450 ns = 45,--    6502, 6522, 6532, 6551, 8257-5: á 28,--
MC 6845 CP  = 68,--     74LS240, -241,-243,-244, -245, 8T28: á 4,50
```

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

DIPL.-ING. HORST NEUDECKER

INGENIEURBÜRO FÜR MESSTECHNIK

Mehringplatz 13
1000 Berlin 61

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

Video-Interface für AIM 65 / PC 100

Europakarte 100x160 mm, Stromaufnahme 5V, 1A, kompatibel mit den RAM-, ROM-, Analog-E/E-, Digitalinterfaces und anderen Baugruppen im Europaformat unseres Erweiterungssystems für AIM 65 und PC 100.

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

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

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

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

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

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

Preise:	AIM65-Videointerface	615,- +MWSt=	694,35 DM
	Option 1	86,- +MWSt=	97,18 DM
	Option 2	140,- +MWSt=	158,20 DM
	NEC-Daten-Monitor 12 Zoll, grün (P31), 20 MHz, für Lichtgriffel geeignet	582,- +MWSt=	657,66 DM
	NEC-Daten-Monitor 12 Zoll, farbig (RGB), für 25 Zeilen zu je 80 Zeichen	2356,- +MWSt=	2662,28 DM
	Farb-Video-Monitore für Darstellung in Farb-TV-Qualität (25 Zeilen x40 Zeichen ab 1200,-		
	Centronix 737-2 Nadeldrucker: Normal-, Kompress- (132 Z./Z.), Proportional- (9x9 Matrix) -Schrift, ASCII/deutscher Zeichensatz umschaltbar, Endlospapier und Einzelblatt, mit AIM-Interfacesoftware	1592,- +MWSt=	1798,96 DM

DIPL.-ING. HORST NEUDECKER

INGENIEURBÜRO FÜR MESSTECHNIK

GWK

GESELLSCHAFT
FÜR TECHNISCHE ELEKTRONIK mbH.

NEU! NEU! NEU! NEU!

6809 Multiuser/ Multitasking Computersystem

auf Europakarten

CPU-Karte 6809

Floppy Disk-Controller

Seriell- und Parallel-I/O

AD/DA-Converter

RAM 32K

EPROM Board 16/32 KByte

Bus Board 64/96

GWK Gesellschaft für Technische Elektronik mbH.
Asternstraße 2
D-5120 Herzogenrath
Tel.: 024 06 - 6 23 94

65xx MICRO MAG

COMPUTING · SOFTWARE · HOBBY

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

65xx MICRO MAG erscheint zweimonatlich. Beiträge, die nicht besonders gekennzeichnet sind, stammen vom Herausgeber. - Offsetdruck: Druckartist Gerhard M. Meier, Hamburg 70.

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

Nachliefermöglichkeiten: Die Hefte 1-6 sind nur noch als Buch lieferbar, siehe Anzeige unten. Die Hefte 7-17 können einzeln oder komplett nachgeliefert werden, und zwar zum Endpreis von DM 7,80/Stück.

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