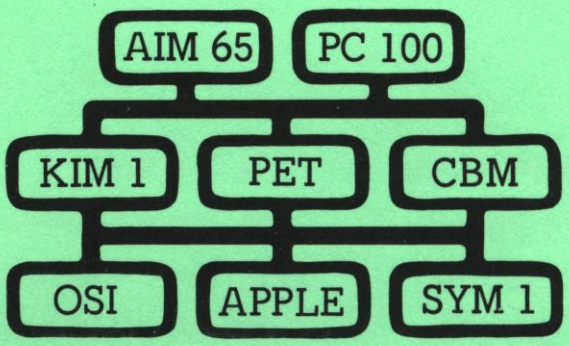
# 65xx MICRO MAG

COMPUTING · SOFTWARE · HOBBY

DM 8,50 Nr. 23 Februar 1982




## Inhaltsverzeichnis

# Rockwell's AIM 65/40

Heft 16 dieser Zeitschrift enthielt bereits im Dezember 1980 eine Vorbesprechung des Rechnersystems. Nunmehr kann eine ausführlichere Darstellung gegeben werden, nachdem hier dankenswerterweise ein Vorführsystem der Fa. Rockwell International für einige Zeit zur Verfügung stand. Wir gehen dabei auf die Hardware, das Betriebssystem und auf die Dokumentation ein.

**1. Hardware**

**1.1. Die Hauptplatine**

Der AIM 65/40 ist ein Single Board Computer (SBC) mit einer 8-Bit-CPU 6502, mit Interfacebausteinen, die den Anschluß von Ein- und Ausgabegeräten gestatten, sowie einem I/O-ROM (4K), das den Betrieb der Interfaces ermöglicht. Zu dieser Grundausführung, die bereits den Einsatz des Computers als Anwendergerät erlaubt, gibt es Ausbauoptionen für diverse Anwenderwünsche. Es sind dies ein 8K-Monitorprogramm in ROMs, Assembler (angekündigt) und BASIC (angekündigt), eine Tastatur, ein 40 Zeichen breiter Thermodrucker und ein 40-Zeichen-Fluoreszenzdisplay. Die Ausrüstung des Computers kann damit sehr modular erfolgen, anschließbar sind ferner die RM 65-Module. Abb. 1 gibt einen schematischen Überblick. Die umlaufende Linie ist die Begrenzung des SBC gegenüber der Peripherie.

Welches sind nun die wichtigsten Unterscheidungsmerkmale gegenüber dem vieltausendfach eingesetzen und bewährten AIM 65? Man darf sie wohl wie folgt zusammenfassen: Die CPU ist vom Dienst für die Peripherie weitgehend entlastet. Die 'subassemblies' Display und Drucker haben jeweils eine eigene Intelligenz mit 6504-CPU, Speicher und Interfacebaustein. Auch die serielle Schnittstelle hat mit dem 6551 ACIA eine weitgehende Eigenverwaltung. Weitere wichtige Unterscheidungsmerkmale sind: Die Hauptplatine kann bis zu 48K RAM-Speicher aufnehmen und bis zu 32 K Festwertspeicher, es ist ferner ein memory banking, eine Speicherumschaltung möglich, die einen Adreßraum von 131 K erreichbar macht. Hinzu kommt die Fähigkeit, verschiedenen Interruptquellen eine individuelle Priorität zuzuordnen.

Unterscheidend sind ferner sämtliche Kontaktleisten, das Tiefenmaß (ca. 320 mm), I/O-Adressen, Leistungen, Pointer und Einsprungadressen des Betriebssystems und die Möglichkeit des write protect, das ROM enable sowie die Konfigurationsmöglichkeiten durch Jumper. Und schließlich ist der Expansionsbus gebuffert. Auch die ASCII-Tastatur bietet durch 57 Tasten einschl. 'All Caps' (Umschaltung des Zeichensatzes), das Vorhandensein von 8 Funktionstasten nebst 'RESET' und 'ATTN' erweiterte Möglichkeiten.

Hinsichtlich des bekannten Vorläufers AIM 65 sind folgende Merkmale gemeinsam: Der Computer wird ohne Stromversorgung und Gehäuse geliefert. Diese items sollen jedoch in Vorbereitung sein. Trotz der Verwendung dynamischer RAMs kommt die Hauptplatine mit +5 Volt aus, die Hilfsspannungen werden 'on board' erzeugt. Bei Verwendung eines Thermaldruckers sind zusätzlich +24 Volt erforderlich. Die Spannungen für Drucker und Display werden durch die Hauptplatine hindurchgeleitet, die auch Bohrungen und Stecker für diese Peripherie enthält. Gemeinsam sind ferner eine für den Anwender freie und auf eine eigene Kontaktleiste herausgeführte VIA 6522, das Audio-Cassetteninterface für zwei fernsteuerbare Cassettenrecorder und das 20 mA TTY-Interface, letztere beide auch auf eine besondere Kontaktleiste geführt. Zusätzlich ist das serielle RS232-Interface mit eigener Kontaktleiste. Gemeinsam auch die Bus-Expansionsleiste z.B. für den Anschluß von RM 65-Modulen.

Von der Hardware her läßt sich der AIM 65/40 SBC damit zunächst als ein größerer und leistungsfähigerer Computer kennzeichnen. Mit bis zu 48 K RAM oder bis zu 32 K Festwertspeicher 'on board' wird man den Computer in vielen Fällen ohne Erweiterungsmodule betreiben können, und zwar sowohl als Anwender- wie auch als Entwicklungssystem. Der Computer ist weiterhin vielfältig konfigurierbar und zwar durch Jumper und Dipschalter für Speicher-Select und -banking. Das I/O-ROM ab Adresse hex F000 enthält die Grunddienstleistungen der Ein- und Ausgabe, für den

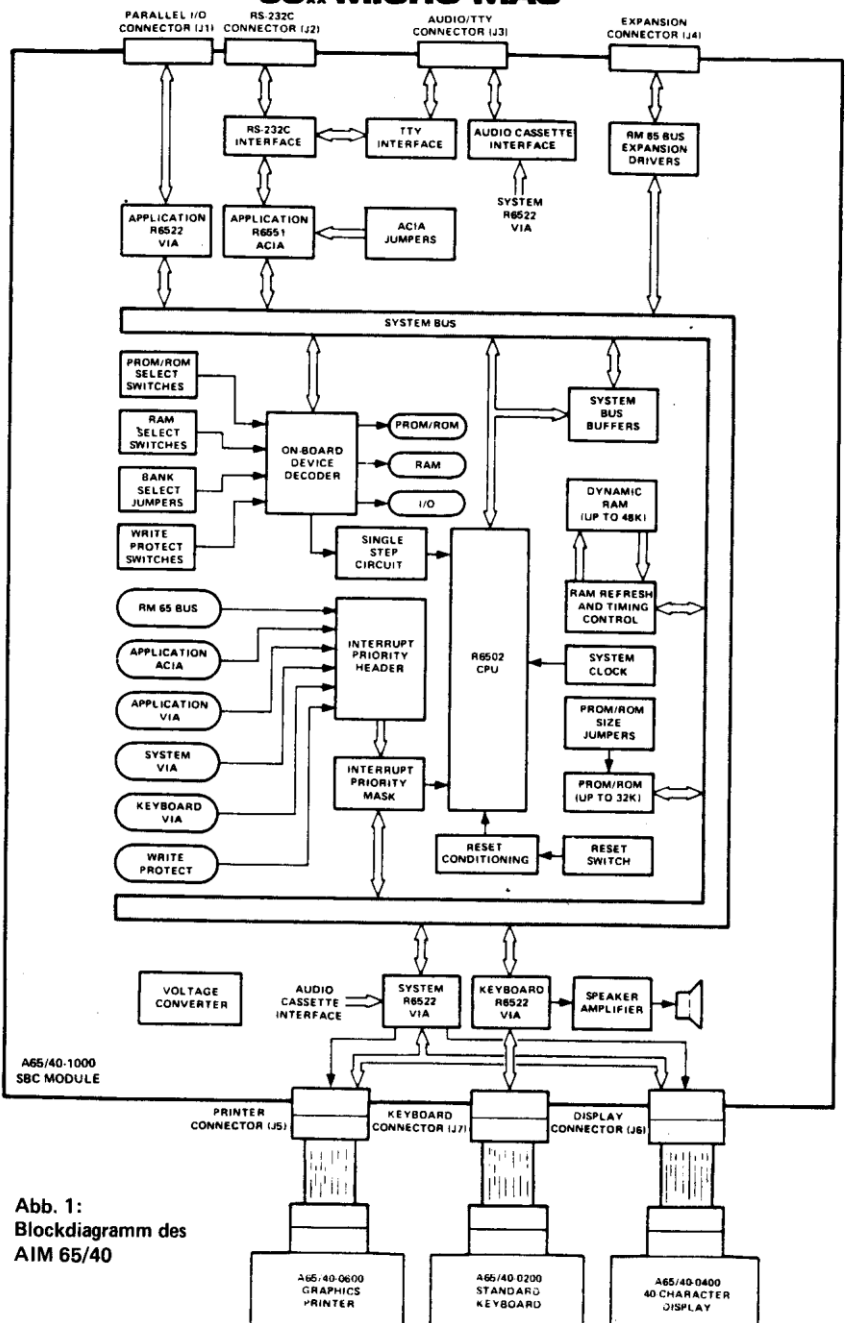



**Abb. 1:**
**Blockdiagramm des AIM 65/40**

Auto-Start in ein Anwenderprogramm. Der System-I/O-Bereich ist in der 'äußersten Ecke' in den Adressen FF80 bis FFDF angesiedelt. Das Interface zu Thermalprinter und Display mit seinen 8 Datenleitungen sowie den aktiv-low-Signalen STROBE, ACK und BUSY ist von Haus aus als Centronix-kompatibel anzusprechen.

**1.2 Der Thermodrucker**

Wie beim AIM 65 handelt es sich um einen Punkt-Matrix-Drucker, allerdings mit eigener Intelligenz versehen. Das Thermoprinzip gewährleistet den geräuscharmen Lauf. Es wird in 40 Zeichen Breite gedruckt. Die Zeichen werden in 7x8 Matrixpunkten dargestellt. Im normalen Modus stehen 96 ASCII-Zeichen sowie 160 semigrafische und Sonderzeichen zur Verfügung. Der Drucker erzeugt 240 Zeilen pro Minute. Daneben können im Grafik-Modus beliebige Muster oder Zeichnungen abgebildet werden, wobei die horizontale Auflösung 280 Punkte beträgt.

**1.3 Das Fluoreszenz-Display**

Dieses Modul ist ebenso wie der Thermodrucker über die 8-Bit-parallele Schnittstelle mit dem Computer verbunden und mit eigener Intelligenz versehen. Es hat ebenfalls 40 Anzeigestellen von je 6x6 mm Größe. Die Zeichen werden aus 16 Balkensegmenten gebildet. Im Normalmodus beträgt der Zeichenvorrat 96 ASCII-Zeichen (Kleinbuchstaben werden als Großbuchstaben mit einem zusätzlichen Punkt abgebildet) sowie 96 semigrafische und Sonderzeichen. Im Grafik-Modus kann jede Art der Darstellung aus 16 Segmenten erreicht werden. Die Anzeige ist gut ablesbar. Ebenso wie der Drucker versteht auch das Display Kontrollcodes als Befehlscodes. Beide Geräte können per Programm oder von der Tastatur her direkt mit Befehlen angesprochen werden.

**1.4 Die Tastatur**

Die ASCII-Tastatur ist über eine VIA 6522 mit dem Rechner verbunden und wird von diesem in einer 8x9 Matrix dekodiert. Sie akzeptiert Klein- und Großschreibung (SHIFT). Die Rast-Taste ALL CAPS bewirkt die ständige Umschaltung auf Großschreibung. Neben den Zeichentasten, RESET und ATTN stehen 8 Funktionstasten zur Verfügung, die vom Anwender nach seinen Bedürfnissen auf seine Programme vektoriert werden können.

**2. Die Firmware**

**2.1 Benutzung des Adreßraumes**

Auch beim Betriebssystem hat Rockwell neue Wege beschritten. Die Adressen der Ein- und Ausgabebausteine werden ab hex FF80 dekodiert. Zum Ausrüstungsstandard gehört das I/O ROM, das die Systemkonfiguration beim Einschalten und die Bedienung der Systeminterfaces besorgt. Deutlich getrennt davon sind die beiden optionalen Monitor-ROMs für Entwicklungszwecke, die ab A000 und B000 residieren. Sprach-ROMs werden ab C000 gesteckt (BASIC, PL 65, FORTH). Der Assembler wird zwischen 9000 und 9FFF dekodiert.
Der untere Speicherbereich ist je nach Konfiguration und Sprachen bis 04FF vom System belegt. Der darüber liegende Bereich bis 07FF ist für die Benutzung durch Sprachen und RM 65-Module reserviert. Die Stackseite (01) ist voll und ganz für die CPU freigehalten. Die Zeropage ist im Anwendersystem praktisch frei, sie wird aber stark von Assembler und BASIC benutzt.

Hinter der aufgezeichneten Zuordnung der Adreßbereiche (memory map) ist folgende Überlegung zu sehen: Ein Anwendersystem soll möglichst viel 'freie Bahn' haben. Dafür werden die Zero Page und die Stackseite möglichst frei gehalten. Die Vektoren und Variablen des Systems sind oberhalb Adresse 0200 angeordnet. Im Extremfall (keine Sprachen, kein Monitor) wird der empfohlene Bereich von 0800 bis EFFF voll verfügbar. Wird nur eine Sprache ab C000 verwendet, so ist für sie der Arbeitsbereich von 0800 bis BFFF frei. - Ein Entwicklungszwecken dienendes System ist bis 8FFF (mit Assembler) oder bis 9FFF (nur mit Monitor) für Programme, Text, Variablen usw. frei, gleichwohl können auch Sprachen benutzt werden.

Wo in dieser Landschaft ein an die Busse angeschlossenes Video-Interface zweckmäßig zu dekodieren wäre, um keine Behinderungen im Adreßraum zu haben, ist nicht unmittelbar ersichtlich. Die

Empfehlung könnte auch lauten: Man verwende ein Interface, das über die 8 Bit parallele Schnittstelle betrieben wird oder über eine serielle Schnittstelle.

### 2.2 Der Monitor/Editor

Diesem Abschnitt ist voranzustellen, das das vorübergehend benutzte Vorführgerät aus der Vorserie stammt und viele der nachstehend genannten Leistungserweiterungen bezüglich des Editierens und der Cursorbewegung noch nicht aufwies. Ebenso hatte es keinen Assembler, der ja erst demnächst zur Auslieferung kommt. Wegen der festen Disposition aller Geräte konnte Rockwell offensichtlich keines aus der laufenden Serie für die Besprechung abzweigen, was für den Autor enttäuschend war. Es sind diese zusätzlichen Befehle, die den AIM 65/40 auch als Entwicklungssystem leistungsfähiger gemacht haben. Statt umständlicher verbaler Erklärungen geben wir nachstehend einen Abdruck der Monitor-/Editor-Befehlsliste. Der Betreiber eines AIM 65 findet darin viel bekanntes, so daß nur die Besonderheiten herausgestellt werden sollen.

## DEBUG MONITOR COMMANDS

**Monitor Control Commands**

| | |
|---|---|
| CTRL RESET | Enter and Initialize Monitor (Cold Reset) |
| RESET | Enter Monitor (Warm Reset) |
| ATTN | Non-Maskable Interrupt |
| ESC | Escape to Monitor Command Level |
| E | Initialize Text Buffer and Enter Text Editor |
| C | Recover Text Buffer and Enter Text Editor |
| T | Reenter Text Editor |
| F1 - F8 | Enter Function 1 - Function 8 |
| + | Repeat Last Command |
| & | Execute Command String |
| O | Toggle Memory Bank |
| CTRL Z | Direct Peripheral Control |
| CTRL Z CTRL Z | SBC Module RAM Self Test |
| CTRL C | Clear Display And Home Cursor |
| CTRL N | Home Cursor |

**Display/Alter Registers**

| | |
|---|---|
| R | Display Register Contents |
| A | Display/Alter Accumulator |
| P | Display/Alter Processor Status |
| S | Display/Alter Stack Pointer |
| X | Display/Alter X Register |
| Y | Display/Alter Y Register |
| * | Display/Alter Program Counter |

**Display/Alter Memory**

| | |
|---|---|
| M | Display Selected Memory Contents |
| SPACE | Display Higher Memory Locations |
| - | Display Lower Memory Locations |
| / | Alter Current Memory Contents |

**Enter/Disassemble Instructions**

| | |
|---|---|
| I | Enter Mnemonic Instruction |
| K | Disassemble Memory |
| ; | Enter Symbolic Address |

**Execution/Trace**

| | |
|---|---|
| G | Execute Program |
| Z | Toggle Instruction Trace |
| J | Display Register Heading |
| H | Display Jump and Branch History |
| V | Toggle Symbol Table On/Off |

**Breakpoint Manipulation**

| | |
|---|---|
| ? | Display Breakpoints |
| # | Clear Breakpoints |
| 4 | Toggle Breakpoint Enable On/Off |
| B | Set Breakpoint |

**Load/Dump Memory**

| | |
|---|---|
| L | Load Memory |
| D | Dump Memory |
| F | Verify Memory |

**Peripheral Control**

| | |
|---|---|
| CTRL P | Toggle Auto-Print On/Off |
| PRINT | Print Display Contents |
| 1 | Toggle Recorder 1 Control On/Off |
| 2 | Toggle Recorder 2 Control On/Off |
| 3 | Verify Tape Checksum |

**Screen Oriented Commands**

| | |
|---|---|
| F1 (or CTRL Q) | Home Cursor on Line |
| F2 (or CTRL R) | Clear Line to Right |
| F3 (or CTRL S) | Toggle Insert Mode On/Off |
| F4 (or CTRL T) | Delete Character Under Cursor |
| F5 (or CTRL U) | Move Cursor Left (Left Arrow) |
| F6 (or CTRL V) | Move Cursor Right (Right Arrow) |
| F7 (or CTRL W) | Move Line/Cursor Down (Down Arrow) |
| F8 (or CTRL X) | Move Line/Cursor Up (Up Arrow) |
| CTRL A | Add (Insert) a Line |
| CTRL B | Break a Line |
| CTRL D | Delete a Line |

## TEXT EDITOR COMMANDS

**Editor Control Commands**

| | |
|---|---|
| S | Enter Screen Edit Mode |
| Q | Quit Editor and Enter Monitor |
| ESC | Return to Editor Command Level |
| + | Repeat Last Command |
| CTRL C | Clear Display and Home Cursor |
| CTRL N | Home Cursor |

**Line Oriented Commands**

| | |
|---|---|
| L | List Multiple Lines |
| R | Read Multiple Lines |
| I | Insert One Line |
| O | Overlay Current Line |
| K | Delete Multiple Lines |
| (SPACE) | Display Current Line |
| ? | Display Current and Last Line Addresses |
| G | Go to Line Number |
| U | Go up Multiple Lines |
| D | Go down Multiple Lines |
| T | Go to Top Line |
| B | Go to Bottom Line |
| F7 (or CTRL W) | Go Down One Line |
| F8 (or CTRL X) | Go Up One Line |

Neu ist, daß man von der Tastatur her sowohl einen warmen wie einen kalten RESET auslösen kann. Die Taste/der Befehl ATTN löst einen NMI-Interrupt aus und führt ohne RESET-Funktion in die Befehlsebene des Monitors zurück. Befehle werden gespeichert und können mit '+' wiederholt werden. Das ist z.B. bei längeren Befehlen (DUMP mit Adressen, Namen und Gerät) arbeitserleichternd. Noch größeren Komfort bietet der Befehl '&': Abarbeitung von Kommandostrings, die als Editor-Textfile abgelegt wurden. Man kann also ganze Folgen von Monitor-Befehlen mit wenigen Tastendrücken abarbeiten lassen, z.B. Assemblierung, Setzen von Breakpoints (auch symbolischer), Ausführen ab Startadresse mit weiteren Aktionen an den Breakpoints usw..

Mit 'C' und nachfolgender Adreß- oder Labeleingabe kann man auf beliebig viele unabhängige Textbuffer ohne Neu-Initialisierung zugreifen. Es können auch Kommandostrings ('&') aus verschiedenen Textspeichern miteinander verknüpft werden. Mit ';' wird dem Monitor der Wert eines Symboles mitgeteilt, das dann auch auf der Monitor-Ebene weiterverwendet werden kann.

Weitere Unterschiede bei den Monitor-Befehlen: Der M-Befehl schlägt den Inhalt von 8 Speicherstellen auf und versucht auch, die gefundenen Bytes als ASCII-Zeichen rechts im Display/Printer zu interpretieren. Auf diese Weise findet man leicht Textstellen. Man kann auch in Richtung auf niedrigere Adressen hin aufschlagen. Bei der Registeranzeige ('R') wird der Prozessorstatus nicht nur hexadezimal, sondern auch als binäre Strichliste angezeigt. Mit 'K' ist eine Disassemblierung auf beliebige Peripherie und sogar in ein Editor-Textfile (OUT=M) möglich.

LOAD und DUMP können mit einem Adressen-Offset gegen die wahre Herkunftsadresse bewirkt werden. Der F-Befehl führt einen protokollierten Vergleich Memory gegen ein beliebiges File durch. Im TRACE-Modus können bis zu 8 Breakpoints benutzt werden.

Auch der **Editor** ist leistungsfähiger geworden. Das Arbeiten mit einem beweglichen Cursor (Zeiger in der offenen Zeile) läßt natürlich ein viel beweglicheres Arbeiten zu, als die Textveränderungen mit Hilfe des CHANGE-Befehles. Wir kennen das von allen bildschirmorientierten Geräten (wie gesagt, das konnte hier leider nicht erprobt werden). Diese Art der Editierung wird auf der Befehlsebene des Editors mit 'S' hereingerufen und mit 'ESC' verlassen. Man kann die Zeilen vorwärts und rückwärts laufen lassen (REPEAT bei gedrückter Taste). In der offenen Zeile kann man Zeichen überschreiben, löschen und einfügen und längere Zeilen beliebig zerlegen (CTRL/B). In der normalen Befehlsebene arbeitet der Editor mit virtuellen (scheinbaren) Zeilen-Nummern. Mit 'G' und Zahl setzt man auf eine Zeile mit Nummer auf. Die Rangierbefehle wie UP und DOWN arbeiten mit Argument (Menge der Zeilen).

Der OVERLAY-Befehl vereinigt die Funktionen von KILL und INSERT, Ersatz einer Zeile durch eine andere. Die Befehle 'F' und 'C' (FIND und CHANGE, nicht in der vorstehenden Aufstellung enthalten, gleichwohl aber vorhanden) arbeiten in der gewohnten Weise zeichenkettenorientiert.

**In der Summe ist zum Monitor/Editor zu sagen:** Die Entwicklung von Software ist zeit- und damit kostenaufwendig. Die entstehenden Kosten übersteigen schnell den Anschaffungspreis eines Systemes. In Hinsicht darauf ist jede Hilfe zu begrüßen, die das Schreiben und Austesten von Programmen erleichtert und beschleunigt, auch wenn anfänglich mehr investiert werden muß. Der AIM 65/40 bietet sowohl vom Monitor her (Befehlswiederholgung, Kommandostrings, Annahme von Labels, Setzen von Breakpoints) wie auch von den Editierungsmöglichkeiten (mehrere Textspeicher können unterhalten werden, Cursor-orientierte Editierung, virtuelle Zeilennummern) die arbeitssparenden Hilfen, auf die man aus heutiger Sicht nicht mehr verzichten sollte. Ein Teil ähnlicher Dienstleistungen, speziell des Editors (z.B. LINED), wurde vor allem durch Veröffentlichungen in dieser Zeitschrift auch für den AIM 65 nachimplementiert und werden sich künftig aus den gegebenen Anregungen beim 65/40 noch für diesen nachvolziehen lassen. Beim AIM 65/40 stehen sie neben der zusätzlichen Hardware von Anfang an zur Verfügung und fügen sich in das Firmware-System ein.

### 2.3 Der Assembler

Hinchsichtlich des Assemblers steht dem Anwender z.Zt. nur sein ca. 80-seitiges Handbuch zur Verfügung, die ROMs sind noch nicht lieferbar. Man kann sagen, daß er weitgehend mit dem des

AIM 65 identisch ist. Zusätzlich hat er die Möglichkeit der Neu-Assemblierung unter den im vorherigen Lauf geltenden Verfügungen hinsichtlich Ein- und Ausgabe (der Kommandostring ist gespeichert). Listings haben einen erweiterten Seitenkopf sowie den beim AIM 65 erst nach Anwendung eines Reformatters möglichen Zeilenaufbau mit vorangestellter Speicheradresse, Hexcode und rechts daran anschließendem Quelltext.

**3. Dokumentation**

Die dem AIM 65/40 beigegebene Dokumentation ist wiederum vorbildlich: Anwenderhandbuch, I/O-ROM Program Listing, Monitor/Editor Program Listing, Assembler User's Manual, AIM 65/40 Summary Card und Wandplan der Hardware. Die Bücher sind zusammen ca. 7 cm dick. Für Monitor und Editor ist jeder Befehl mit Beispielen erklärt, die Bedienung der peripheren Einheiten (Display, Drucker, serielle Schnittstellen, Cassetteninterface, Tastatur) nebst Signalen und Signalbelegung und hardwaremäßigem Aufbau ist dokumentiert. Besondere Abschnitte sind dem Gebrauch des I/O-ROMs nebst Interruptbehandlung und dem Einsatz der Anwender-VIA gewidmet.

**4. Zusammenfassung**

Nach der voraufgehenden Besprechung darf man den AIM 65/40 von Rockwell International als einen sehr wandelbaren, leistungsfähigen und gut durchdachten Rechner bezeichnen:

**a) Der OEM-Kunde** (der gewerbliche Verarbeiter) erhält mit dem Bezug der Hauptplatine ein Gerät mit weit offenem durchgehendem Adressenraum (empfohlen ab hex 0800 bis EFFF), geregeltem RESET, I/O- und Interrupthandling. Diesen Adreßraum kann er in ausreichend weiten Grenzen nach Bedarf mit RAM oder mit Festwertspeichern belegen. In einer solchen Konfiguration stehen ihm für das Interfacing 3 VIAs, eine TTY-, eine RS232- und eine Audio-Cassettenschnittstelle zur Verfügung. Die drei VIAs (zusammen 60 E/A-Pins) können dabei nach Wahl für Meß- und Steuerzwecke benutzt werden. Befehle und Informationen werden dabei ggfs. über die seriellen Schnittstellen ausgetauscht.

**b) Der AIM 65/40 bietet sich für die verschiedensten hochkarätigen Sprachimplementierungen an:** Von den 32 K Steckplätzen für ROMs benötigt die Ein- und Ausgabe ab F000 nur 4 K. Man kann also bis zu 28 K ROM mit fest zu steckenden Sprachmodulen belegen. Wenn man bedenkt, daß das für den AIM 65 entwickelte 'Instant PASCAL' 20 K Festwertspeicher belegt, so sind noch leistungsfähigere Versionen vorstellbar, auch für künftige andere Sprachen. Ihnen kann noch 'on board' ein großer RAM-Adreßraum für den Programmtext und für die Ablage von Variablen zur Verfügung gestellt werden. Im Zusammenhang mit dem 'memory banking' sind noch weiterweisenden Softwaresysteme vorstellbar. Es ist damit eigentlich nur die Frage der Zeit angesprochen, in der die gedankliche Arbeit zu Implementierungen führt. Hier gilt die vorsichtig ausgedrückte Erwartung: In etwa zwei Jahren werden wir uns wahrscheinlich weniger über den Preisunterschied im Verhältnis zum herkömmlichen AIM 65 unterhalten, als uns vielmehr über die leistungsfähigen Sprachimplementierungen wundern, die für dieses Gerät zur Verfügung stehen und die es attraktiv machen. Dabei kann man sich gut vorstellen, daß diese Fortschritte nicht nur unter der Firma Rockwell angeboten werden, sondern von mittelständischen Unternehmen, die den Rechner entsprechend ausrüsten. - Mit entsprechender Dekodierung ließe sich der AIM 65/40 auch als eine CP/M-ähnliche 'fast nur RAM-Maschine' betreiben. Man soll sich weiter vor Augen halten: Die schon implementierten und kommenden Sprachen (ggfs. Steuersprachen) wird man in den OEM-Versionen auch 'unsichtbar' einsetzen, sie treten nur mit ihren Prompts auf den Benutzer zu.

**c) Der Entwickler von Hard- und Software** erhält mit dem neuen Rechner nebst Monitor/Editor und Assembler ein leistungsfähiges Instrument, das ihm Zeit und damit Kosten zu sparen hilft. In dieser Konfiguration können auch Sprachen betrieben werden.

In der Summe sollte man den AIM 65/40 als großzügig konzipert und gelungen bezeichnen. Er ist ein Gerät zum Aufrüsten nach Bedarf: Mit Gehäuse und Netzteil und ggfs. Sprach-ROMs als Fertiggerät und ggfs. auch als Entwicklungssystem. Oder er ist ein Gerät zum Einbau in Textautomaten, Steuerungen usw.. Wegen dieses Leistungsspektrums sollte man auch der Erwartung Ausdruck geben, daß nach der Abdeckung der Entwicklungskosten mit den erreichten Stückzahlen eine Verbilligung eintritt, die den Markt zweifellos sehr elastisch erweitern würde. R.I.

*Michael Zimmermann, 6102 Pfungstadt*

# Der TRS-80 Color Computer

*Beschreibung und erster Erfahrungsbericht*

*Seit rund einem Jahr wird in den USA der Color-Computer der Firma Tandy-Radio Shack vertrieben. Was ihn für die Leser von MICRO MAG interessant macht, ist sein Prozessor, der 6809. Sicher hat es sowohl in den USA als auch in Europa eine Reihe von Systemen mit dem 6809 gegeben, diese waren aber mehr am selbst aufbauenden Benutzer orientiert. Mit dem Color-Computer kommt erstmalig ein 6809-System auf den Markt, das bewußt auf den Konsumenten zielt. Beachtlich ist, daß Radio Shack für diese Maschine seine bisherige Hardware-Politik, die ausschließlich auf die Z80 gerichtet war, aufgegeben hat; ein Tatbestand mehr, der für die Qualität des 6809 spricht.*

**1. Das Äußere**

*Der TRS-80 Color-Computer erreicht den Endkunden in einem Faltkarton von 50x50x20 cm Grösse. Beim Auspacken fällt die gute Abpolsterung des Gerätes ins Auge. Der Computer selbst hat ein GFK-Gehäuse mit den Maßen 37x36x8 cm. In den vorderen abgeschrägten Teil ist die Tastatur eingelassen, das Gehäuse ist in silber-metallik gehalten, die Tastatur besteht aus grauen Buchstaben- und Zifferntasten, weißen Funktionstasten und einer roten Break-Taste. Die Tasten machen einen etwas taschenrechnerhaften Eindruck, die Tastenabstände entsprechen aber der Norm. Ein 'Fingerspitzer' wie für den PET 2001 ist also nicht erforderlich. An der Rückseite des Gerätes sind zwei Druckschalter angebracht. Auf der linken Seite bedindet sich der Netzschalter, rechts die Reset-Taste. Zu beachten ist, daß der Netzschalter nur die Niederspannung abschaltet und daß der Trafo also weiterhin mit dem Netz verbunden bleibt. Diese Beschaltung findet man in den USA auch bei anderer brauner und weißer Ware.*

*Auf der Rückseite sind weitere Buchsen angebracht. Über eine Cinch-Buchse wird das Fernsehsignal mit 75 Ohm herausgeführt. Zum Lieferumfang gehört weiterhin ein Umschalter für Computer und Antenne, wobei dieser von einem Wellenwiderstand von 300 Ohm ausgeht, auch hier entsprechend den Gegebenheiten in den USA. Neben der Video-Buchse befindet sich ein Schiebeschalter für die Umschaltung von Kanal 3 auf 4. Eine 5-polige DIN-Buchse dient zum Anschluß eines Cassettenrekorders mit Start-Stop-Betrieb. Eine vierpolige Buchse zur Realisierung einer RS-232-Schnittstelle ist vorhanden. An diese ist z.B. ein Drucker anschließbar. Zwei weitere 5-polige Buchsen dienen dem Anschluß von zwei Joysticks, vornehmlich für Spiele. Im weiteren werden Hard- und Software besprochen und abschließend über Erfahrungen berichtet.*

**2. Hardware**

*Die gesamte Hardware ist auf einer Platine untergebracht. Für den Trafo wurde extra eine Aussparung gelassen. Kernstück der Maschine sind 3 LSI-Chips: 6809 CPU, 6883 SAM und 6847 VDG.*

*Der Taktgenerator hat neben weiteren Bauteilen einen Quarz mit 14,318 MHz. Seine Freqenz wird dem 6883 SAM zugeführt, der zunächst weiter besprochen wird: Er teilt die Ausgangsfrequenz zunächst durch 4 für den Video-Takt von 3,57 MHz, zum anderen durch 16 auf 0,89 MHz als Frequenz für den 6809. Weiterhin handhabt der 6883 die Speicheradressierung für den 6847 VDG. Er ist zu diesem Zweck durch Ansprechen bestimmter Adressen auf die unterschiedlichen Modes des 6847 einstellbar.*

*Mit der Video-Speicheradressierung ist das Multiplexing und der Refresh der dynamischen RAMs eng verbunden. Hier werden vom 6883 alle erforderlichen Signale erzeugt. Als letztes ist der 6883 ein Adreßdekoder, der zum einen RAMs und ROMs als auch die I/O dekodiert, diese im Bereich der letzten Page. Die Reset- und Interruptvektoren werden dabei ausgespart.*

*Der 6883 ist insgesamt ein sehr vielseitiger Baustein, der eine Vielzahl sonst erforderlicher TTL-Schaltkreise einzusparen hilft. Leider ist er noch nicht für Europa freigegeben. Er bewirkt auch einen Video-Speicherzugriff oder einen Refresh in 01 und einen Speicherzugriff vom Prozessor in 02. Die erforderlichen RAS- und CAS-Signale werden ebenfalls aus der Taktfrequenz erzeugt.*

*Ein weiterer wichtiger Baustein ist der 6847 VDG, von dessen Funktionen, insbesondere bei der Speicheradressierung, bereits ein Großteil vom 6883 übernommen wurde. Der 6847 ist somit der Erzeuger für das Video-Signal, entweder direkt als Punktmatrix aus dem Speicherinhalt oder indirekt über einen Zeichengenerator. Auf die unterschiedlichen Möglichkeiten des 6847 soll hier nicht detailliert eingegangen werden. Es sei erwähnt, daß er eine Vielzahl von Bildschirmformaten unterstützt, so z.B. ein alphanumerisches Format mit 16x32 Zeichen oder eine Punktmatrix mit 256x 192 Pixels. Der Zeichengenerator ist ebenfalls auf dem 6847 untergebracht, auch eine Reihe von Farbdarstellungen ist möglich. Das Video-Signal inkl. Farbinformation wird einem MC1372 TV-Modulator eingespeist. Dieses Signal wird in einem Seitenbandfilter um das Tonsignal ergänzt und steht als solches an der Cinch-Buchse auf der Rückseite zur Einspeisung in die Antennenbuchse eines normalen Fernsehers nach NSTC-Norm zur Verfügung.*

*Als Prozessor wird der 6809E benutzt. Das 'E' bedeutet, daß es sich um die Version mit externer Taktzuführung handelt. Die erforderlichen Signale werden hier durch den 6883 erzeugt. In diesem Zusammenhang soll auf die Möglichkeiten und Vorteile dieses Prozessors nicht eingegangen werden sie sind dem interessierten Leser sicher geläufig. - Auf der Platine sind 2 Sockel für 8K x8-ROMs vorhanden. In einem steckt das 'Color-BASIC', der andere ist für die Aufnahme der Erweiterung dieses Color-BASIC vorgesehen. Der RAM-Speicher hat je nach Ausstattung 4K x1 oder 16K x1 dynamische RAMs. Welche Bauteile bei der ebenfalls angebotenen 32K-Version verwendet werden, ist nicht bekannt.*

*Dem Betrieb der Ein- und Ausgabe dienen 2 PIAs. An der einen ist die Tastatur angeschlossen, die andere dient zum einen der Erzeugung der Grafik-Modes für den 6847, zum anderen für Kassetteninterface, D/A-Wandler für Tonausgabe und Joystick-Eingabe sowie für die serielle Schnittstelle. Das Kassetteninterface arbeitet mit einer Übertragungsgeschwindigkeit von etwa 1500 Baud, eine volle Schwingung von 1200 Hz stellt eine '0', eine solche von 2400 Hz eine '1' dar.*

*Die Spannungsversorgung ist konservativ aufgebaut, sie liefert: 5V 1,35 A, 12V 400 mA, -5V 100 mA, -12V 100 mA. Für alle Spannungen mit Ausnahme der 5V werden 78xx bzw. 79xx-Regler verwendet, die 5V-Spannung wird hingegen mit einem 723 stabilisiert. Dank der dynamischen RAMs sind die Stromstärken sehr gering. Für Erweiterungen steht ein 40-poliger Stecker zur Verfügung, auf dem alle erforderlichen Signale herausgeführt sind. Dieser Stecker dient haupsächlich dem Anschluß der 'ROMPACKS' mit fertiger Software in ROMs.*

**3. Software**

*Der Color-Computer ist eine reine BASIC-Maschine. Logischerweise befindet man sich nach dem Einschalten in der BASIC-Eingabe-Ebene. DBASIC wird in zwei Ausbaustufen angeboten, einmal als 'Color-BASIC' für die 4K-Maschine, zum anderen als 'Extended Color-BASIC' für die 16K-Maschine. Beide Versionen scheinen von Microsoft zu stammen und sind deswegen mit anderen BASICs dieser Firma vergleichbar. Beim Vergleich mit dem des AIM 65 zeigt sich, daß Color-BASIC keine Integer-Variablen unterstützt, dafür aber Befehle für Cassetteneln- und -ausgabe hat. Ebenso ist eine Reihe von Befehlen für die Verwaltung des Bildschirmes im Semi-Grafik-Mode des 6847 vorhanden. Die Stringfunktionen sind im selben Maße wie beim AIM implementiert. Was aber fehlt, das sind die geometrischen Funktionen; nur der Sinus ist implementiert.*

*Das Extended Color-BASIC bietet eine Reihe von Möglichkeiten, die weit über das AIM-BASIC hinausgehen. Bei den Befehlen gibt es die Möglichkeit, neben BASIC-Files auch Dateien in Maschinensprache zu sichern und zu laden. Für den kommerziellen Anwender ist sicher das PRINTUSING von Interesse. Zur Programmentwicklung steht ein Editor für BASIC-Zeilen zur Verfügung. Ein RENUMBER oder DELETE von Zeilenblöcken ist möglich. Dem Austesten von Programmen dient die TRACE-Möglichkeit. Seine ganze Stärke zeigt das Extended Color-BASIC aber bei grafischen Darstellungen. Hier werden alle Grafik-Modes des 6847 unterstützt. Weiterhin gibt es Befehle für das Zeichnen von Linien, Kreisen, Quadraten und Rechtecken. Diese Figuren können wahlweise mit den zur Verfügung stehenden Farben ausgefüllt werden. Für komplette Zeichnungen steht ein Befehl DRAW zur Verfügung, der in Strings abgelegte Zeichenanweisungen interpretiert. Ein*

*ähnlicher Subprozessor steht mit PLAY für den Aufbau von Tonfolgen zur Verfügung. Auch bei den Funktionen bietet das Extended BASIC eine Reihe von Erweiterungen. Einmal sind mehrere trigonometrische Funktionen vorhanden. Zum anderen können bis zu 10 vorher definierte Maschinen-Unterprogramme aufgerufen werden. Für die Ausnutzung maschinensprachlicher Fähigkeiten ist der HEX $ -Befehl nützlich, der den Hexadezimalwert eines Byteinhaltes darstellt. Das Extended Color-BASIC dürfte insgesamt für grafische Darstellungen interessant sein. Das normale BASIC sollte für den Anfänger und Spieler voll ausreichen. Zeitvergleiche mit Benchmarks zeigen, daß das Extended Color-BASIC etwa im Mittelfeld des Rugg-Feldman-Tests rangiert, angesichts der niedrigen Clockfrequenz ein Beweis für die Fähigkeiten des 6809-Prozessors.*

*Neben BASIC werden für den Anwender 'Program Paks' (Warenzeichen von Tandy) angeboten. Es handelt sich hierbei um ROM-Cassetten, die in einem Schacht an der rechten Seite des Gerätes an den Expansionsstecker anschließen. Das Schwergewicht dieser Program Paks liegt bei Spielen. In dieser Form gibt es aber auch einen Editor/Assembler/Debugger und auch einen Wordprozessor.*

**4. Dokumentation**

*Der Color-Computer wird vor allem für den Erstanwender angeboten, der mit der Computerei beginnt. Aus diesem Blickwinkel ist die Dokumentation zu betrachten. Die hardwaremäßigen Anschlüsse und deren Testmöglichkeiten sind in einem Operation–Manual zusammengefaßt. Es beschreibt alle Anschlüsse, die Bedienung des Computers und eines Teiles der Peripherie. Die BASIC-Lehrbücher mit den Titeln 'Getting Started with Color-BASIC' und 'Going Ahead with Extended Color-BASIC' sind mit sehr viel Liebe didaktisch gut gemacht und enthalten für den Neuling eine Reihe von wertvollen Informationen über das Programmieren in BASIC allgemein und die Ausnutzung der Möglichkeiten des Color-Computers im besonderen. Der Kenner in BASIC kann sich allerdings mit der Reference-Card begnügen, die die Syntax der BASIC-Sprachelemente aufzeigt, wie sie im Color-Computer implementiert wurden. So gut wie die Darstellung des BASIC ist, so unzureichend sind die Informationen zum Programmieren in Maschinensprache. Leider liegt eine Beschreibung zum Editor/Assembler noch nicht vor, so daß auch nicht beurteilt werden kann, ob mit diesem Program Pak einiges in dieser Richtung getan wurde. Die Angaben in den BASIC Manuals sind nur sehr oberflächlich und teilweise sogar falsch. Die Fehler sind zum einen vom Setzer oder Korrektor zu verantworten, teilweise liegen aber auch richtige logische Fehler vor. Dies betrifft insbesondere die dokumentierten Einsprungadressen für Dienstleistungen des Betriebsprogrammes. Ein Teil dieser Startpunkte wurde etwas weltfremd gewählt und stellt für den Benutzer keine echte Hilfe dar, besonders wenn eine sehr viel sinnvollere Startstelle nur wenige Speicherstellen davor oder dahinter liegt. Insoweit kann die Dokumentation einen vom AIM verwöhnten Benutzer nicht befriedigen.*

*Zusätzlich zu dieser Literatur liegt ein Service-Manual vor, das in Grundzügen Reparatur- und Testmaßnahmen beschreibt. Ebenso sind die Schaltpläne in diesem Heft enthalten.*

**5. Der Color-Computer und seine Stellung am Markt**

*Mit einem Anfangspreis von 399 Dollar - bei freien Händlern herunter bis zu 300 Dollar - ist der Color-Computer vergleichbar dem VC20, dem C1P und dem Atari 400. Diesen Maschinen hat er die modernere CPU voraus, dem VC20 und C1P zusätzlich einen besseren Bildschirmaufbau und dem Atari eine solidere Tastatur. Mit Extended BASIC und 16 K RAM kostet der Color-Computer 599 Dollar. Er liegt damit unter Atari 800 und APPLE II, freilich auch ohne deren Möglichkeiten zu haben. Bei der weiten Verbreitung von Radio-Shack und einem Preis, der durchaus in die Landschaft paßt, steht einer weiten Verbreitung damit eigentlich nichts im Wege. In Deutschland sieht die Situation freilich etwas anders aus. Zum einen ist hier die Stellung von Radio-Shack, verglichen mit Commodore, bedeutend schwächer. Deswegen wird auch der VC20 relativ preiswert, der Color-Computer vergleichsweise teuer angeboten. Dieser Umstand und die verzögerte Einführung in Europa werden von Tandy mit der Umstellung auf die PAL-Norm erklärt.*

**6. Erweiterungsmöglichkeiten**

*Die Grundversion mit normalem Color-BASIC und 4 K RAM ist erweiterbar auf 16K und 32K. Ebenso ist Extended Color-BASIC nachrüstbar. An Druckern wird eine breite Palette von low cost*

*Matrixdruckern für unter DM 1000 bis hin zu leistungsfähigen Textverarbeitungs-Schreibwerken für mehrere tausend DM angeboten. Diese sind an die RS232-Schnittstelle anschließbar. An diese Schnittstelle kann man auch eine Reihe von Modems anschließen. Für eine Reihe von Informationsdiensten (Dow Jones, Compuserve) wird ein Video-Text Program Pak angeboten. Für spielerische Anwendungen sind Anschlüsse für zwei Joysticks vorgesehen.*

*Weiterhin ist ein Floppy-Disk-System anschließbar. Für dieses wird das Betriebssystem nebst Controller in einem Program Pak in den Erweiterungsschacht gesteckt und über ein Kabel mit den 5 1/4''-Laufwerken verbunden. Bis zu vier Laufwerke mit je 156K stehen dem Benutzer so zur Verfügung. - Langsam beginnen sich auch unabhängige Hersteller um Erweiterungen zu kümmern, so wird von einem kanadischen ein universeller Bustreiber angeboten, der ebenfalls in den Erweiterungsstecker im Schacht paßt und der die Signale auf einen 44-poligen Normstecker herausführt.*

**7. Erfahrungen mit dem Color-Computer**

*Beim Verfasser liegt mittlerweile eine Betriebserfahrung von mehreren Monaten vor. Das Gerät wurde bei einem Aufenthalt in den USA erworben und stellt den Stand dar, in dem es dort ausgeliefert wird. Ausschlaggebend für den Kauf war, daß mit dem Color-Computer eine Maschine mit der 6809 als Komplettgerät zu einem niedrigen Preis zur Verfügung stand. Inwieweit sich Änderungen zu in Deutschland vertriebenen Geräten ergeben, kann nicht gesagt werden. Auf die Möglichkeiten der Farbdarstellung auf dem Bildschirm und die Tonausgabe auf den Fernsehlautsprecher wurde beim Einsatz hier verzichtet. - Für den Betrieb in Deutschland war lediglich ein 220V/110V/0 9A-Trafo und ein selbstgelötetes Anpaßstück zwischen Cinch-Stecker und Fernsehantennenbuchse erforderlich. Nach einigem Probieren an der Einstellung des Fernsehers war in den Kanälen 3 und 4 ein gutes Bild zu erhalten. Der Fernseher ist ein preisgünstiger Import irgendwo aus Asien, wie man ihn für unter DM 200 überall erhält. Angesichts der einfachen Umstellung des Computers kann man in Anbetracht der Preisrelationen auch an einen Direktimport denken.*

*Auch im täglichen Betrieb macht das Gerät einen überaus guten Eindruck. Einige Kritik soll allerdings nicht unerwähnt bleiben: Die Tastatur ist einfach und taschenrechnermäßig, sie ist aber durchaus nach etwas Gewöhnung benutzbar. Möglicherweise bietet hier der eine oder andere Hersteller eine professionellere Lösung an, was leider durch die von der Norm abweichende Dekodierung erschwert wird. Manchem Benuzter mag die Länge der Bildschirmzeilen mit 32 Zeichen als zu kurz erscheinen, sie ist aber für Zwecke der Datenerfassung durchaus ausreichend, lediglich für Textverarbeitung sind mehr Zeichen angebracht. Viel wichtiger ist die Länge, die eine BASIC-Zeile haben kann; mit 256 Zeichen dürften hier keine Wünsche offen bleiben.*

*Als langjähriger Benutzer erst des KIM und dann des AIM hat man naturgemäß einige Erwartungen hinsichtlich der Transparenz des Betriebssystems und der Experimentiermöglichkeiten am System. Hier wird man allerdings etwas enttäuscht, sollte dies aber mit den Fähigkeiten, die man eben an den vorgenannten Systemen erworben hat, z.B. Programmeingabe in Hexa, als Herausforderung betrachten, diese Dinge auch hier zu implementieren. Der BASIC-Interpreter kommt einem hier mit dem HEX$-Befehl und der Möglichkeit zur Eingabe hexadezimaler Konstanten weitgehend entgegen. Große Freude bereitete die Entdeckung der Grafik-Fähigkeiten. Die Implementierung eines Programmes z.B. zur Analyse von Aktienkursen ist eine Kleinigkeit. Aber auch hier eine bittere Nuß: Entsprechend den Fähigkeiten der 6847 zeigt der Bildschirm Alpha-Zeichen oder Grafik an. Aber auch diese Beschränkung sollte als Herausforderung und nicht als unumstößliche Gegebenheit betrachtet werden.*

*Zusammenfassend ist der Verfasser mit den Gegebenheiten des Color-Computers durchaus zufrieden, die Nachteile werden bewußt in Kauf genommen, entweder kann man mit ihnen leben oder man sinnt auf Abhilfe. Vorteilhaft sind insbesondere die leistungsfähige CPU und die umfangreichen Möglichkeiten zur grafischen Darstellung.*

#

# Commodore VC-20

**1. Vorbemerkungen**

Seit mehreren Wochen wird der farbtüchtige VolksComputer von Commodore zu einem Endverbraucherpreis von etwa DM 900,- (inkl. 13%) angeboten. Er hat die CPU 6502 und ein leistungsfähiges BASIC (Microsoft), das dem Commodore-BASIC 3 entspricht. Er gehört damit zur Systemfamilie, besonders auch wegen seines attraktiven Preises.

In den USA VIC-20 genannt, wird er auch hier als Volks-Computer werblich herausgestellt und in einer ansprechenden Verpackung geliefert, die auch Kaufhauskunden zum Zugreifen veranlassen kann. Man findet in ihr den Computer in der Größe eines Keyboard-Terminals (405 mm breit, 80 hoch und 205 tief), einen Trafo im Gehäuse mit Netzanschlußschnur und 9V-Ausgangsschnur nebst Stecker für den Computer (allerdings ohne Ausschalter in der Schnur), einen TV-Hochfrequenzmodulator und die Verbindungsschüre mit Steckern von diesem zum Computer und zur Antennenbuchse eines TV-Gerätes. Dazu gehört ein 164-seitiges deutsches Handbuch. Nach wenigen Minuten ist man mit dieser steckerfertigen Ausrüstung ungefähr im UHF-Kanal 36 'im Bilde', s/w oder auch farbig, je nach TV-Gerät. Während der Herausgeber noch anderen Aufgaben nachgeht, hat der 18jährige Sohn schon das mit Farben und Geräuscheffekten versehene Spiel UFO gegen Tank mühelos zum Laufen gebracht. Einige Tage später schon experimentiert er mit der Erzeugung eigener grafischer Zeichen.

Ist es ein Volks- oder Spielkomputer? Ja, aber auch noch mehr, wie wir sehen werden. Er setzt voraus, daß es fast überall ein benutzbares TV-Gerät gibt und dessen Wert man normalerweise nicht in den Anschaffungspreis einkalkuliert. Man sollte aber keinen Computer ohne Datasette (etwa DM 200) für die Programm- und Datenspeicherung betreiben. Bei vielen PET- und CBM Betreibern ist diese auch schon vorhanden.

Die logischen und Ein- und Ausgabefähigkeiten des implementierten BASIC 3 gehen über vieles hinaus, was man sonst in dieser Preisklasse vorfindet. Eingeständnisse gibt es eigentlich nur bei der Bildschirmausgabe. Ein TV-Gerät hat von Haus aus nur eine Bandbreite von etwa 3,5 MHz. Das erzeugte Bild ist je nach Gerät ganz leicht 'griesich', eine Bildschirmzeile hat wegen der noch zuträglichen Bildschirmauflösung nur 22 Zeichen, 23 Zeilen können je Bildschirm dargestellt werden. Gleichwohl kann eine zusammenhängende BASIC-Eingabezeile bis zu ca. 88 Zeichen enthalten.

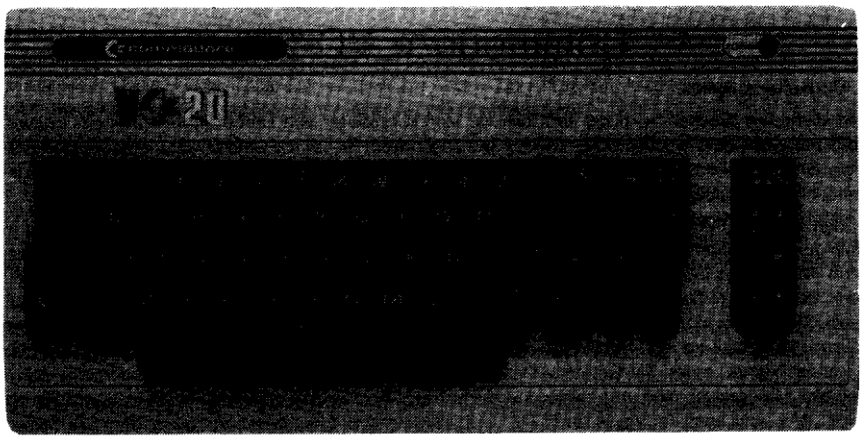
Wenn man will, hat der VC-20 ein gut ablesbares 'Bannerschrift-BASIC' ohne Einbußen an der Sprachimplementierung. Auch im gewerblichen Einsatz reicht das für die Übermittlung von Nachrichten zu entsprechend aufgestellten Bildschirmen aus. Auch in der Datenerfassung mit vorwiegend kurzen Zeilen mag das Bildformat in vielen Fällen ausreichen. Von der Datasette beschriebene Files können von den größeren Commodore-Systemen gelesen werden, so daß man sich vorstellen kann, daß der VC-20 als preiswertes Datenerfassungsgerät auftraggebender Firmen bei Hausfrauen aufgestellt wird. Dieser Computer und auch ähnliche werden die Anfänge eines 'distributed processing' aus dem privaten Heim erlauben.

What else? Der Computer erlaubt viele Spiele und gags. Der von Commodore ausgeschriebene und noch laufende Programmierwettbewerb wird da einiges zur persönlichen Unterhaltung zutage fördern und natürlich im Sinne des VolksComputers Anregungen für die Sicherung der Grundstücke und für das Haushalts- und Energiebudget geben. Vor allem aber werden wir ihn in die Hand unse-

rer heranwachsenden Kinder legen können. Jugendliche lernen mit ungeheuer fleißigen Experimenten die Computersprache und -bedienung fast von selbst.

**2. Die Hardware**

Unter der abgeschrägten Terminal-Tastatur verbirgt sich ein Single Board Computer mit 6502 CPU, 2 Interfacebausteinen VIA 6522, ein TV- und Geräuschchip 6561 nebst 2 ROMs zu je 8KB und elf statische RAMs 2114, von denen zehn als System-Memory oder freies RAM (3,5 KB) dienen




und das elfte den Farbauszug des TV-Bildes speichert (1Kx4). Es treten Treiberschaltkreise für die Peripherie hinzu. Die Hauptplatine hat keine Reservesteckplätze, führt aber die Interfaces und die Bussignale für eine mögliche Expansion auf Stecker heraus. Der Computer ist radiofreundlich, Störsignale in benachbarten Rundfunkgeräten konnten nicht bemerkt werden.

Die Tastatur darf als solide ausgelegte ASCII-Tastatur bezeichnet werden, fingerfreundlich zu bedienen und prellfrei. Dem Bediener stehen verschiedene Umschaltmöglichkeiten zur Verfügung. Es kann auf gemischte Groß- und Kleinschreibung umgeschaltet werden. Mit der Commodore-Taste stehen daneben als dritte Belegung vieler Tasten Grafiksymbole zur Verfügung. Wenn man das Gerät einschaltet, ist es im Grundzustand mit nur Großschreibung und 2 Grafiksätzen.

Mit den vorgesehenen Erweiterungen bietet der Computer eine RS232-Schnittstelle und auch eine für 20 mA TTY. Anschlußmöglichkeiten für Joysticks und für Lichtgriffel sind vorgesehen aber noch nicht erklärt. Es können nicht nur Farben, sondern auch Geräusche aus vier Tongeneratoren auf den TV-Kanal ausgegeben werden.

Der hardwaremäßige Aufbau ist konventionell. Wegen des Verzichtes auf die preiswerten dynamischen RAMs (stattdessen statische 2114) kommt man mit der einen Versorgungsspannung von 5 Volt aus. - Dem Anwender steht ein ungeteilter PORT B mit seinen verstärkten Treiberfähigkeiten zusammen mit den Kontroll-Leitungen CB1 und CB2 der VIA zur Verfügung, die unter dez. 37116 (hex 9110) dekodiert ist. Auf dieser VIA liegen auch in Port A die Joysticks, der Lichtgriffel, der Cassettenschalter und Serial Out ATN.

Wir kommen damit zu einer Abweichung gegenüber den PET- und CBM-Systemen von Commodore. Diese haben bekanntlich ein Interface mit Treiberbausteinen für den IEEE488-Bus. Beim VC-20 ist dieser Gerätebus nicht implementiert, wenn man auch Interfaces erwarten kann, die IEEE-angeschlossene Drucker und Floppies zu betreiben gestatten. In den USA werden sie schon in den Zeitschriften angeboten.

Das Interface des VC-20 zu seinem Gerätebus ist anders und noch nicht genug soft- und hardwaremäßig durchleuchtet. Man darf feststellen, daß es folgende Signalaustausche gibt: Serial Clock IN/OUT (Synchronisierung), ATN IN/OUT (Attention), Serial Data IN/OUT (Daten) und Serial SRQ IN (Bedienungsanforderung von der Peripherie). Es sind also im wesentlichen drei Leitungen, über die Datenaustausch und Handshake stattfinden. Anders im PET und CBM mit parallelem IEEE Transfer. Beobachtungen am Oszilloskop zeigen, das z.B. nach OPEN 8,8,1 Serial ATN auf low geht und daß Clock Out Signale gibt. Es wird offensichtlich auf eine Antwort lt. Protokoll gewartet. Diese Dinge dürften bald aufgeklärt sein und zu Interface-Vorschlägen führen.

Bei seinem sehr konventionellen Aufbau profitiert der VC-20 vom Interfacechip 6561 für die Bild- und Tonerzeugung. Ein Datenblatt dieses nicht für den allgemeinen Verkauf freigegebenen Bausteines liegt hier nicht vor. Er wird ab Adresse hex 9000 dekodiert und hat offensichtlich 16 Adressen, in deren ersten man für Bildschirmexperimente durchaus herumwühlen kann.

Der Userport wurde bereits erwähnt. Die Kontaktleiste für die Datasette ist pin-kompatibel zu der des PET/CBM, man kann sich gegenseitig lesen.

**3. Firmware**

Der VC-20 ist ein BASIC-Computer. Der Befehlssatz entspricht dem des leistungsfähigen BASIC 3, wie auch eine erste Disassemblierung zeigt. Die Belegung des Speichers mit den Systemvariablen und die Einsprungspunkte in die BASIC-ROMs sind grundverschieden von den früheren Versionen. Der andere Aufbau der Computerplatine bedingt auch ein anderes Ein- und Ausgabehandling. Es gibt keine besonderen grafischen oder akustische BASIC-Befehle. In der offenen Zeile bzw. in PRINT-Statements wird die Farbe der nachfolgenden Zeichen durch ein vorausgehendes CTRL

(1 ... 8) geschaltet. Alles andere geschieht mit dem Befehl POKE: andere Hintergrundfarbe des Bildes und der Umrandung, Farbveränderung an einer Schreibstelle, Ein- und Ausschalten der vier Tongeneratoren und der Lautstärke. Diese POKEs betreffen entweder den Interfacebaustein 6561 oder das 1Kx4 RAM, das den Farbauszug für die einzelnen Schreibstellen enthält. Man ist in der Lage, eigene Zeichen zu definieren, z.B. für die Sonderzeichen der deutschen Sprache. Angenehm für das Editieren ist das automatische REPEAT bei niedergehaltener Taste. Im übrigen gilt die schon früher verwirklichte Regel 'what you see is what you get' (Editieren der Bildschirmzeile).

BASIC-Programme werden ab Speicherzelle dez. 4096 (hex 1000) abgelegt. Es ist möglich, per Datasette Programme der PET- und CBM-Rechner auf den VC-20 zu übertragen, dabei werden BASICTexte, die bei den erstgenannten ja bei dez. 1024 (hex 400) beginnen, automatisch ab 4096 abgelegt. - Der VC-20 ist in der Lage, externe Files anzulegen und auf sie zuzugreifen. Dem dienen die Befehle OPEN (logisches File, Geräteadresse, Sekundäradresse), CLOSE, PRINT , INPUT 'und GET ..

**4. Dokumentation**

Zum Lieferumfang gehört das 164seitige deutsche 'VC-20 VolksComputer Handbuch', das grafisch sehr sorgfältig und liebevoll vor allem für den Anfänger aufgemacht ist. Es beschreibt die Inbetriebnahme, die Bedienung von Tastatur und Bildschirm, Farben und grafische Zeichen, Zeichentrickdarstellung, Musikerzeugung, und gibt Beispiele für die BASIC-Programmmierung.

Der Anhang dokumentiert die BASIC-Befehle und Fehlermeldungen, belegt den verwendeten Zeichencode, zeigt die Anschlüsse für E/A-Geräte und die Signale an den Anschlußleisten und -stekkern.. Der Anfänger ist damit sicher ausreichend bedient. Bei Commodore in Neu-Isenburg befindet sich weiterhin ein 'VC-20 Programmierhandbuch' in der Drucklegung, das sich sowohl an Anfänger wie an Fortgeschrittene wendet.

**5. Zusammenfassung**

Der VC-20 ist Kernstück eines künftig erweiterbaren Gerätesystems (Datasette, Floppy, Modulbox oder Motherboard zur Expansion, Matrixdrucker, Joysticks, Lichtgriffel, Modem usw.). Zur Grundausstattung sollte man mindestens eine Datasette hinzurechnen. Ein IEEE488-Interface ist zu empfehlen, wenn es schon vorhandene Peripherieeinheiten gibt. Er ist mit seiner Tastatur und bezüglich des Gehäuses bedienerfreundlich und solide aufgebaut..

Hinzu kommen ein leistungsfähiges BASIC und 3,5 KB freies RAM für Anwenderprogramme. Der Rechner ist nicht nur als Spielcomputer geeeignet, sondern auch als Lehrsystem für das Programmieren und zur Darstellung technischer und ähnlicher Abläufe im Unterricht. Daneben wird man bei vielen ernsthaften Anwendungen das beschränkte Bildschirmformat von 22x23 Zeichen i Kauf nehmen können. Man wird weiterhin den möglichen Einsatz für den Empfang von Bildschirmtext und in der Telekommunikation (mit Modem) zu beobachten haben.

# Was bietet 'Instant PASCAL' (AIM)?

*Schon vor der ab etwa Februar 1982 beginnenden Auslieferung der 5 PASCAL-ROMs für den Adreßbereich 4000-7FFF nebst B000-BFFF liegt das Anwenderhandbuch vor (Instant PASCAL user's manual, Rockwell Document No. 29650N85, ca. 94 Seiten), so daß hier schon eine Vorschau auf das gegeben werden kann, was den Benutzer erwartet. Auch eine deutsche Übersetzung dieses Büchleins wird bei Hans-Joachim Regge in Bremen lieferbar sein, wenn der Leser diese Zeilen studiert (siehe Hinweis an anderer Stelle im Heft).*

*Zum Betrieb der ROMs benötigt man eine 16K-PROM/ROM-Platine für die R2332 ROMs und den B-Steckplatz (Z26) auf der AIM-Platine. Als RAM reichen für kleinere Quellprogramme bis ca. 1800 Zeichen die 4K RAM auf dem AIM. Bei einem Ausbau des AIM kann maximal der Bereich von hex 300 bis 3FFF als Raum für das Programm benutzt werden. Beim erzeugten sehr dichten Code dürfte das auch für große Programme ausreichen, zumal die zur Strukturierung üblichen Spaces zur Einrückung von Programmzeilen keinen Speicherplatz 'verbraten'.*

*'Instant PASCAL' ist eine Schöpfung von Melvin E. Conway speziell für den AIM Es folgt auf weite Strecken dem Standard PASCAL nach Jensen und Wirth und dokumentiert im Handbuch seine Abweichungen davon. Wir gehen darauf noch ein.*

*Der bemerkenswerte Unterschied besteht in seinem Attribut 'instant': Die Eingabe von Text erfolgt im allgemeinen zeilenweise von der Tastatur her. Beim Betätigen der RETURN Taste wird die Eingabe bereits syntaktisch geprüft und in das gespeicherte Programm übernommen, wenn sie einwandfrei war. Im anderen Falle wird der Fehler gemeldet, und der Operateur hat die Möglichkeit, die beanstandete Zeile auf ein Neues einzugeben.*

*Befehlszeilen (Statements) werden in Form von Texteinheiten (text units) in das gespeicherte Programm übernommen, wobei Schlüsselwörter schon zu 'tokens' verdichtet werden. Die Überprüfung, ob die Folge der Befehlszeilen syntaktisch richtig ist und ob Variable und Prozeduren ausreichend definiert worden sind, erfolgt erst nach Hereinrufen des Binders mit dem Befehl 'G' (GO) oder 'C'. Auch die interne Zuweisung von Speicheradressen erfolgt erst mit dem Binden. Wegen der Verdichtung der Schlüsselwörter zu 'tokens' wird der eigentliche Quelltext als Wegwerfartikel bezeichnet (throw away). Ein Programm kann gleichwohl durch den 'Lister' wieder in Texteinheiten ausgelistet werden.*

*Dieses Konzept des 'Instant PASCAL' unterscheidet sich von anderen Implementierungen, bei denen der gesamte Quelltext über den Sprachkompiler geschickt wird, der die Prüfungen vornimmt und aus ihm im allgemeinen den zur Laufzeit interpretierbaren verdichteten 'P-Code' erzeugt. Mit seinen Texteinheiten erlaubt 'Instant PASCAL' auch eine Text-Editor-mäßige Bearbeitung des Programmtextes noch vor oder auch nach einer Programmausführung. Es gibt die Commands TOP und BOTTOM wie im Editor, LIST/xx Text Units, UP/xx, DOWN/xx und VIEW einer Umgebung, KILL/xx, INSERT und READ n lines, ferner FIND (String).*

*Ein Programm wird mit 'G' (GO) gebunden und zur Ausführung gebracht, vorher kann man mit dem alleinigen Hereinrufen des Binders ('C') schon eine weitere Absicherung des Programmes auf Richtigkeit vornehmen, so daß zur Laufzeit eigentlich nur noch 'runtime errors' auftreten sollten, z.B. Division durch Null oder unzulässige Zuweisungen an Variable.*

*Eine Programmausführung kann entweder mit GO ungebremst erfolgen oder im TRACE-Modus (Befehl ','). Im letzteren Fall gibt es das statement tracing (Anhalten an jeder Texteinheit) oder ein assignment tracing (Anhalten bei Wertzuweisung an Variable und bei FOR ...). Im Anschluß an eine Programmausführung oder einen TRACE-Schritt kann mit 'X' ein Direktkommando ausgeführt werden, z.B. WRITELN(VAR1, VAR2, ...). Damit sind von Haus aus gute Debug-Möglichkeiten gegeben.*

*Das Anwenderhandbuch behandelt des weiteren, was Texteinheiten sind, gibt syntaktische Hinwei-*

*se, Übersichten zum Befehlsvorrat und den Funktionen und erklärt wichtige Parameter in der Zero Page und die Fehlermeldungen. Für das Urteil des Lesers mögen folgende Hinweise in Hinsicht auf das Standard PASCAL nützlich sein:*

*Es werden die Variablen-Grundtyten BOOLEAN, INTEGER, REAL (Gleitkomma, 9 signifikante Stellen), CHAR (1 ASCII-Zeichen) und STRING unterstützt (Textkette, definierbar bis max. 255 Zeichen, dynamische Wertzuweisung mit 1 Byte Längenattribut). Einfachen und strukturierten Variablen können absolute hexadezimale Adressen zugewiesen werden, sie sind dann 'global'. Identifier (Label) können beliebige Länge haben und auch den Unterstreichungs-Strich (underline character) zur optischen Trennung der Wortbestandteile enthalten. Zum Anschluß an maschinensprachliche Routinen dienen die bereits erklärten Prozeduren SUBR und FUNC. Nicht implementiert sind file-Typen, Pointer-Typen, GET, PUT, RESET, REWRITE, PAGE und die Ende-Rrkenner EOLN und EOF (End of Line, of FILE). Ein- und Ausgabeeinheiten sind die des AIM 65. SETS haben im Grundtyp bis zu 8 Byte Elemente. Im Zusammenhang mit dem CASE-Statement ist für die nichtdefinierten Fälle das OTHERWISE implementiert. PACK und UNPACK stehen nicht zur Verfügung.*

*Wir werden so bald als möglich Beispiele für die Bedienung des 'Instant PASCAL' und weitere Hinweise auf seine Besonderheiten geben.*

R.L. #

*Ing. (grad.) Wolfgang Seer, 1000 Berlin 26*

# ISAM, ein Dateityp

*Das Akronym ISAM steht für Indexed Sequential Access Method und stellt neben den Dateitypen*

*SEQ (Sequential) und*
*REL (Relative)*

*einen weiteren Typ dar. ISAM ersetzt den Dateityp REL voll und ganz und geht in seinen Leistungsmerkmalen weit darüber hinaus.*

*Während der Dateityp REL in Microcomputern als Systemsoftware im MOS (Microcomputer Operatingsystem) bzw. DOS (Disk Operating System) vorhanden ist, kann man ISAM bisher nur als Softwareunterstützung in Form einer Assembler-Routine (Größe ca. 2 KB) kaufen. Beim Autor ist seit etwa einem Jahr die ISAM-Support der Fa. Creative Software, Mountain View, USA im Einsatz. Die Leistungsmerkmale des ISAM dieser Firma gelten als typisch und sind stellvertretend für ISAM-Routinen anderer Firmen.*

*Der Dateityp ISAM stellt eine konsequente Weiterentwicklung des Typs REL dar, so daß sich durch die Gegenüberstellung beider Typen die Leistungsmerkmale von ISAM gut erläutern lassen. Beide gehören zur Gruppe der Direktzugriffsdateien.*

**Die ISAM-Datei wächst dynamisch**

*Während die REL Datei von Haus aus eine Festgrößendatei ist, bei der im allgemeinen beim Initialisieren der erforderliche Speicherplatz (Datensatzgröße x Anzahl der Datensätze) reserviert wird, wächst die ISAM-Datei dynamisch durch Hinzufügen von Datensätzen. - Unterliegt eine Datei des Typs REL einer starken Fluktuation in ihren Datensätzen (Zu- und Abgang), so bleiben 'Löcher' in der Datei; und es bleibt dem Anwender überlassen, diese Löcher wieder zu stopfen. ISAM dagegen verwaltet den freigewordenen Speicherplatz selbst.*

**Zugriff auf einen Datensatz (DS) über den Key**

*Ein großer Nachteil des Dateityps REL ist, daß der Zugriff auf einen DS über die relative DS-Nr., bezogen auf den Dateianfang, erfolgt. Der Anwender wird also im Regelfall die DS-Nr. in einen Dateilogischen Schlüssel umsetzen. Sei es nun durch zusätzliches Programmhandling oder durch manuelle Aufzeichnungen.*

*Diesen Mangel behebt ISAM, indem es selbst Adreßtabellen (Indextabellen) anlegt und verwaltet. Beim Microcomputer ist die Tabellengröße im allgemeinen auf die physikalischen Gegebenheiten der Diskette abgestimmt (Blöcke zu 256 Bytes). Die Indextabellen werden den Datenblöcken, in denen sich die Datensätze befinden, vorangestellt. Sie enthalten, entsprechend ihrem Fassungsvermögen, eine Anzahl von Suchschlüsseln (Keys) mit der zugehörigen Blockadresse T,S (track, sector). Im weitesten Sinn kann die DS-Nr. beim Typ REL auch als Key betrachtet werden, nur kann bei ISAM der Key nach dateilogischen Gesichtspunkten vom Anwender frei definiert werden.*

*Innerhalb des DS verkörpert der Key das erste Datenfeld. Die Gesamtlänge eines DS wird daher oft definiert als: DS = Key + Record. Der Key muß alphanumerisch, also vom Datentyp 'String' sein. Der Wertebereich seiner Zeichen liegt zwischen hex 20 und hex 5F des ASCII-Codes. Aus Vereinfachungsgründen werden oft die Zeichen des Schlüssels von Kleinbuchstaben (lower case) in Großbuchstaben (upper case) umgesetzt. Die maximale Länge des Keys ist begrenzt und liegt im allgemeinen zwischen 16 und 32 Zeichen.*

*Der Key ist gleichzeitig Sortierkriterium für die ISAM-Verwaltung. Beim Hinzufügen eines DS wird der Key im Datenblock ggfs. auch in einem Indexblock an der Stelle eingefügt, die seinem Rang entspricht. Jede ISAM-Datei ist damit jederzeit alphabetisch geordnet.*

**Sequentielles Lesen**

*Was wäre eine ISAM-Datei, die zwar sortiert ist, aber ohne Kenntnis der Schüssel nicht gelesen werden kann? Die Sortierung wäre nicht nutzbar. Folglich besitzt jedes ISAM-System ein Feature, das sequentielles Lesen erlaubt. Bei einigen ISAM-Versionen besteht die möglichkeit, alternativ vorwärts (in aufsteigender Reihenfolge) oder rückwärts (in absteigender Reihenfolge) sequentiell zu lesen. Interessant ist für viele Anwendungen die mögliche Kombination:*

*Direktzugriff auf einen DS über Key und anschließend sequentielles Weiterlesen.*

*Nicht alle ISAM-Versionen bieten die Möglichkeit, bei unbekanntem Schlüssel nach einer Scheinpositionierung mit einem .Beinahe-Schlüssel' (Ergebnis: Key not found) sequentiell den nächsten Schlüssel (mit Datensatz) zu lesen.*

**ISAM-Befehle**

*Neben einigen versionsbedingten Befehlen kennt ISAM folgende Befehle:*

| | |
|---|---|
| *CREATE* | *initialisiert eine ISAM-Datei* |
| *OPEN* | *öffnet eine ISAM-Datei* |
| *WRITE* | *schreibt Schlüssel und Daten in eine ISAM-Datei* |
| *READ* | *liest Schlüssel und Daten aus einer ISAM-Datei* |
| *READNEXT* | *liest sequentiell Schlüssel und Daten aus einer ISAM-Datei* |
| *DELETE* | *löscht Schlüssel und Daten aus einer solchen* |
| *CLOSE* | *schließt eine ISAM-Datei* |

*Bei der ISAM-Assemblerroutine der Creative Software verbergen sich hinter den Befehlen Einsprungadressen, die von BASIC mit einem SYS-Befehl angesteuert werden. Meistens muß vor dem ISAM-Befehl der analoge BASIC-Befehl ausgeführt werden, in einigen Fällen auch erst danach. Die ISAM-Routine greift dann auf die BASIC-Operationspointer der Zero Page zu und kann so die zuletzt angesprochenen Variablen, Filenamen usw. übernehmen.*

**ISAM-Dateistatus und Fehlermeldungen**

*Zu den üblichen Meldungen des DOS kommen hinzu:*

| | |
|---|---|
| *ISAM-EOF* | *gibt ISAM-Dateiende an* |
| *KEY-NOT-FOUND* | *Schlüssel nicht gefunden* |
| *INVALID KEY* | *Schlüssel enthält unzulässige Zeichen* |
| *INVALID* | *bei Definition des ISAM-DS wurde die max.* |
| *KEY-LENGTH/* | *zulässige Länge des Schlüssels bzw* |
| *RECORD-LENGTH* | *des DS überschritten.* |

**Aufbau der Indextabelle**

*Die ISAM-Indextabelle besteht aus Blöcken zu 256 Bytes, die dynamisch nach einem modifizierten Prinzip des binären Suchbaums angelegt werden. Sie enthalten außer Blockpointer (T,S) den jeweils höchsten (letzten) Schlüssel (high key) des Blockes, auf den sie zeigen:*

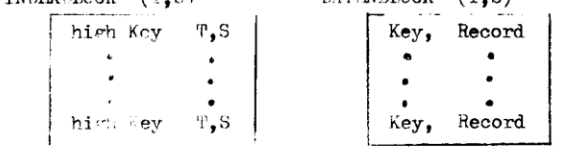



Abb. 1 Aufbau der Index- und Datenblöcke

*Der Block, auf den der Pointer zeigt, kann entweder ein weiterer nachgeordneter Indexblock oder ein Datenblock sein, in dem der gewünschte Datensatz sich befindet, wenn der Schlüssel in der Datei vorhanden ist. Um weitere Datenblöcke dynamisch einfügen zu können, ist jeder Datenblock über je einen Pointer mit dem logisch vorangehenden und nachfolgenden Datenblock verbunden. Beim Initialisieren einer ISAM-Datei wird der erste und zu diesem Zeitpunkt einzige Indexblock mit Zeichen hex 'FF' entsprechend der Länge des Keys als high key vorbelegt, der auf den zu diesem Zeitpunkt logisch ersten und leeren Datenblock zeigt.*

*Alle vom Anwender einzugebenden Schlüssel werden also logisch davor eingefügt. Das Suchen und Belegen freier Blöcke auf der Diskette überläßt ISAM dem DOS. Die Anzahl der Indexstufen und Indexblöcke hängt von der Länge der Keys und der Größe des Datensatzes ab.*

**MSAM**

*Vergleicht man ISAM unter den Datenhaltungssystemen mit einem 'Mercedes', dann ist MSAM der 'Rolls Royce' in der Datenhaltung. MSAM steht für Multi-indexed Sequential Access Method. Während ISAM nur einen Schlüssel je Datei erlaubt, sind bei MSAM mehrere Schlüssel möglich. Im Grenzfall kann jedes Datenfeld eines DS als Key deklariert werden. Leider ist MSAM in dieser Form als Systemsoftware auf dem Microcomputermarkt noch nicht erhältlich, so daß der Anwender das hierzu erforderliche Datenhaltungsprogramm unter Einbeziehung von ISAM selbst erstellen muß. Das ist in einer höheren Programmiersprache wie BASIC leichter als in Assembler. Erkauft wird dies dann um den Preis eines größeren Speicherplatzbedarfes und einer geringeren Verarbeitungsgeschwindigkeit.*

*Die praktische Obergrenze in der Anzahl der Keys ist die Zahl der gleichzeitig offenen Datenkanäle zur Floppy. Bei MSAM unterscheidet man zwischen dem Hauptschlüssel (Prime Key) und den Zweitschlüsseln (Secondary Keys). Der Grund liegt u.a. darin, daß jeder DS eine Kennung besitzen muß, die ihn innerhalb der Datei einmalig macht. Andererseits ist diese Eindeutigkeit bei Zweitschlüsseln nicht unbedingt erforderlich und manchmal auch gar nicht gegeben.*

**Quasi-MSAM**

*Eine Quasi-MSAM-Datei kann man in einfacher Weise dadurch erzeugen, daß man entsprechend der Anzahl der Keys ISAM-Dateien anlegt. Um die Dateinamen eindeutig zu machen und einander zuordnen zu können, empfiehlt es sich, den Key an den Dateinamen anzuhängen. Da man auch auf einer Diskette mit dem zur Verfügung stehenden Speicherplatz haushalten muß, haben die Dateien der Zweitschlüssel ein besonderes Aussehen. Ihr Datensatz besteht aus nur zwei Datenfeldern:*

1. *dem Key und*
2. *dem Prime Key als Pointer*

*auf den Datensatz der Hauptdatei, der allein die gesamte Information erhält. Jeder erfolgreiche Lesezugriff mit einem Zweitschlüssel auf eine ISAM-Datei zieht immer einen Zugriff auf die Hauptdatei (Prime Keyfile) nach sich.*

*Um den zusätzlichen Speicherplatzbedarf auf der Diskette so gering wie möglich zu halten, empfiehlt es sich, die Zweitschlüssel (die ja als Key deklariert und damit ebenfalls in der Indextabelle verwaltet werden) unabhängig von ihrer wahren Länge im Datenfeld der Hauptdatei auf z.B. max. 10 Zeichen zu begrenzen. Da aber durch diese Maßnahme die Eindeutigkeit der Keys im allgemeinen nicht mehr gewährleistet ist und weil andererseits jede ISAM-Datei einen eindeutigen Schlüssel verlangt und keine 'Duplicates' zuläßt, kann man sich so helfen: man hängt einen 'Qualifier' an jeden Zweitschlüssel. Dieser Qualifier kann z.B. durch einen vierstelligen Zähler dargestellt werden, der durch laufende Datensatzzugänge oder andere Maßnahmen hochgezählt wird.*

**Aufbau einer ISAM-Datei**

*Zur Demonstration soll eine ISAM-Datei mit 1000 Namen gebildet werden. Die Namen sollen über einen speziellen Datensatzschlüssel sowie auch über ihren Namen angesprochen werden. Dies bedingt einen Haupt- und einen Zweitschlüssel. Der Einfachheit halber soll der Zähler, der die 1000 Datensätze zählt, als Prime Key verwendet werden. In der Praxis wird man dies sicher nicht so machen. - Die Namen werden mit Hilfe der Random-Funktion des CBM-Rechners gebildet, um eine große Streubreite zu erreichen. Das Prime Keyfile möge 'TEST.SCH' heißen. Seine Datensätze liegen wegen der o.g. Art ihrer Erzeugung sortiert vor. Das Secondary Keyfile wird 'TEST.NAME' genannt. Seine Datensätze liegen extrem ungeordnet vor. - Für die so aufgebauten Dateien sind nachfolgend einige Auslistungen und Erkärungen gegeben.*

*Datenfeld "SCHL" = 5 Zeichen + CR / davon 4 als Qualifier*
*Datenfeld "Name" = 10 Zeichen + CR*

| File | Key | Record | Gesamt |
|---|---|---|---|
| ISAM-TEST.SCHL | 5 | CR+10+CR = 12 | 17 |
| ISAM-TEST.NAME | 10+4 | CR+5 +CR = 7 | 21 |

Abb. 2 Aufbau der Datensätze beider Dateien

```
spur: 18  sektor: 1

  0 ]   0 255 129  17   0 201 ] ▯↑▯▯I
  6 ] 211 193 205  45 212 197 ] SAM-TE
 12 ] 211 212 160 160 160 160 ] ST
 18 ] 160 160 160   0   0   0 ]    ▯▯▯
 24 ]   0   0   0   0  17   0 ] ▯▯▯▯▯▯
 30 ]   1   0   0   0 129  17 ] ▯▯▯▯▯▯
 36 ]   1 201 211 193 205  45 ] ▯ISAM-
 42 ] 212 197 211 212  32  46 ] TEST .
 48 ] 211 195 200 204 160   0 ] SCHL ▯
 54 ]   0   0   0   0   0   0 ] ▯▯▯▯▯▯
 60 ]   0   0   1   0   0   0 ] ▯▯▯▯▯▯
 66 ] 129  17   2 201 211 193 ] ▯▯▯ISA
 72 ] 205  45 212 197 211 212 ] M-TEST
 78 ]  32  46 206 193 205 197 ]  .NAME
 84 ] 160   0   0   0   0   0 ] ▯▯▯▯▯▯
 90 ]   0   0   0   0   1   0 ] ▯▯▯▯▯▯
```

Block 18-1

Dieser Block zeigt einen Ausschnitt aus dem Disketteninhaltsverzeichnis, das vom CBM-DOS für die ISAM-Dateien angelegt wurde.

Die erste Datei 'ISAM-TEST' stellt ein zusätzliches Descriptorfile für beide ISAM-Dateien dar und ist hier ohne Bedeutung. - Es fällt auf, daß beide ISAM-Dateien vom Typ SEQ sind (Kennung 129) und ihre Scheingröße für das DOS auf einen Block festgelegt wurde.

Für die weitere Untersuchung ist die Zweitschlüssel-Datei ISAM-TEST.NAME die interessantere von beiden. Anfangsblock dieser Datei ist 17-2.

spur: 17 sektor: 2

```
  0 ]   0 128 199   0   0  14 ]
  6 ]   7  67  87  75  82  77 ]  cwkrm
 12 ]  74  80  66  80  86  32 ]  jpbpv
 18 ]  51  48  53   6  18  69 ]  305
 24 ]  72  89  69  81  68  76 ]  hyeqdl
 30 ]  66  83  65  32  55  54 ]  bsa 76
 36 ]  53  13  14  70  87  74 ]  5
 42 ]  71  72  68  71  66  86 ]  ghdgbv
 48 ]  70  32  49  52  48   4 ]  f 140
 54 ]   0  73  65  84  88  82 ]  iatxr
 60 ]  82  72  70  71  76  32 ]  rhfgl
 66 ]  55  55  53  12  17  74 ]  775
 72 ]  84  72  84  85  70  67 ]  thtufc
 78 ]  72  77  75  32  50  52 ]  hmk 24
 84 ]  56   6   9  78  71  84 ]  8
 90 ]  80  74  83  84  75  72 ]  pjstkh

 96 ]  70  32  32  32  50   2 ]  f   2
102 ]   9  81  71  69  74  76 ]
108 ]  70  84  65  88  84  32 ]  ftaxt
114 ]  32  49  48   7   0  83 ]   10
120 ]  70  90  78  81  89  66 ]  fznqyb
126 ]  85  82  72  32  56  51 ]  urh 83
132 ]  52  12   5  84  89  69 ]  4
138 ]  65  77  76  86  70  77 ]  amlvfm
144 ]  72  32  50  48  51   4 ]  h 203
150 ]   2  86  77  76  67  71 ]  vmlcg
156 ]  79  84  71  82  83  32 ]  otgrs
162 ]  55  54  55  13   3  88 ]  767x
168 ]  66  85  83  69  82  85 ]  buseru
174 ]  66  69  79  32  51  53 ]  beo 35
180 ]  48   8  15 255 255 255 ]  0
186 ] 255 255 255 255 255 255 ]  ↑↑↑↑↑↑
```

spur: 6 sektor: 18

```
  0 ]   0 128 247   0   0  14 ]
  6 ]   7  65  72  90  82  88 ]  ahzrx

 12 ]  83  83  76  88  79  32 ]  sslxo
 18 ]  32  52  49  10   2  65 ]   41a
 24 ]  78  85  76  70  90  72 ]  nulfzh
 30 ]  67  71  77  32  32  54 ]  cgm  6
 36 ]  52   4  17  65  83  77 ]  4asm
 42 ]  89  81  89  89  67  73 ]  yqyyci
 48 ]  80  32  50  57  53  12 ]  p 295
 54 ]  12  65  84  76  82  73 ]  latlri
 60 ]  71  90  88  84  86  32 ]  gzxtv
 66 ]  32  57  50   2   2  65 ]   92a
 72 ]  90  69  73  87  66  67 ]  zeiwbc
 78 ]  77  80  72  32  49  54 ]  mph 16
 84 ]  51   5   5  66  70  80 ]  3
 90 ]  69  75  84  82  66  79 ]  ektrbo

 96 ]  67  32  51  54  54   6 ]  c 366
102 ]  17  66  76  75  67  71 ]  blkcg
108 ]  88  69  67  67  90  32 ]  xeccz
114 ]  52  50  54  11   3  66 ]  426b
120 ]  80  85  67  90  70  90 ]  puczfz
```

Block 17-2

Dieser Block stellt die Indexwurzel des binären Suchbaumes dar. Typisches Merkmal ist der höchste Key mit dez. 255 ... dez. 255 ab Byte Nr. 183. Alle Pointer (T,S) der Keys zeigen auf Blöcke, die Indexblöcke darstellen. So ist der erste Block der nächsten Ebene Block Nr. 6-18.

Der höchste in diesem Block eingetragene Schlüssel ist: cwkrmjpbpv305. Wegen der oben genannten Verknüpfung erkennt man aus dem Qualifier den 305. Datensatz.

Block 6-18

Um zum logisch ersten Datenblock und damit zum logisch ersten Namen zu gelangen, muß man sich den ersten high key dieses Blockes ansehen. Er zeigt mit dem Schlüssel 'ahzrxsslxo 41' auf den Datenblock Nr. 10-2.

```
126 ]  65  76  71  32  49  54 ] als 16
132 ]  56   3   9  66  84  68 ] 8▒▒otd
138 ]  79  71  75  67  77  70 ] oskcmf
144 ]  87  32  51  51  56  10 ] w 338▒
150 ]  14  66  88  80  83  73 ] ▒bxpsi
156 ]  90  81  90  67  88  32 ] zqzcx
162 ]  32  56  55   5   3  67 ]  87▒▒c
168 ]  67  66  86  84  82  74 ] cbvtrj
174 ]  80  87  75  32  50  48 ] pwk 20
180 ]  54   7  19  67  74  67 ] 6▒▒cjc
186 ]  79  73  88  74  68  75 ] oixjdk

192 ]  67  32  32  32  55   1 ] c   7▒
198 ]  12  67  81  72  83  78 ] ▒cqhsn
204 ]  79  66  80  70  75  32 ] obpfk
210 ]  53  50  57   8  10  67 ] 529▒▒c
216 ]  85  67  82  86  78  87 ] ucrvnw
222 ]  74  67  89  32  56  51 ] jcy 83
228 ]  49  11  13  67  87  75 ] 1▒▒cwk
234 ]  82  77  74  80  66  80 ] rmjpbp
240 ]  86  32  51  48  53   6 ] v 305▒
246 ]   2   0   0   0   0   0 ] ▒▒▒▒▒▒
252 ]   0   0   0   0       ] ▒▒▒▒
spur: 10  sektor: 2

  0 ]   0   0 175   0   0   4 ] ▒▒_▒▒▒
  6 ]  17  65  65  76  68  70 ] ▒aaldf
 12 ]  65  74  78  71  77  32 ] ajnqm
 18 ]  56  48  57  13  32  32 ] 809▒
 24 ]  56  48  57  13  65  66 ] 809▒ab
 30 ]  86  81  69  76  71  83 ] vqelgs
 36 ]  83  66  32  49  56  53 ] sb 185
 42 ]  13  32  32  49  56  53 ] ▒  185
 48 ]  13  65  69  84  67  87 ] ▒aetcw
 54 ]  87  87  88  77  65  32 ] wwxma
 60 ]  56  57  56  13  32  32 ] 898▒
 66 ]  56  57  56  13  65  70 ] 898▒af
 72 ]  70  65  69  86  88  67 ] faevxc
 78 ]  66  89  32  51  52  52 ] by 344
 84 ]  13  32  32  51  52  52 ] ▒  344
 90 ]  13  65  71  75  82  80 ] ▒agkrp
 96 ]  65  81  69  68  71  32 ] aqedg
'02 ]  54  53  54  13  32  32 ] 656▒
 08 ]  54  53  54  13  65  71 ] 656▒ag
114 ]  86  90  80  80  75  72 ] vzppkh
120 ]  83  86  32  52  55  54 ] sv 476
126 ]  13  32  32  52  55  54 ] ▒  476
132 ]  13  65  72  87  87  82 ] ▒ahwwr
138 ]  86  79  74  84  70  32 ] vojtf
144 ]  32  49  56  13  32  32 ]  18▒
150 ]  32  49  56  13  65  72 ]  18▒ah
156 ]  90  82  88  83  83  76 ] zrxssl
162 ]  88  79  32  32  52  49 ] xo  41
168 ]  13  32  32  32  52  49 ] ▒  41
174 ]  13  65  76  78  78  80 ] ▒alnnp
180 ]  70  72  87  78  70  32 ] fhwnf
186 ]  49  52  49  13  32  32 ] 141▒
```

Block 10-2

Dieses ist der erste Datenblock der ISAM-Datei ISAM-TEST.NAME. Er enthält 175 gültige ISAM-Bytes, besitzt keinen vorangehenden Datenblock (0,0) und als Nachfolger den Block 4-17.

Der erste Schlüssel lautet: 'aaldfajngm 809'. Die einzelnen Datenfelder sind durch Carriage-Return-Zeichen voneinander getrennt. Man erkennt weiterhin, was auch demonstriert werden sollte, daß das zweite Datenfeld den Prime Key ' 809' als Pointer für den Zugriff auf die Hauptdatei enthält, denn nur sie enthalält in der Praxis den gesamten Datensatz.

Literaturhinweise

ISAM Referende Manual, Creative Software, Mountain View, USA

CBM 4040 Bedienungshandbuch, Commodore GmbH, Neu-Isenburg.

*Bernhard Kokula, 6700 Ludwigshafen*

# Prozeßtechnik (4)

*Multiplex-Ausgabe von internen Meßwerten aus 7-Segmentanzeigen*

*Vorbemerkung: Die Abfolge in dieser Artikelserie ist geändert worden, um noch Gelegenheit zur möglichen Vereinheitlichung aller Regler zu geben.*

*Das Modul ANZEIG dient der Ausgabe von einem 3-stelligen Meßwert aus einer Menge von 4 Meßwerten in dezimaler oder hexadezimaler Darstellung. Hierfür werden nur 8 Bit eines Ausgabeports benötigt. Grundlage des Moduls sind CMOS-7-Segment-Treiber, die bei Preisen um DM 1,60 ein 4-Bit-Datenlatch, einen BCD zu 7 Segment Dekoder und die 7-Segment-Treiber enthalten. Es sind dies der CD4511 (geeignet für BCD) und der MC14495 (BCN/7-Segment Dekoder, einschl. Widerständen von je 290 Ohm auf der Ausgangsseite). Der 4511 stellt dar: 0,1,2,3,4,5,6,7,8,9,BLANK. Der 14495 ist nicht pingleich aber bringt BCN: 0,1,2,3,4,5,6,7,8,9,A,b,C,d,E,F. Sie bieten weiterhin z.T. Eingänge für Lampentest, Dunkelsteuerung, Dezimalpunkt.*

*Die Ausgabe der Daten an die Anzeigen erfolgt sequentiell. Ausgewählt werden die Lampen durch einen kurzen LOW-Puls am LE-Eingang. Ein drittes Bit sowie die andere Hälfte des Stellen-Decoders CD4556 (+ 1 Inverter) ließe auch einen aus 7 Meßwerten errreichen. Bei 4 Bit und Benutzung des CD4515 steigt die Zahl der anwählbaren Meßwerte auf 15!*

*Die 7-Segment-Anzeigen, zusammengebaut mit den Decodern kann man bei verschiedenen Firmen fertig kaufen, z.B. bei Contraves, München (Typ ZMB/702/S), Cherry, 8572 Auerbach Opf. (T, baugleich Siebert) und Fa. Siebert, 6613 Eppelborn, hier: D 65-01 für BCD0-9 oder -03 für BCN für Darstellung von 0-F.*

*Das Programm-Modul ANZEIG ist für den USER-PORT 6522 im AIM 65 geschrieben. Er besteht aus den Teilen*

| | |
|---|---|
| *INIT* | *Port initialisieren* |
| *MESNR* | *Meßwert-Nummer erzeugen* |
| *ANZEIG* | *Meßwert ausgeben* |

*INIT bereitet den Port vor: Bit 0-5 als OUTPUT, Bit 6 und 7 als INPUT und legt Nullen auf die Ausgabe. MESNR schafft bit 6 und 7 nach ANZNR-Bit 0+1 zur Bedienung von ANZEIG.*

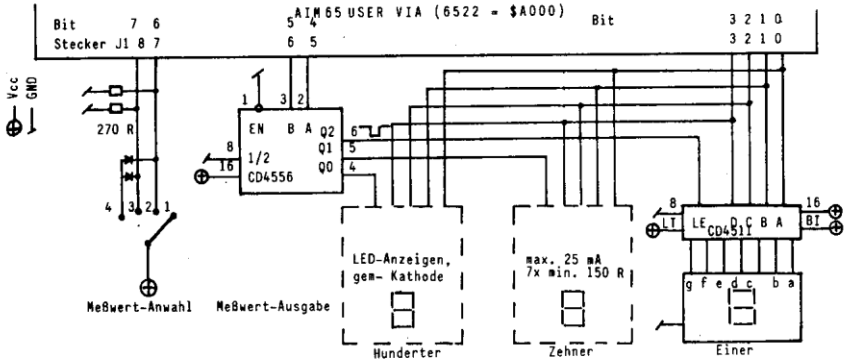
*ANZEIG gibt den mit MESNR von außen angewählten Meßwert an die Anzeigen aus. Die Ausgabe erfolgt in einer Schleife in der Reihenfolge Hunderter, Zehner, Einer, die zuvor im Stack 'versteckt' worden sind (daher die Bezeichnung 'Stack', Anm. d. Hrsg.). Der Zahlenbereich der Ausgabe umfaßt dezimal 000 bis 999 (realistisch bis 255 entspr. hex FF). Durch eine andere Vorgabe in Y können beliebig viele Stellen ausgegeben werden. Zur Ausgabe genügt ein einmaliger Aufruf von ANZEIG, da die Daten ja im Latch der 7-Segment-Treiber gespeichert bleiben.*

*Bei Ausgabe von Daten größer '9' (z.B. 'F') wird die Anzeige per Software BLANK, das heißt, sie wird dunkel. Das gilt für den Treiber CD4511. Durch zyklischen Wechsel zwischen Date und 'F' kann die Anzeige somit blinken. - Der Aufruf von ANZEIG kann aber auch vom Programm zur Signalisierung einer Störung veranlaßt werden, das Programm muß dann nur die Meßnummer vorgeben.*

*Dieser Programm-Modul verträgt sich mit den schon beschreibenen MUX555 usw., ist aber unabhängig und ohne andere sofort lauffähig. Allerdings müssen zuvor die 'DEZREG' geladen werden. Das kann mit dem Monitor-Kommando 'M' hexadezimal geschehen. Man kann ANZEIG auch einen laufenden Dezimalzähler ausgeben lassen oder das Modul HEXDEZ aus Heft 22 mit dem Teil 2 dieser Serie vorschalten.*

*Start dieser Version erfolgt bei START = hex 248.*

| | | |
|---|---|---|
| *Die DEZREG liegen:* | *LOBYTE bei ZYKLUS=8 von hex* | *14 bis 1B* |
| | *HIBYTE bei "* | *1C bis 23* |
| *Das bedeutet:* | *Die Meßstelle '00' liegt* | *1C HIBYTE* |
| | | *14 LOBYTE* |




```
PROZESSTECHNIK MODUL /ANZEIGE/

0000                    0002 ;.                                   01.01.82/KOKUL
0000                    0003 ;.
0000                    0004 ;.FUER 3 DEKADEN UND 4 MESSWERTE AUS 1 PORT (8 BIT)
0000                    0005 ;.
0000                    0006 ;.
0000                    0007 ;.+++++++++ VARIABLEN AUS FRUEHEREN MODULEN
0000                    0008 ;.
0000                    0009 ZYKLUS =8                  ;MESSSTELLENZAHL, 1-8 WAEHLBAR
0000                    0010 VAR2   =$14                ;AUS TEIL 2, HEX/DEZ-WANDLER
0000                    0011 ;.
0000                    0012        *=VAR2
0014                    0013 DEZREG *=*+ZYKLUS+ZYKLUS
0024                    0014 ;.
0024                    0015 ;.
0024                    0016 ;.+++++++++ MODUL /ANZEIGE/
0024                    0017 ;.
0024                    0018 ;.---- PARAMETER ZU MODUL /ANZEIGE/
0024                    0019 ;.
0024                    0020 ANZNR  *=*+1               ;MESSELLEN-NR.
0025                    0021 TEMP   *=*+1               ;ZWISCHEN SPEICHER
0026                    0022 ;.
0026                    0023        *=$0200             ;*** PROGRAMMSTART
0200                    0024 ;.
0200                    0025 ;.---- PROGRAMM
0200                    0026 ;.
0200                    0027 ;.---- INIT AIM 65-USER PORT.A FUER ANZEIGE
0200  A9 3F            0028 INIT   LDA #%00111111
0202  8D 03 A0         0029        STA UDDRA           ;PORT B DIREKTION
0205  A9 30            0030        LDA #%00110000
0207  8D 01 A0         0031        STA UDRA            ;CLEAR PORT
020A  60                0032        RTS
020B                    0034 ;---- HOLE MESSTELLEN NUMMER VOM PORT, BIT 6+7
020B  A9 00            0035 MESNR  LDA #0
020D  85 24            0036        STA ANZNR           ;CLEAR
020F  AD 01 A0         0037        LDA UDRA
0212  0A                0038        ASL A               ;BIT 7 IN (C)
```

```
0213  26 24     0039          ROL ANZNR          ;(C) IN BIT 0
0215  0A        0040          ASL A              ;NEXT
0216  26 24     0041          ROL ANZNR
0218  60        0042          RTS
0219            0043 ;.
0219            0044 ;---- GIB DEN MESSWERT AN DIE DISPLAYS AUS
0219  A6 24     0045 ANZEIG LDX ANZNR          ;HOLE MESSTELLEN-NR.
021B  B5 14     0046          LDA DEZREG,X       ;EINER & ZEHNER
021D  48        0047          PHA                ;SAVE EINER
021E  4A        0048          LSR A
021F  4A        0049          LSR A
0220  4A        0050          LSR A
0221  4A        0051          LSR A              ;4X SHIFT RIGHT
0222  48        0052          PHA                ;SAVE ZEHNER
0223  B5 1C     0053          LDA DEZREG+ZYKLUS,X
0225  48        0054          PHA                ; SAVE HUNDERTER
0226            0055 ;---- AUSGABE SCHLEIFE (HUNDERTER/ZEHNER/EINER)
0226  A0 02     0056          LDY #2             ;3 MAL RUM
0228  68        0057 ME1      PLA                ;HOLE DATE
0229  29 0F     0058          AND #%00001111
022B  85 25     0059          STA TEMP
022D  AD 01 A0  0060          LDA UDRA           ;HOLE PORT
0230  39 42 02  0061          AND LOPULS,Y       ;SETZE CLOCK & CLEAR DATA
0233  05 25     0062          ORA TEMP           ;LADE MESSWERT
0235  8D 01 A0  0063          STA UDRA           ;GIB AUS
0238  19 45 02  0064          ORA HIPULS,Y       ;ENDE VON CLOCK
023B  8D 01 A0  0065          STA UDRA
023E  88        0066          DEY
023F  10 E7     0067          BPL ME1
0241  60        0068          RTS
0242            0069 ;.
0242            0070 ;.----- KONSTANTEN, EPROM-FAEHIG
0242            0071 ;.       SETZE DISPLAY-CLOCK & CLEAR DATA
0242  C0        0072 LOPULS .BYT %11100000,%11010000,%11000000
0243  D0        0072
0244  C0        0072
0245            0073 ;.       RESET DISPLAY-CLOCK
0245  10        0074 HIPULS .BYT %00010000,%00100000,%00110000
0246  20        0074
0247  30        0074
0248            0075 PROG4    =*
0248            0076 ;.
0248            0077 ;.----- AIM-ADRESSEN ZUM MODUL /ANZEIGE/
0248            0078          *=$A000            ;USER-VIA 6522
A000            0079 UDRA     =*+1               ;DATA REG A
A000            0080 UDDRA    =*+3               ;DIR  REG A
A000            0083 ;.+++++++++ TESTPROGRAMM
A000            0084 ;.
A000            0085          *=PROG4
0248            0086 START    =*
0248  20 00 02  0087          JSR INIT           ;INITIALISIERE DEN PORT
024B  20 0B 02  0088 LOOP     JSR MESNR          ;HOLE MESSTELLEN NUMMER
024E  20 19 02  0089          JSR ANZEIG         ;GIB MESSWERT AUS
0251  20 07 E9  0090          JSR RCHEK          ;HALT/STOP WANTED?
0254  4C 4B 02  0091          JMP LOOP
0257            0092 ;.
0257            0093 ;.----- AIM-ADRESSEN NUR ZUM TESTEN
0257            0094 RCHEK    =$E907             ;STOP MIT 'ESC'
```

Ingenieurbüro Stecker, Dipl.-Math Peter Mees

# Lösung der kubischen Gleichung

Ein kleines 'Abfallprodukt' aus einer größeren Arbeit ist das nachstehende BASIC-Programm zur Berechnung der reellen und komplexen Lösungen der kubischen Gleichung in der Form

$$a*x^3 + b*x^2 + c*x + d = 0$$

mit den Koeffizienten a, b, c, d, $a \neq 0$.

Das Programm liefert daneben ein anschauliches Beispiel dafür, daß schon minimale Rundungsfehler ein Programm 'sterben' lassen, wenn diese in der Analyse nicht beachtet werden. Man ändere nur einmal in der Zeile 310 das 1.E-8 zu 0 und streiche Zeile 320, wie es vom Algorithmus her richtig wäre und teste am Beispiel a=1, b=1, c=0, d=0.....

Das Programm und die zugehörige Dokumentation können auf Diskette (für den CBM 4040) oder auf Bandcassette (für den AIM 65) für DM 15,00 bezogen werden von: Ingenieurbüro Stecker, Delmenhorster Str. 20, 5000 Köln 60.

```
100 OPEN1,0:OPEN2,4
110 PRINTCHR$(147)
120 PRINT"PROGRAMM    KUBISCHE GLEICHUNG":PRINT
130 U$="------------------------------------------------------------------------"
140 U1$="---------------------------------------"
150 U2$="-----------------------------------------------------"
160 PRINT"DATUM: ";:INPUT§1,DA$:PRINT
170 PRINT§2,"DATUM: ";DA$
180 PRINT:PRINT
190 PRINT"KUBISCHE GLEICHUNG:":PRINT:PRINT" A*X|3 + B*X|2 + C*X + D  =  0"
200 PRINT:PRINT"EINGABE DER KOEFFIZIENTEN( A²³0 ): ":PRINT
210 PRINT"A = ";:INPUT§1,A:PRINT:IFA=0 GOTO210
220 PRINT"B = ";:INPUT§1,B:PRINT
230 PRINT"C = ";:INPUT§1,C:PRINT
240 PRINT"D = ";:INPUT§1,D:PRINT
250 PRINT§2:PRINT§2,"KUBISCHE GLEICHUNG:":PRINT§2
260 PRINT§2,STR$(A)+"*X|3+("+STR$(B)+")*X|2 +("+STR$(C)+")*X+("+STR$(D)+") = 0"
270 P=(-B*B/(3*A*A)+C/A)/3
280 Q=(2*B|3/(27*A|3) - B*C/(3*A*A) + D/A)/2
290 BA=B/(3*A)
300 PQ=Q*Q+P*P*P
310 IF PQ³=1.E-8 THEN460
320 IF ABS(PQ)²1.E-8 THEN PQ=0:GOTO460
330 PRINT§2
340 PRINT§2,"GLEICHUNG BESITZT DREI VERSCHIEDENE REELLE LÖSUNGEN:"
350 PRINT§2,U2$:PRINT§2
360 X=-Q/SQR(-P|3)
370 PH=-ATN(X/SQR(-X*X+1))+°/2
380 PH=PH/3
390 X1= 2*SQR(-P)*COS(PH) - BA
400 X2=-2*SQR(-P)*COS(PH+°/3) - BA
410 X3=-2*SQR(-P)*COS(PH-°/3) - BA
420 PRINT§2,"X1 = ";X1
430 PRINT§2,"X2 = ";X2
440 PRINT§2,"X3 = ";X3
450 GOTO760
460 V1=-Q-SQR(PQ)
470 U1=-Q+SQR(PQ)
480 IF V1²0 THEN510
```

Legende
Auslistung erfolgte
auf Olivetti ET 221
mit Interface 8103
der Fa. Hard + Soft
Microcomputer GmbH

| | |
|---|---|
| § | entspr. # |
| \| | ↑ |
| ° | π |
| ² | < |
| ³ | > |

```
490 V=V1|(1/3)
500 GOTO520
510 V=-(-V1)|(1/3)
520 IF U1²0 THEN550
530 U=U1|(1/3)
540 GOTO560
550 U=-(-U1)|(1/3)
560 Y1=U+V
570 X1= Y1-BA
580 XR=-Y1/2-BA
590 X1=SQR(3)*(U-V)/2
600 IF PQ³OTHEN690
610 PRINT§2
620 PRINT§2,"GLEICHUNG BESITZT DREI REELLE LÖSUNGEN,"
630 PRINT§2,"VON DENEN MINDESTENS ZWEI GLEICH SIND:"
640 PRINT§2,U1$:PRINT§2
650 PRINT§2,"X1 = ";X1
660 PRINT§2,"X2 = ";XR
670 PRINT§2,"X3 = ";XR
680 GOTO760
690 PRINT§2
700 PRINT§2,"GLEICHUNG BESITZT EINE REELLE UND ZWEI KONJUGIERT-KOMPLEXE ";
710 PRINT§2,"LÖSUNGEN:":PRINT§2,U$
720 PRINT§2
730 PRINT§2,"X1 = ";X1
740 PRINT§2,"X2 = ";XR;" + I*";XI
750 PRINT§2,"X3 = ";XR;" - I*";X1
760 PRINT:PRINT:PRINT"NOCHMALIGE BERECHNUNG ?"
770 PRINT"ANTWORT(J/N): ";:INPUT§1,A$:PRINT
780 IF A$="J"THEN PRINTCHR$(147):PRINT§2:PRINT§2:GOTO190
790 IF A$<>"N" GOTO770
800 PRINT"ENDE DER BERECHNUNG"
810 CLOSE1:CLOSE2
READY.
```

# Formatierte Zahlenausgabe

**in technisch-wissenschaftlicher Notation**

Bei Tabellen, die mit den normalen BASIC-Befehlen ausgedruckt worden sind, stören sich Auge und Verstand oft an den verspringenden Zahlenformaten. Im Wertebereich von 0,01 bis etwas unter 1 Mrd (999999999.2) wird im Microsoft-BASIC mit PRINT ein Zahlenformat erzeugt, das unserer täglichen Schreibweise entspricht, z.B. also 12345,6789 oder .0123456. Beim Überschreiten der angegebenen Grenzen erfolgt automatisch eine Darstellung im Exponentialformat. Auch wenn man sich um die gleichmäßige Ausrichtung des Dezimalkommas in der Ausgaberoutine bemüht, so bleibt doch noch immer das störende Verspringen zwischen der normalen Darstellung der Zahlen als gemischte Dezimalzahl (ganzzahliger Teil, Dezimalpunkt, Dezimalbruch) und der exponentiellen.

Das nachfolgende BASIC-Programm erzeugt eine einheitliche Zahlenausgabe in technisch-wissenschaftlicher Notierung. Seine Dienstleistung liegt in den Zeilen 20 bis 500, die übrigen Zeilen dienen nur der Demonstration. Zur deutlicheren Lesbarkeit wird zwischen Mantisse und Exponent mindestens ein Leerraum gelassen. Auf eine mögliche Raffung des Programmes wurde bewußt verzichtet, um dem Leser den Nachvollzug zu erleichtern. - In das Listing wurde ein Probeabzug für verschiedene Zahlen eingeblendet.

Das BASIC-Programm benutzt in erheblichem Umfang Stringmanipulationen, die sich abkürzen lassen, wenn eine INSTRING-Funktion zur Verfügung steht. Mindestens für CBM-Rechner ist der Hinweis zu geben, daß der Befehl A=STR$(A) die Stringdarstellung der Zahl zunächst ab Adresse hex 0100 erzeugt, die Textkette wird dabei mit einem '00' abgeschlossen. Mit einigem Fleiß wäre es daher auch möglich, die dort vorhandene Textkette entsprechend der beabsichtigten einheitlichen Zahlendarstellung zu manipulieren und sie dann von dort einer Stringvariablen zuzuweisen oder direkt auszugeben.

```
5 REM FORTATIERTE TECHN.-WISS. ZAHLENAUSGABE
6 REM COPYRIGHT 1982 BY ROLAND LOEHR
10 INPUT"ZAHL";A:REM NUR ZUR  DEMO
20 A$=STR$(A)      :REM STRINGUMWANDLUNG
30 L=LEN(A$)
40 IFL=15THEN200 :REM  WAR SCHON FLOAT
50 IFL<3THENB$="":GOTO90: REM KLEINE GANZZAHL
60 IFINT(A)<>ATHEN300:REM BRUCHZAHL
61 :
63 REM PRUEFUNG AUF VORHANDENE FLOAT-DARSTELLUNG
65 IFL>3THENIFMID$(A$,L-3,1)="E"THEN200
69 :
70 REM GANZZAHL
80 B$=RIGHT$(A$,L-2):A$=LEFT$(A$,2)
90 B=LEN(B$)
100 A$=A$+"."+B$:L=L+1
110 IFL>=12THEN150
120 FORL=LTO12
130 A$=A$+" "
140 NEXT
150 A$=A$+"E "+RIGHT$(STR$(B),2)
160 GOTO500
170 :
200 B$=RIGHT$(A$,4):L=L-4:A$=LEFT$(A$,L)
210 IFL>12THEN250
220 FORL=LTO12:A$=A$+" ":NEXT
250 A$=A$+B$:GOTO500
300 :
305 REM GEMISCHTER BRUCH
310 IFA<1THEN400
320 B$=RIGHT$(A$,L-2)
330 A$=LEFT$(A$,2)+"."
340 B=0:A=LEN(B$)
350 FORI=1TOA
360 C$=MID$(B$,I,1):IFC$="."THENB=I-1:GOTO380
370 A$=A$+C$
380 NEXT
390 GOTO110
400 :
410 REM REINE DEZIMALBRUECHE
412 IFA<.01THEN200:REM IST FLOAT
415 REM VERBLEIBEN DEZIMALBRUECHE VON .01 BIS .99
420 B$=RIGHT$(A$,L-2)
430 B=-1
435 A$=" "
440 IFLEFT$(B$,1)="0"THENB$=RIGHT$(B$,L-3):B=B-1:L=L-1
450 A$=A$+LEFT$(B$,1)+".":B$=RIGHT$(B$,LEN(B$)-1):A$=A$+B$
455 GOTO110
460 :
500 PRINTTAB(20)A$
510 GOTO10
```

Die erzeugte Zahlendarstellung

```
1.            E  0
0.            E  0
1.            E -2
1.23          E-03
1.23456       E-11
5.            E  0
1.2345        E  4
1.2345679     E+09
1.23456789    E+17
```

Wolfgang E. Müller, 1000 Berlin 37

# HISTO

Für die Demonstration statistischer Schwankungen wurde ein Versuchsaufbau gewählt, bei dem der Haltepunkt eines auf einer schiefen Ebene abrollenden Wagens gemessen werden sollte. Um die Tauglichkeit der Anlage zu testen, sind zunächst nur 69 Längenmessungen aufgenommen worden; Für den eigentlichen Versuch sind etwa eintausend Messungen erforderlich. Bei der Auswertung mit dem Programm HISTO konnte schon festgestellt werden, daß das Vorkommen zweier Maxima auf einen systematischen Fehler in der Kontruktion schließen ließ. Ein 'Ausreißer', der Wert 2,4 ist offenbar durch fehlerhafte Eingabe eines Wertes zu deuten.

Probelauf des Programmes mit 69 Eingabewerten

```
RUN
V: 1...50? 1
LAST INPUT: 1E10
? 12.4
? 12.5
? 11.3
? 12.4
......
......
......
? 11.5
? 1E10
WAIT

V= 1

DATA        FREQUENZ
--------------------
 2.4        I
 11.1       I
 11.2       IIIIII 11
 11.3       IIIIII 13
 11.4       IIIIII 8
 11.5       II
 12.2       I
 12.3       III
 12.4       IIIIII 13
 12.5       IIIIII 10
 12.6       IIII
 12.7       I
 12.8       I
--------------------
AGAIN WITH NEW V ?
Y OR N? Y
V? 10

V= 10

DATA        FREQUENZ
--------------------
 2.4        I
 :
 :
 :
 :
 :
 :
 :
 11.1       I
 11.2       IIIIII 11
 11.3       IIIIII 13
 11.4       IIIIII 8
 11.5       II
 12.2       I
 12.3       III
 12.4       IIIIII 13
 12.5       IIIIII 10
 12.6       IIII
 12.7       I
 12.8       I
--------------------
AGAIN WITH NEW V ?
Y OR N? N

SUMME       Z= 809
WERTE       I= 69
A.MITTEL    M= 11.72
```

Das für den AIM 65 und seinen Drucker geschriebene Programm HISTO zeigt die Häufigkeit der Meßwerte in einem Balkendiagramm (Histogramm) an. Die maximale Balkenlänge ist 6 Einheiten. Bei einer größeren Frequenz in einer Gruppe wird die Häufigkeit zusätzlich als Zahl ausgedruckt. In Beantwortung der interaktiven Abfrage ist es möglich, mit dem Vergrößerungsfaktor V das Diagramm gleichsam auseinanderzuziehen (oder wieder zu komprimieren). Löcher in einer Meßreihe werden durch Doppelpunkte markiert, so daß ein maßstabsgerechter Überblick einer Zahlengeraden ermöglicht wird. - Die Zahleneingabe in den Computer wird durch den Wert 1E10 als Begrenzer abgeschlossen.

```
100 REM HISTO/PC100/WEMUE
110 REM HAEUFIGKEITSSPEKTRUM EINER FOLGE POSITIVER ZAHLEN
120 L$="--------------------
130 DIMA(150):T=1:T$="IIIIII
140 INPUT"V: 1...50";V:REM VERGROESSERUNGSFAKTOR
150 PRINT!"LAST INPUT: 1E10"
160 INPUTA(I):N=I:REM MESSWERTE
170 IFA(I)<>1E10THENI=I+1:GOTO160
180 POKE 42001,128:REM PRINTER ON
190 PRINT"WAIT"
200 FORI=0TON-1:Z=Z+A(I):NEXT
210 B=0:REM SORT
220 FORI=0TON
230 IFA(I)<=A(I+1)THEN250
240 C=A(I):A(I)=A(I+1):A(I+1)=C:B=1
250 NEXTI
260 IFB=1THEN210
270 D=A(N)-A(1)
280 PRINT" ":PRINT"V=";V:PRINT" ":PRINT"DATA","FREQUENZ":PRINTL$
290 P=N-2:Q=N-1
300 FORI=1TON-1
310 DI=A(N-P)-A(N-Q)
320 X=INT(DI*V/D):S=S+1:S=I
330 IFA(I)<>A(I+1)THENGOSUB480:GOTO350
340 IFA(I)=A(I+1)THENT=T+1
350 IFX<1THEN370
360 FORJ=1TOX:PRINT":":NEXTJ
370 P=P-1:Q=Q-1
380 NEXTI
390 GOSUB480:PRINTL$
400 PRINT"AGAIN WITH NEW V ?"
410 INPUT"Y OR N";V$
420 IF V$="Y"THEN470
430 PRINT" ":PRINT"SUMME","Z=";Z:PRINT"WERTE","I=";I
M=Z/I:M=INT(M*100+.5)/100
450 PRINT"A.MITTEL","M=";M
460 END
470 INPUT"V";V:T=1:GOTO280
480 IFT>6THENPRINTA(I),T$;T:T=1:RETURN
490 PRINTA(I),LEFT$(T$,T):T=1:RETURN
OK
```

Wolfgang E. Müller, 1000 Berlin 37

# SHAKE

Schütteln Sie einmal! Wieviele Möglichkeiten gibt es, die Buchstaben eines Wortes aus fünf Zeichen umzustellen? 5! = 120, gewiß. Aber nun schreiben Sie einmal die 120 Variationen des Wortes SHAKE auf. Wer dabei noch nach längerem Bemühen in Verwirrung gerät, wird den Wert eines entsprechenden Programmes schätzen

Zum Ablauf: Nach RUN wird eine Kette von vier oder fünf Buchstaben, Zahlen oder auch Zeichen eingegeben. Entsprechend der Eingabe erfolgt der Ausdruck von 24 b/w. 120 Variationen. Um weniger als vier Zeichen zu schütteln, braucht man keine Maschine. Sechs oder mehr Zeichen variiert, würden rasch den Speicher des AIM 65 überfluten und, bei Buchstabeneingabe, die Chance immer mehr verringern, sinnvolle Wort-Neuschöpfungen zu bekommen.

Methode: Aus dem eingegebenen Wort (5 Elemente) werden zwei Teilstrings gebildet A$="1234" und B$="5". Die durch Unterstreichen gekennzeichneten Zeichen in der nachstehenden Tabelle werden getauscht und in D$(J) gesammelt. Die letzten beiden Zeichen aller Gruppen aus D$(J) werden anschließend getauscht und bilden nun den Block des Feldes E$(L) Ein weiteres Element, hier "5", kann noch in jede mögliche Position aller zuvor entstandener Gruppen gestellt werden Damit ergeben sich 120 Variationsmöglichkeiten aus fünf Elementen der gegebenen Menge.

| D$(J) | E$(L) | |
|---|---|---|
| 1 2 3 4 | 1 2 4 3 | 5 1 2 3 4 |
| 2 1 3 4 | 2 1 4 3 | 1 5 2 3 4 |
| 3 1 2 4 | 3 1 4 2 | 1 2 5 3 4 |
| 4 1 2 3 | 4 1 3 2 | 1 2 3 5 4 |
| 1 4 2 3 | 1 4 3 2 | 1 2 3 4 5 |
| ... | | |

```
100 REM SHAKE / PC 100 / WEMUE
110 REM VARIATIONEN DER ELEMENTE EINER GRUPPE
120 REM VON 4 BZW. 5 ZEICHEN
130 REM ENTSPRECHEND 24 BZW. 120 LESARTEN EINES WORTES.
140 REM ZUERST WERDEN DIE ELEMENTE NR.1;2, 1;3, 1;4 GETAUSCHT
150 REM UND IN DER MATRIX D$(J) GESPEICHERT,
160 REM ANSCHLIESSEND DER TAUSCH VON NR. 3 MIT NR. 4.
170 REM WAREN 4 ELEMENTE VORGEGEBEN SO ERFOLGT AUSDRUCK.
180 REM EIN FUENFTES ZEICHEN WIRD IN JEDE POSITION
190 REM EINES WORTES DER 24 GRUPPEN GEFUEGT.
200 REM UND 120 FUENFERGRUPPEN AUSGEGEBEN.
210 INPUT "A$";A$: PRINT " ": A=LEN(A$)
220 IF A<4 OR A>5 THEN 210
230 R$=MID$(A$,5,1):A$=MID$(A$,1,4)
240 DIM D$(11): DIM E$(11)
250 FOR I = 1 TO 4
260 GOSUB420: GOSUB480
270 GOSUB440: GOSUB480
280 GOSUB460: GOSUB480
290 NEXT I
300 FOR J= 0 TO 11
310 E$(L)=MID$(D$(J),1,2)+MID$(D$(J),4,1)+MID$(D$(J),3,1)
320 L=L+1
330 NEXT J
340 IF A=4 THEN GOSUB620:END
350 FOR K=0 TO 11
360 GOSUB500
370 NEXT K
380 FOR L=0 TO 11
390 GOSUB560
400 NEXT L
410 END
420 A$=MID$(A$,2,1)+MID$(A$,1,1)+MID$(A$,3,2)
430 RETURN
440 A$=MID$(A$,3,1)+MID$(A$,2,1)+MID$(A$,1,1)+MID$(A$,4,1)
450 RETURN
460 A$=MID$(A$,4,1)+MID$(A$,2,2)+MID$(A$,1,1)
470 RETURN
480 D$(J)=A$:J=J+1
490 RETURN
```

```
500 PRINT R$+D$(K);SPC(1);
510 PRINT MID$(D$(K),1,1)+R$+MID$(D$(K),2,4);SPC(1);
520 PRINT MID$(D$(K),1,2)+R$+MID$(D$(K),3,3)
530 PRINT MID$(D$(K),1,3)+R$+MID$(D$(K),4,2);SPC(1);
540 PRINT MID$(D$(K),1,4)+R$+MID$(D$(K),5,1)
550 RETURN
560 PRINT R$+E$(L);SPC(1);
570 PRINT MID$(E$(L),1,1)+R$+MID$(E$(L),2,4);SPC(1);
580 PRINT MID$(E$(L),1,2)+R$+MID$(E$(L),3,3)
590 PRINT MID$(E$(L),1,3)+R$+MID$(E$(L),4,2);SPC(1);
600 PRINT MID$(E$(L),1,4)+R$+MID$(E$(L),5,1)
610 RETURN
620 FOR K=0 TO 11:PRINT D$(K);:NEXT K
630 FOR L=0 TO 11:PRINT E$(L);:NEXT L
640 RETURN
```

```
RUN
A$? KOELN

NOKEL ONKEL OKNEL
OKENL OKELN
NEKOL ENKOL EKNOL
EKONL EKOLN
NLKOE LNKOE LKNOE
LKONE LKOEN
NKLOE KNLOE KLNOE
KLONE KLOEN
NOLKE ONLKE OLNKE
OLKNE OLKEN
NELKO ENLKO ELNKO
ELKNO ELKON
NLEKO LNEKO LENKO
LEKNO LEKON

NKELO KNELO KENLO
KELNO KELON
NOELK ONELK OENLK
OELNK OELKN
NEOLK ENOLK EONLK
EOLNK EOLKN
NLOEK LNOEK LONEK
LOENK LOEKN
NKOEL KNOEL KONEL
KOENL KOELN
NOKLE ONKLE OKNLE
OKLNE OKLEN
NEKLO ENKLO EKNLO
EKLNO EKLON
NLKEO LNKEO LKNEO
LKENO LKEON
NKLEO KNLEO KLNEO

KLENO KLEON
NOLEK ONLEK OLNEK
OLENK OLEKN
NELOK ENLOK ELNOK
ELONK ELOKN
NLEOK LNEOK LENOK
LEONK LEOKN
NKEOL KNEOL KENOL
KEONL KEOLN
NOEKL ONEKL OENKL
OEKNL OEKLN
NEOKL ENOKL EONKL
EOKNL EOKLN
NLOKE LNOKE LONKE
LOKNE LOKEN
NKOLE KNOLE KONLE
KOLNE KOLEN
```

#

StDir. Peter Rix, 2350 Neumünster

# Druckerausgabe auf TTY (AIM)

Das nachfolgende Programm benutzt die serielle Schnittstelle (TTY) des AIM 65, um einen Drukker mit ebenfalls seriellem Interface (20 mA) zu betreiben, z.B. den EPSON MX80F/T mit Interface Nr. 8141. Es wird keine zusätzliche Hardware benötigt, zur Verbindung der Geräte reicht ein 4-adriges ungeschirmtes Kabel aus, es darf unbedenklich 10m lang sein. Man kann ohne weiteres mit einer Übermittlungsgeschwindigkeit von 9600 Baud fahren.

```
EINFACHVERSION EINES DRUCKERTREIBERS FÜR DEN EPSON MX80FT
-----------------------------------------------------------

DATEINAME: "MXTR2"
RIX, 2350 NMS, 30.11.81

0000 START          =$BA00          ;GGF. ABÄNDERN
0000
0000          ;MONITORADRESSEN
0000 DILINK         =$A406
0000 CNTH30         =$A417
0000 TTYRTN         =$AB00
```

```
0000 OUTTTY          =$EEAB
0000 COMIN           =$E1A1
0000
0000          ;INITIALISIERUNG
0000                 *=START
8A00 START    A917   LDA #<OUT       ;DILINK UMSETZEN
8A02          8D06A4 STA DILINK
8A05          A98A   LDA #>OUT
8A07          8D07A4 STA DILINK+1
8A0A
8A0A          A900   LDA #0          ;9600 BAUD
8A0C          8D17A4 STA CNTH30      ;UBERTRAGUNGSRATE
8A0F          A925   LDA #$25        ;EINSTELLEN
8A11          8D18A4 STA CNTH30+1
8A14          4CA1E1 JMP COMIN       ;ZURUCK ZUM MONITOR
8A17
8A17 OUT      297F   AND #$7F       ;BIT 7 AUSMASKIEREN
8A19          C90D   CMP #$D         ;RETURN?
8A1B          D00A   BNE PRT1        ;DRUCKEN, FERTIG
8A1D          20278A JSR PRT1        ;DRUCKEN
8A20          A90A   LDA #$A         ;LINE FEED
8A22          48     PHA
8A23          A909   LDA #9          ;HORIZONTAL-TAB
8A25          D001   BNE PRT2        ;NACHSCHIEBEN
8A27 PRT1     48     PHA
8A28 PRT2     2C00A8 BIT TTYRTN      ;DRUCKER FERTIG?
8A2B          70FB   BVS PRT2        ;WARTEN, FALLS NEIN
8A2D          4CA9EE JMP OUTTTY+1    ;ZEICHEN DRUCKEN
8A30
8A30                 .END MXTR2
```

#

StDir. Peter Rix, 2350 Neumünster

# Assembler-Reformattor

**für AIM mit TTY-Ausgabe**

Heft 9 dieser Zeitschrift enthielt mit der Arbeit 'Assembler-Listing für den AIM 65' von Horst Steder das Werkzeug, das die Assembler-Liste des AIM überhaupt erst gut lesbar machte, indem es die Begrenzung der Zeilenbreite auf 20 Zeichen aufhob und den Zeilen den Stand des Zuordnunszählers voranstellte. Der TPASM ist inzwischen ein Standard geworden, wie man den vielen seither veröffentlichten Assembler-Listings entnehmen kann. Der TPASM war seinerzeit für eine IEEE488-Schnittstelle geschrieben. Hier folgt nun eine Überarbeitung für TTY-Schnittstelle, die vielen Lesern die Mühe der Umkodierung ersparen dürfte und die man nicht nur für einen seriell angeschlossenen Drucker, sondern auch für ein serielles Video-Display benutzen kann.- Hinsichtlich der Kommentierung sei im wesentlichen auf die zitierte Arbeit verwiesen.

```
DIESE REFORMATERVERSION TREIBT DEN EPSON MX80F/T
MIT SER. INTERFACE NR.8141
UNMITTELBAR AUS DER 20MA-STROMSCHLEIFE DES AIM-65

      ";.CTRLX" KOMMENTAR LINKSBUNDIG
      ";."      KOMMENTAR BUNDIG MIT HEXCODE
      ";"       KOMMENTAR RECHTS AUSGERUCKT

ASSEMBL.REFORMATER VON STEDER,MICROMAG.9.79
GEANDERT VON RIX 24.11.81 FUR 20MA/MX-80
DATEINAME: "ASSRX"
```

```
0003 START          =$F00           ;GGF. ABANDERN
0003
0003 CNTH30         =$A417
0003 ASSMBL         =$D000
0003 EQUAL          =$E7D8
0003 INPUT          =$E95F
0003 TTYRTN         =$A800
0003 OUTTTY         =$EEA8
0003 PHXY           =$EB9E
0003 PLXY           =$EBAC
0003 UOUT           =$10A           ;USER-OUT-VEKTOR
0003 COMFLD         =35             ;BEGINN KOMMENTAR
0003 ZEILEN         =65             ;ZEILEN/SEITE
0003
0003 ADDRS          =$32            ;ZEROPAGE
0003                *=$F0
00F0 CNTR           *=*+1           ;ZEICHENZAHLER
00F1 EQFLG          *=*+1           ; '='-FLAG
00F2 CRFLG          *=*+1           ; 'CR'-FLAG
00F3 SAVA           *=*+1
00F4 SAVX           *=*+1
00F5 SMFLG          *=*+1           ; ';'-FLAG
00F6 ICNT           *=*+1           ;TAB-ZAHLER
00F7 ZCNT                           ;ZEILENZAHLER
00F7
00F7                *=UOUT
010A          0D0F  .WOR STRT1      ;LINK-ADRESSE
010C
010C                *=START
0F00 START    A90D  LDA #<STRT1     ;REFORMATER-START
0F02          8D0A01 STA UOUT       ;AUS ROM
0F05          A90F  LDA #>STRT1
0F07          8D0B01 STA UOUT+1
0F0A          4C00D0 JMP ASSMBL     ;ASSEMBLER-START
0F0D
0F0D STRT1    B02C  BCS TDSP        ;ASS. OUT=U
0F0F          D8    CLD
0F10          A900  LDA #0          ;20MA-SCHLEIFE
0F12          8D17A4 STA CNTH30     ;AUF
0F15          A925  LDA #$25        ;9600 BAUD
0F17          8D18A4 STA CNTH30+1   ;EINSTELLEN
0F1A          A911  LDA #$11        ;CTRL Q
0F1C          20A8EE JSR OUTTTY     ;DRUCKER EIN
0F1F          A941  LDA #ZEILEN
0F21          85F7  STA ZCNT
0F23 CRLF     A90D  LDA #$D         ;'RETURN'
0F25          C6F7  DEC ZCNT        ;'SEITENENDE'
0F27          D006  BNE PRT
0F29          A941  LDA #ZEILEN
0F2B          85F7  STA ZCNT
0F2D          A90C  LDA #$C         ;'FORMFEED'
0F2F PRT      20DC0F JSR PRTR
0F32          A200  LDX #0
0F34          86F0  STX CNTR
0F36          86F2  STX CRFLG
0F38          86F5  STX SMFLG
0F3A          60    RTS
0F3B
```

```
0F3B TDSP    86F4    STX SAVX        ;FORMATIERROUTINE
0F3D         68      PLA
0F3E         297F    AND #$7F        ;ENTFERNE BIT 7
0F40         85F3    STA SAVA
0F42         C90D    CMP #$0D        ;'RETURN' ?
0F44         D003    BNE N1
0F46         66F2    ROR CRFLG
0F48         60      RTS
0F49
0F49 N1      C93D    CMP #'='        ; '==ADRESSE'?
0F4B         D014    BNE N6
0F4D         24F2    BIT CRFLG
0F4F         1008    BPL N2
0F51         20230F  JSR CRLF
0F54         38      SEC
0F55         66F1    ROR EQFLG
0F57         306D    BMI R1
0F59 N2      24F1    BIT EQFLG
0F5B         1004    BPL N6
0F5D N3      46F1    LSR EQFLG
0F5F         1065    BPL R1
0F61 N6      24F2    BIT CRFLG
0F63         1045    BPL N10
0F65         C93B    CMP #';'
0F67         D02E    BNE N8
0F69         66F5    ROR SMFLG
0F6B         46F2    LSR CRFLG
0F6D         1057    BPL R1
0F6F
0F6F NC      E080    CPX #$80
0F71         F003    BEQ NK
0F73         20E60F  JSR RESET
0F76 NK      A921    LDA #COMFLD-2
0F78         85F5    STA SMFLG
0F7A         C5F0    CMP CNTR
0F7C         B002    BCS SP1
0F7E         A5F0    LDA CNTR
0F80 SP1     6902    ADC #2
0F82         85F6    STA ICNT
0F84 SP2     A6F0    LDX CNTR
0F86         E4F6    CPX ICNT
0F88         F006    BEQ N7
0F8A         20C10F  JSR SPACE
0F8D         4C840F  JMP SP2
0F90
0F90 N7      A93B    LDA #';'
0F92         20DC0F  JSR PRTR
0F95         D00B    BNE NN
0F97 N8      A6F0    LDX CNTR
0F99         E00C    CPX #12         ;BEGINN HEXCODE PRI
0F9B         B00A    BCS N9
0F9D         46F2    LSR CRFLG
0F9F         20F30F  JSR COUNT
0FA2 NN      A5F3    LDA SAVA
0FA4         4CC30F  JMP NEXT
0FA7 N9      20E60F  JSR RESET
0FAA N10     A5F3    LDA SAVA
0FAC         A6F5    LDX SMFLG
```

```
OFAE            100D    BPL N11
OFB0            C92E    CMP #'.'
OFB2            D0BB    BNE NC
OFB4            20E60F  JSR RESET
OFB7            A93B    LDA #';'
OFB9            85F5    STA SMFLG
OFBB            D006    BNE NEXT
OFBD N11        C97F    CMP #$7F
OFBF            9002    BCC NEXT
OFC1 SPACE      A920    LDA #$20
OFC3 NEXT       20DC0F  JSR PRTR
OFC6 R1         A6F4    LDX SAVX
OFC8            60      RTS
OFC9
OFC9 NUMA       48      PHA
OFCA            4A      LSR A
OFCB            4A      LSR A
OFCC            4A      LSR A
OFCD            4A      LSR A
OFCE            20D40F  JSR NOUT
OFD1            68      PLA
OFD2            290F    AND #$F
OFD4 NOUT       0930    ORA #$30
OFD6            C93A    CMP #$3A
OFD8            9002    BCC PRTR
OFDA            6906    ADC #6
OFDC PRTR       2C00A8  BIT TTYRTN      ;DRUCKER FERTIG?
OFDF            70FB    BVS PRTR        ;WARTESCHLEIFE
OFE1            E6F0    INC CNTR
OFE3            4CA8EE  JMP OUTTTY      ;ZEICHEN AUSGEBEN
OFE6
OFE6 RESET      20230F  JSR CRLF
OFE9            A533    LDA ADDRS+1     ;ADRESSE
OFEB            20C90F  JSR NUMA        ;DRUCKEN
OFEE            A532    LDA ADDRS
OFF0            20C90F  JSR NUMA
OFF3 COUNT      20C10F  JSR SPACE
OFF6            A6F0    LDX CNTR
OFF8            E00D    CPX #13
OFFA            D0F7    BNE COUNT
OFFC            60      RTS
OFFD
OFFD                    .END ASSRX
```

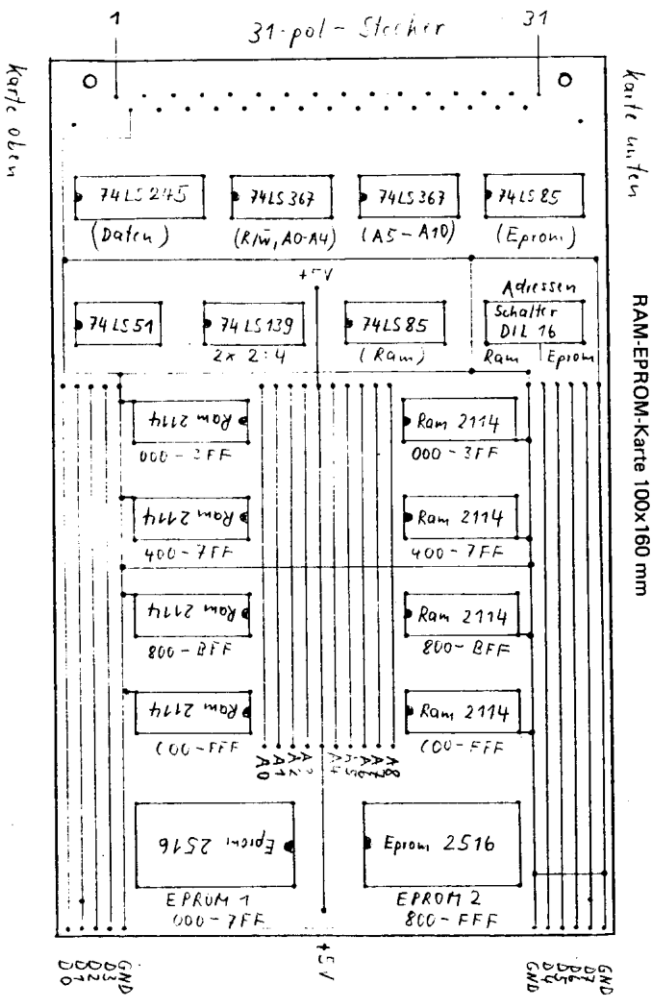
Anton Dichtel, 7800 Freiburg

# RAM-EPROM-Karte 4 KB & 4 KB

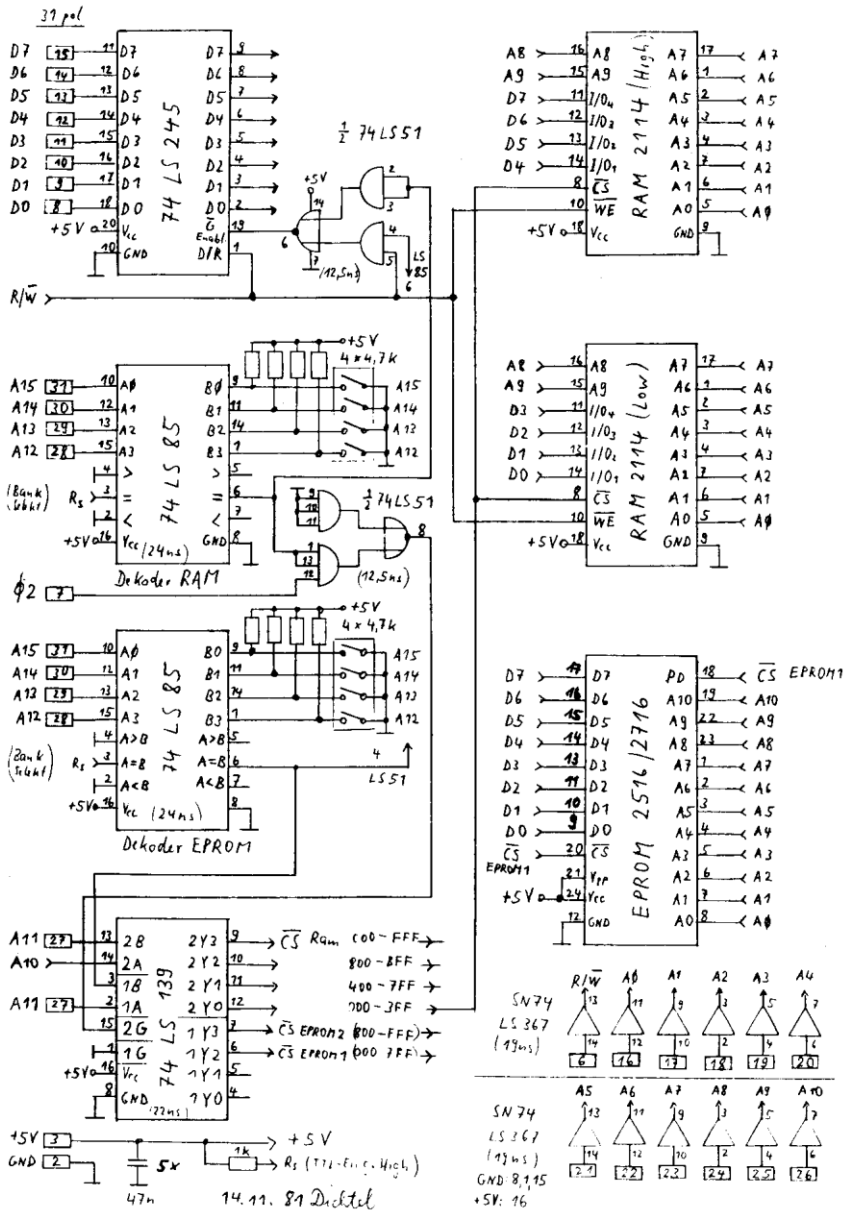
Auf einer Veroboardplatte im Europaformat wurden 4 KB RAM (8x2114, 450 ns) und 4 KB EPROM (2x2516/2716, 450 ns) untergebracht. Dabei sind die RAM- und EPROM-Adressen unabhängig voneinander über DIL-Schlater im gesamten 64-K-Bereich verschiebbar. Daten und Adressen sind voll gepuffert und stellen für den 31-poligen 'hauseigenen Sparbus' eine Belastung von 1xLS oder weniger dar. Ausnahmen: A12-A15: 6xLS, A10: 2xLS +2x A10 von 2516).

Besonderheiten: Der Baustein 74LS51 liefert ein NAND und ein NOR. Nebenbei sorgt er dafür, das der Datenbus-Treiber im hochohmigen Zustand bleibt, falls versucht wird, in den EPROM-Bereich zu schreiben. Anschluß 3 der beiden 74LS85 ermöglicht ein zusätzliches Bank-Select der Karte. Die Schaltung zum DIL-Schalter führt nur Gleichspannung und läßt sich beliebig verlängern.

Der Bestückungsplan zeigt, daß versucht wurde, den Verdrahtungsaufwand gering zu halten. Es wurde eine Handverdrahtung vorgenommen, so daß beim Autor auch keine Platine erhältlich ist. Die Masseleitung ist netzförmig ausgelegt (Draht 1 mm stark) und die Betriebsspannung an mehreren Kreuzungspunkten mit 47 µF abgeblockt.
-Die Karte wird hier nicht direkt am AIM 65 betrieben, sondern huckepack auf der AIM-Platine sitzt ein Adreß- und Datenbustreiber mit 1x 74LS245, 3x 74LS244 und 1x 74LS145 als Dekoder für den Bereich hex 1000-9FFF. - Außer als Speicher dient sie auch dem Austesten von Programmen als Vorbereitung für das Brennen in EPROMs.



# RAM / ROM - Karte 4K + 4K Byte



# BASIC mit Struktur

Die höhere Programmiersprache BASIC ist leicht zu erlernen und bietet in den meisten Implementierungen komfortable Möglichkeiten sowohl für die Berechnung höherer mathematischer Funktionen als auch zur Stringbearbeitung. Sie fordert vom Programmierer allerdings auch keine besondere Disziplin bei der Anordnung seiner Statements oder für die Erklärung der benutzten Variablen. Wir finden daher eine wahre 'Freistil-Show' der Programme. - Einige Anwender bemühen sich jedoch, für den Haugebrauch nach in etwa gleichbleibenden Strukturen zu programmieren und auch zu dokumentieren. Andere überlassen es von Aufgabe zu Aufgabe mehr oder minder dem Zufall, wie sich die Statements aneinanderreihen, Hauptsache, 'das Programm läuft'. Man spricht daher auch von 'Spaghetti-Programmierung', Anfänge und Enden liegen durcheinander. Es kommt vor, daß eine Anweisung wegen zu engen Zeilenabstandes nicht mehr im linearen Verlauf angeordnet werden konnte, man springt daher mit einem GOTO zu einem 'Rucksack' und später von dort zurück, statt besser mindestens ein RENUMBER der Zeilen durchzuführen.

Es ist nicht zuletzt diese Kritik am Freistil, die seitens der Hochschulen sauber strukturierte Sprachen fordern und z.B. PASCAL für Lehrzwecke bevorzugen ließ. - Wenn es für BASIC auch keine Einbuße an stilistischer Freiheit gibt, so sollen hier doch einmal einige Gesichtspunkte der Strukturierung und weiterer Zweckmäßigkeiten angeregt und zur Diskussion gestellt werden.

**Die innere Sprachstruktur**

Als Interpretersprache ordnet BASIC die Befehlszeilen intern in aufsteigender Numerierung an. Am Kopfe jeden gespeicherten Statements steht im 'LINK', unter welcher Speicheradresse das nächste LINK zu finden ist. Darauf folgt in zwei Bytes die Zeilen-Nummer des laufenden Statements. Während der Programmabarbeitung ergeben sich damit 3 Möglichkeiten:

a) Im linearen Ablauf ist das nächste Statement abzuarbeiten. Das wird mit dem geringsmöglichen Zeitaufwand erreicht.

b) Ein GOTO oder GOSUB weist auf eine höhere Zeilennummer. Es hängt von der Menge der dazwischen liegenden nicht benötigten Statements ab, nach welcher Zeit die Ziel-Zeile gefunden ist.

c) Ein GOTO oder GOSUB weist auf eine niedrigere Zeilennummer hin. Jetzt müssen alle Statements von Anfang an durchkämmt werden, ob sie dem zu erreichenden entsprechen. Auch hier ist die Zeit eine Funktion der zu überstreichenden Zeilen.

Nur wenige Programme werden ein einziges Mal linear durchlaufen. Es ist vielmehr Regel, daß Programmabschnitte sehr häufig durchlaufen werden. Damit wirkt sich die Anordnung der Abschnitte erheblich auf die Ausführungszeit aus.Man sollte also, statistisch auf die Häufigkeit gesehen, mit durchschnittlich kurzen Schritten zur nächsten benötigten Zeilennummer vorrücken können.

**Blockmäßige Programmanordnung**

Die für Wiederholungen grundsätzlich empfohlene Unterprogrammtechnik führt zu weiteren Überlegungen: Häufig benötigte Unterprogramme sollten am Programmanfang schnell erreichbar sein. Je nach Aufgabe werden dieses rechenintensive Programmabschnitte sein (Iterationen), solche einer zeitaufwendigen Stringumwandlung/Umcodierung oder solche der Ein- und Ausgabe.- Die Beobachtung zeigt, daß man Unterprogramme bisher stattdessen im allgemeinen an den Programmschluß legte.

Nun ist Gelegenheit, in die Abfolge der Unterprogramme zugleich auch 'hausinterne Korsettstangen' einzuziehen, so daß sich etwa folgende blockmäßige Programmanordnung ergeben könnte:

```
10      GOTO 20000 : REM Sprung zum Hauptprogramm
100     Unterprogramme, z.B. Rechnen, Stringmanipulation
500     GET-Schleifen
1000    INPUT von Floppy
2000    PRINT auf Floppy
3000    Drucker-Hauptunterprogramme
        Formatierungsunterprogramme
5000    Ausgabe auf andere Peripherie
        Formatierungsanweisungen für andere Peripherie
8000    DATA-READ-Routinen
10000   DATA-Statements, gesammelt
20000   Hauptprogramm
        Initialisierung der am häufigsten gebrauchten Variablen
        Menue zur Hauptablaufsteuerung
        Befehlsebene mit ON ... GOTO 30000,40000 usf.
```

Je nach Bedarf wird man also drei große Segmente haben: Die Unterprogramme, die DATA und das Hauptprogramm. Man sollte also etwas planen und sich dann an ein durchdachtes Prinzip halten.

**Behandlung der Variablen**

BASIC weist Variablen erst zur Laufzeit einen Speicherplatz zu, und zwar in der Reihenfolge, in der sie zur Benutzung kommen, nicht in der Reihenfolge, in der sie im Programm stehen. Bei der vorstehenden Programmanordnung würden also zunächst die nach Statement 20000 benutzen Variablen angelegt und nicht die bei 100 benutzten.

Für Zählschleifen (FOR ... TO ...) wird man immer einige Laufvariable benötigen, die man immer wieder verwenden kann, z.B. I, J, K ... . Andererseits braucht man eigentlich immer Stringvariable, um interaktive Abfragen oder andere Ausgaben zu machen. Man bemühe sich, mit möglichst wenigen Variablen auszukommen. Oder anders: Ein Variablenname sollte möglichst oft, auch in anderen Aufgaben, wiederverwendet werden. Es ist dann aber zu empfehlen, ein erklärendes REMARK in das Statement aufzunehmen, um die derzeitige Rolle der Variablen zu beschreiben, etwa als REM A=Kundenummer

Dahinter steht folgende Überlegung: Variable stehen im Variablenbereich des BASIC in einer Art 'Warteschlange' in der Abfolge der bisherigen Benutzung. Der BASIC-Interpreter muß diesen Bereich bei der Ansprache einer Variablen in der Abfolge durchkämmen und eine Variable sogar neu anlegen, wenn sie noch unbenutzt war. Die Suchzeit ist abgekürzt, wenn man schnell auf einen bereits bekannten Namen stößt. Wir haben daher im vorstehenden Blockdiagramm ab 20000 die Tätigkeit 'Initialisierung der am häufigsten gebrauchten Variablen' eingeschrieben, um für die Ständig wiederverwendeten und ggfs. zeitkritischen Variablen eine optimale Anordnung zu ermöglichen, z.B.:

```
20010   I=0:K=0:A2=0:A$="":B$=""
```

Arrays: Man sollte sich nicht scheuen, die Felder eines Datensatzes in ein Array zu stellen. Ein Programm ist für die Zukunft sicher pflegefreundlicher, wenn man z.B. für die Eingabe eines Kundenkontos mit 14 Feldern von Floppy schreibt:

```
1000    FOR I=0 TO 13: INPUT#8,K$(I): NEXTI: RETURN
20000   DIM K (13): REM Hier ist das Hauptprogramm
```

Es sieht wohl besser aus, als wenn man z.B. schreiben würde:

```
1000    INPUT#8, KN$ ,AN$ ,VN $,NN$',...
```

Analog hat man es z.B. bei der Ausgabe nur des Adressenfeldes im Konto wesentlich einfacher, wenn man schreibt:

```
3100    FOR I=0 TO 6: PRINT#4,K$(I):NEXT I:RETURN
```

Die vorstehenden Handhabungen setzzen voraus, daß man die extern gespeicherte Information grundsätzlich in Strings speichert (was sehr zu empfehlen ist) und nicht gemischt in Strings und in Zahlendarstellung.

**Interaktives PRINT und INPUT" ..."**

In den veröffentlichten Programmen finden wir für die Benutzerführung immer einen ganzen Rattenschwanz gleichnamiger Anweisungen wie:

```
80    PRINT"Dies" oder
90    INPUT"Das";B $ oder
500   PRINT"Jenes"
510   GET A $:IF A$ = ""THEN510
520   IF A$ ="X"THEN ....'
530   IF A$ ="Y"THEN ....
```

Es wäre doch viel bequemer zu schreiben

```
80    PRINTA$
```

wenn man vorher A$ aus einer DATA-Tabelle mit READ gefüllt hat, siehe unten.

Die Zeile 510 sollte man als immer wieder aufzurufendes Unterprogramm schreiben, z.B. als:

```
510   GET A $:IF A$ =""THEN 510
520   RETURN
```

und für den Rattenschwanz der Abfrage ab Zeile 520 sollte man eine komprimierte wiederverwandbare Befehlsebene mit ON ... GOTO ... schaffen, siehe unten.

**RESTORE DATA-Pointer to Line Number ...**

Heft 9 dieser Zeitschrift enthielt ein maschinensprachliches Programm, das den von BASIC unterhaltenen Pointer auf die DATA-Statements auf eine bestimmte DATA-Zeile setzt, so daß man das nächste DATA ganz gezielt heranziehen kann. Dieses Prinzip ist inzwischen in vielen käuflichen BASIC-Erweiterungen benutzt worden. Man kann den gleichen Erfolg aber auch mit reinen BASIC-Statements erzielen:

Bekanntlich speichert der Interpreter zwischenzeitlich immer den Pointer auf die nächste Programmzeile. Wenn man nun vor die anzusteuernde DATA-Zeile ein einzeiliges Unterprogramm setzt und aufruft, das nur den Folgezeilen-Pointer in den DATA-Pointer umPOKEd, dann ist schon alles getan. Im weiter unten stehenden Beispielprogramm machen wir z.B. in den Zeilen 10000 und 10100 davon Gebrauch und können danach die vielen PRINT"Dies und das"-Statements umfunktionieren zu schlicht PRINT A und sie sogar auch in FOR ...NEXT-Schleifen einbinden.

Eine weitere Möglichkeit ist die 'DATA-Pumpe' Man kann eine allgemeine Dienstleistung installieren, die DATA nur liest (ohne PRINT), um den DATA-Pointer weiterzubewegen. Beim nächsten PRINT hat man dann das gewünschte Item auf dem Bildschirm. Unser Beispielprogramm macht davon ab Zeile 8000 Gebrauch.

**Befehlsebenen**

Befehlsebenen werden üblicherweise mit GET und einem erwarteten Tastendruck eingerichtet. Es folgt eine längliche Abfrage mit IF ... THEN, wie etwa oben ab Zeile 520, um festzustellen, welche Taste gedrückt wurde. Es ist viel kompakter und universeller, alle erlaubten Tastendrücke in eine Stringvariable einzustellen und den String dann abzukämmen, ob er das Tastenzeichen enthält. Danach geht man mit dem Statement ON I GOTO ..., ..., in die jeweilige Ausführung. Unser Beispiel macht ab 20210 davon Gebrauch.

**Zusammenfassung**

Mit etwas Überlegung läßt sich auch für BASIC-Programme ein knapper und möglichst schnell ausführender Code entwickeln, der zudem eine in etwa wiederkehrende Gliederung hat, die den Wünschen des Betreibers dient und die Dokumentation und Programmpfelge verbessert. Dabei kann auch einiges für Dritte getan werden, die das Programm verstehen sollen. Die Schaffung von Mo-

dulen erleichtert zugleich die künftige Programmierung, denn man kann dann auf die erhältlichen MERGE-Befehle zurückgreifen. - Es gibt natürlich soviele tausend BASIC-Betreiber mit ausgeprägten Erfahrungen, daß diese Überlegungen nur ein Anfang sein können. 65xx MICRO MAG wird gern auf weitere Vorschläge eingehen.

```
10 GOTO20000:REM SPRUNG ZUM HAUPTPROGRAMM
100 :
500 GETT$:IFT$=""THEN500:REM GET-ROUTINE
510 RETURN
550 GOSUB500:FORI=1TOLEN(G$):IFMID$(G$,I,1)=T$THENZ=I:I=LEN(G$):NEXTI
560 RETURN
600 :
8000 REM FORI=0TOZ:READG$:NEXTI:RETURN:REM BLINDES LESEN,  DATENPUMPE
8010 FORI=0TOZ:READG$:PRINTG$:NEXTI:RETURN
8100 :
10000 DATA1,2,3,4,5
10010 :
10040 P=PEEK(58):POKE62,P:P=PEEK(59):POKE63,P:RETURN
10050 DATA"MENUE",,"1 JOB1",,"2 JOB2",,"3 JOB3",,"4 JOB4",,"5 JOB5",
10060 DATA"6 JOB6",,"7 JOB7",,"8 JOB8",,"9 JOB9",,,
10070 DATA"WAHL MIT TASTENDRUCK"
11100 P=PEEK(58):POKE62,P:P=PEEK(59):POKE63,P:RETURN
11110 DATA,"DATUMSEINGABE",,"TAG",,"MMNAT",,"JAHR",,"OKAY? J/N"
11300 :
20000 POKE59468,14:REM EINSCHALTUNG ZEICHENSATZ
20010 ZW$="":Z=0:T$="":G$="":J=C:I=0:K=0:REM INITIALISIERE VARIABLE
20200 GOSUB10040
20204 Z=22:GOSUB8010:REM MENUE ANZEIGEN
20210 G$="123456789":GOSUB550:REM KOMMANDOEBENE
20300 ONZGOTO31000,32000,33000,34000,35000,36000,37000,38000,39000
```

R.L. 4

*K. F. Reinstein, 5880 Lüdenscheid*

## Geisterzeilen im Microsoft-BASIC

*Ein Bekannter hatte sich ein Programm gekauft, das mit einer Copyright-Zeile abschloß, etwa in der Form:*

*64000 PRINT"Copyright XYZ  11.11.81":END*

*Da er das Programm in teilweise veränderter Form benutzt und neue Teile angefügt hatte, wollte er diese Zeile aus dem Programm antfernen und evtl. durch eine eigene Zeile ersetzen. Das geschieht normalerweise durch die Eingabe 64000 und RETURN. Das Ergebnis war negativ Die Zeile ließ sich weder aus dem Programm entfernen, noch ändern und tauchte beim Listen immer mit auf. Sie wurde im Programm korrekt ausgeführt. - Was tun? Was ist mit einer solchen Zeile los, und wie kann das für andere Zwecke benutzt werden?*

*Laut Manual sind Zeilennummern nur bis 63999 erlaubt. Sieht man sich einmal das Maschinenprogramm zur Dezimal-Hexumwandlung an (z.B. für AIM 65/PC 100 im GWK-Listing, Adresse B7DA) so ist diese Begrenzung auch verständlich. Die LIST-Routine verwendet zur Rückrechnung der BASIC-Zeilennummer die normale Routine (CB0C), die auch größere Zahlen zurückwandeln kann, wie bei normalen Variablen. Daher werden die Zeilennummern größer 63999 korrekt ausgegeben.*

*Weiterhin: Die Ausführungsroutine schert sich nicht um Zeilennummern, sondern hangelt sich an den Anfangsadressen für die folgende BASIC-Zeile entlang. Nur bei Sprüngen (GOSUB, GOTO) wird die ominöse beschränkte Umwandlungsroutine verwendet. Daher können Zeilen mit Nummern ab 64000 voll gefahren werden, außer bei diesen beiden Befehlen.*

*Die Eingaberoutine sucht ebenfalls mit der beschränkten Routine die entsprechende Zeilennummer und kann sie somit weder finden noch ändern, geschweige denn löschen. Damit ergeben sich Möglichkeiten, Zeilen oder Programmteile zu schützen, wenn man den hohen Zeilenummern-Raum ausnutzt. Eine Änderung, bzw. die Eingabe der Zeilennummer kann normalerweise nur im Maschinenmonitor erfolgen. Interessanterweise kann man dann sogar doppelte Zeilennummern benutzen, wenn man eine Blockstruktur und jeden Programmblock mit einer größeren ZN beginnt, da dann die folgenden Zeilen automatisch geschützt sind, z.B.:*

```
64000 REM Eingabe
   10 INPUT ...
64500 REM Ausgabe
   10 PRINT
   20 PRINT
64600 REM Berechnung
   10 X= ...
```

*Schließlich noch eine Anmerkung für TRS 80-Besitzer. Bei diesem System werden die größeren Zeilennummern auch beim Listen ausgelassen, obwohl sie voll funktionsfähig sind, stellen also echte Geisterzeilen dar (andere größte Nummer!).*
*Literatur: GWK BASIC-Listing für AIM, 1979, Herzogenrath. Zur Darstellung der ZN: 65xx MICRO MAG, Heft 10 'AIM Spezial 5'.*

#

*Friedrich Geissler, 8120 Weilheim*

# Grafik-Plot am AIM/PC 100

*Seit einiger Zeit gibt es Video-Displays mit Einzelpunktansteuerung, mit denen man recht gut zweidimensionale Funktionen (Z=f(X,Y)) darstellen kann. Dadurch angeregt, versuchte ich etwas Ähnliches am Thermodrucker des PC100 zu realisieren. Der Thermodrucker ist ja bekanntlich per Software anzusteuern. Dabei sollte das Programm schon auf der 4-K-Version des AIM lauffähig sein.*

*Um ein annähernd quadratisches Bild zu bekommen, bietet sich eine Auflösung von 100x100 Bildpunkten an. Das ergibt einen minimalen Speicherbedarf von 10.000 Bit = 1250 Byte. Das vorliegende Programm benötigt 1300 Byte Bildspeicher, um eine einfachere Adressierung zu bekommen. Ein Byte stellt 8 übereinanderliegende Bildpunkte dar, das nächsthöhere Byte die rechts daneben liegenden 8 Bildpunkte. Das ergibt 12,5 Zeilen mit je 100 Byte = 1300 Byte je Bild. Dabei werden in der obersten Zeile nur die unteren 4 Bit benutzt. - Der Synchronisationsteil des Programmes entstand aus einem vorhandenen Programm für das Plotten von eindimensionalen Funktionen (1). Es war schwierig, die nötige Geschwindigkeit zu erreichen, denn bei jeder Zeile, die ausgegeben wird, muß je 10mal für die 10 Thermoelemente berechnet werden, ob der Punkt zu setzen ist oder nicht.*

*Das Assembler-Programm ist geeignet, beliebige Punktkombinationen auszugeben, die in der Bildpunkttabelle ab "TABAN" abgespeichert sind. Denkbar wären außer zweidimensionalen Funktionen auch: mehrere eindimensionale Funktionen übereinander, Relationen, komplexe Funktionen, Bildgrafik oder sogar japanische Schriftzeichen.*

*Das vorliegende BASIC-Programm berechnet für ein Feld (X,Y) mit 80x50 Punkten den Z-Wert, führt eine Koordinatentransformation auf die Druckerkoordinaten durch (für die perspektivische*

*Darstellung) und setzt die entsprechenden Bildpunkte. Dabei ist das BASIC im Speicherbereich auf 2300 zu begrenzen (memory size?). In der Betriebsart "Fein" wird jeweils eine Linie zwischen dem alten und dem neuen Y-Wert gelöscht, um zu erreichen, daß Punkte gelöscht werden, die hinter dem vorgestellten Gebilde liegen (damit es nicht 'durchsichtig' wird).*

*Zum Betrieb: Vor Benutzung ist das Programm unbedingt sorgfältig zu überprüfen, dann muß man die Druckgeschwindigkeit mit dem Potentiometer VR3 auf der Platine des AIM auf eine Geschwindigkeit bringen, bei der ein klares Bild erscheint. Dazu läßt man am besten eine Funktion grob berechnen (z.B. Z=0) und mehrmals bei veränderter Druckgeschwindigkeit mit GOTO 330 ein Bild ausgeben, bis es optimal eingestellt ist (bei Z=0 ergibt sich ein Netz, das in Rautenform aufgespannt ist). Nach Start des BASIC-Programmes werden die gewünschten Bereiche abgefragt, in denen die Funktion zu plotten ist. Dann wird die Betriebsart festgelegt: 'Grob/Fein'.*

*Bei 'Grob' wird nur ein Netz erzeugt, das die Funktion in verminderter Auflösung darstellt, und es werden auch überdeckte Bildpunkte nicht gelöscht (Rechenzeit ca. 3-4 min.). Bei 'Fein' wird mit optimaler Auflösung gearbeitet (Rechenzeit ca. 10-15 min.). Die Funktion ist in den Zeilen 2030-2050 einzusetzen (Z=f(X,Y)). Das Programm schaltet den Drucker am Beginn aus und am Ende zum Listen der Funktion wieder ein.*

*Hier noch zwei Beispiele von Funktionen, die ein 'schönes' Bild liefern:*

*Z=1/SQR(1+X*X+Y*Y) mit X: -6,6 Y: -6,6 Z: 0,1*

*Z=SQR(X*X+Y*Y):Z=EXP(-Z/2.5)*COS(Z) mit X: -6,6 Y: -6,6 Z: -7,1*

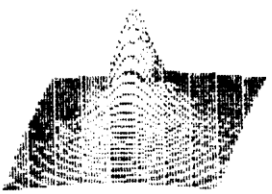
*Literaturhinweise: (1) Plotprogramm für AIM, Funkschau 12/80, von Michael Konz, (2) PC 100 Monitor-Listing.*

```
0 REM --- PLOTPROGRAMM FUER MEHRDIMENSIONALE FUNKTIONEN
1 REM ---                        VERSION 2.9
2 REM ---        14.1.1982                    GEISSLER FRIEDRICH
5 POKE42001,0:REM - DRUCKER AUSSCHALTEN
10 REM
20 REM - LOESCHEN
30 REM
40 POKE4,97:POKE5,15:Q=USR(0)
50 REM
60 REM - EINGABE
70 REM
80 INPUT"X - BEREICH";A,B:IFA>=BTHEN80
90 INPUT"Y - BEREICH";C,D:IFC>=DTHEN90
100 INPUT"Z - BEREICH";E,F:IFE>=FTHEN100
110 XS=(B-A)/80:YS=(D-C)/50:ZS=(F-E)/65
120 XN=-A/XS:YN=-C/YS:ZN=-E/ZS
130 INPUT"   GROB   (Y/N)";A$:IFA$="Y"THEN500
140 IFA$<>"N"THEN130
150 REM
160 REM - MAXIMALE PUNKTDICHTE
170 REM
180 FORX=ATOBSTEPXS
190 PRINTINT((X-A)*1.25/XS+.5);"% BERECHNET"
200 FORY=DTOCSTEP-YS
210 GOSUB2000:REM - FUNKTION UND TRANSFORMATION
220 REM
230 REM - LINIE LOESCHEN
240 REM
250 POKE42104,X1:POKE42103,Y2:POKE42102,Y1
260 POKE4,165:Q=USR(0)
270 REM
```

```
280 REM - SETZEN EINES PUNKTES
290 REM
300 POKE42103,Y1:POKE4,138:Q=USR(0)
310 Y2=Y1
320 NEXTY,X
330 REM
340 REM - AUSGABE DES BILDES
350 REM
360 POKE4,32:POKE5,14:Q=USR(0)
370 PRINT!" ":PRINT!"X :"A"-"B
380 PRINT!"Y :"C"-"D:PRINT!"Z :"E"-"F
390 POKE42001,128:REM - DRUCKER EINSCHALTEN
400 LIST2031-2049
500 REM
510 REM - GROBES NETZ FUER UEBERBLICK
520 REM
530 PRINT"AUGENBLICK BITTE !"
540 FORX=ATOBSTEPXS*5
550 FORY=DTOCSTEP-YS
560 GOSUB2000:REM - FUNKTION UND TRANSFORMATION
570 POKE42103,Y1:POKE42104,X1:POKE4,138:Q=USR(0)
580 NEXTY,X
590 FORX=ATOBSTEPXS
600 FORY=DTOCSTEP-YS*5
605 GOSUB2000
610 POKE42103,Y1:POKE42104,X1:Q=USR(0)
620 NEXTY,X
630 GOTO330
2000 REM
2010 REM - FUNKTION UND KOORDINATENTRANSFORMATION
2020 REM
2030 REM - HIER FUNKTION EINSETZEN:
2040 Z=1/SQR(1+X*X+Y*Y)
2050 REM
2060 IFZ>FTHENZ=F
2070 IFZ<ETHENZ=E
2080 X1=X/XS+XN:Y1=Y/YS+YN:Z1=Z/ZS+ZN
2090 X1=INT(X1+.38*Y1+.5):Y1=INT(Z1+Y1+.5)
2100 RETURN
```

```
0000            ; ================
0000            ; P L O T - 2  !!!
0000            ; ================
0000            ;
0000            ; VERSION 2.9
0000            ; PLOTPROGRAMM FUER GRAPHIK
0000            ; UND MEHRDIMENSIONALE FUNKTIONEN
0000            ; 14.01.1982
0000            ; GEISSLER FRIEDRICH
0000
0000            ;-----------------------
0000              PARAMETER
0000
0000 TABAN              =$09            ;TABELLENANF.
0000
0000            ;-----------------------
0000            Z-PAGE-ADRESSEN
0000
0000                    *=$00A0
```

```
00A0 BYTE           *=*+2           ;ZEILENGRUNDBYTE
00A2 YBYTE          *=*+1           ;HILFSVARIABLE
00A3 YBIT           *=*+1           ;BITMASKE
00A4 YBIT1          *=*+1           ;BITZAHL
00A5
00A5
00A5          ----------------------
00A5          MONITOR-UNTERPROGR.
00A5
00A5 PAT23          =$FFA0
00A5 PRIERR         =$F079
00A5 DE2            =$EC1B
00A5 PINT           =$FOCB
00A5
00A5
00A5          ----------------------
00A5            MONITOR - RAM
00A5
00A5 IDIR           =$A474
00A5 IOFFS          =$A476
00A5 IDOT           =$A477
00A5 IOUTL          =$A478
00A5 IOUTU          =$A479
00A5 IBITL          =$A47A
00A5 IBITU          =$A47B
00A5 DRB            =$A800
00A5 DRAH           =$A801
00A5 T2L            =$A808
00A5 T2H            =$A809
00A5 PCR            =$A80C
00A5 IFR            =$A80D
00A5
00A5
00A5                *=$0E20
0E20          ----------------------
0E20              START
0E20
0E20          ----------------------
0E20
0E20
0E20          ;DAS PROGRAMM GIBT DIE TABELLE (1300 BYTE)
0E20          ;FOLGENDERMASSEN AUS :
0E20          ;    JEWEILS EIN BYTE BESTIMMT 8 SENKRECHT
0E20          ;    UEBEREINANDERLIEGENDE PUNKTE.
0E20          ;    WENN EIN BIT =1 WIRD EIN PUNKT GEDRUCKT.
0E20          ;FORMAT :   100 * 100 PUNKTE
0E20          ;X-KOORDINATE : HORIZONTAL
0E20          ;Y-KOORDINATE : VERTIKAL
0E20          ;DIE DRUCKGESCHWINDIGKEIT MUSS EINMAL
0E20          ;EINGESTELLT WERDEN (SIEHE TEXT)
0E20          ;================================
0E20          ;ZUM ABLAUF DES DRUCKVORGANGES BITTE AUCH DAS
0E20          ;HANDBUCH UND DIE MONITORZEILEN 2367-2609
0E20          ;BEACHTEN.
0E20          ;NACH AUSGABE EINES BILDES MUSS EINE LEER-
0E20          ;ZEILE AUSGEGEBEN WERDEN,SONST IST DIE
0E20          ;FOLGENDE ZEILE DURCHEINANDER.
0E20
0E20
0E20 START    A9FF    LDA #$FF       ;INITIAL. SYNCHR.
```



```
X : -6 -6
Y : -6 -6
Z :  0 -1
 2040 Z=1/SQR(1+X*X+
Y*Y
```

```
0E22          8D74A4 STA IDIR        ;+ RICHTUNG
0E25          A963   LDA #99          ;X-KOORDINATE
0E27          8D76A4 STA IOFFS
0E2A          A963   LDA #99          ;Y-KOORDINATE
0E2C          8D77A4 STA IDOT
0E2F          20DC0E JSR PUNKT       ;INIT./MASKENBILDUNG
0E32          A9C1   LDA #$C1
0E34          8D0CA8 STA PCR
0E37          20A0FF JSR PAT23       ;START/WAIT TI.
0E3A          D008   BNE H1
0E3C          20A0FF JSR PAT23       ;START/WAIT TI.
0E3F          D003   BNE H1
0E41          4C79F0 JMP PRIERR
0E44 H1       A000   LDY #00         ;NEW LINE
0E46          8C01A8 STY DRAH
0E49          A902   LDA #02
0E4B H2       2C0DA8 BIT IFR
0E4E          F0FB   BEQ H2
0E50 H0       AD78A4 LDA IOUTL       ;NEW POINTS
0E53          8D01A8 STA DRAH        ;CLR CA1-INTER. FLAG
0E56          AD00A8 LDA DRB
0E59          0D79A4 ORA IOUTU
0E5C          8D00A8 STA DRB
0E5F          A9A4   LDA #$A4        ;STARTE TIMER
0E61          8D08A8 STA T2L         ;FUER 2.212 MS
0E64          A908   LDA #$08
0E66          8D09A8 STA T2H
0E69          AE76A4 LDX IOFFS       ;BEI BEDARF RICHTUNGSWECHS.
0E6C          2C74A4 BIT IDIR        ;RICHTUNG +: VERZWEIGEN
0E6F          100B   BPL H3
0E71          CA     DEX             ;RICHTUNGSWECHSEL LINKS
0E72          E059   CPX #89
0E74          D00F   BNE H4
0E76          EE74A4 INC IDIR        ;RICHTUNG +
0E79          E8     INX
0E7A          D009   BNE H4
0E7C H3       E8     INX             ;RICHTUNGSWECHSEL RECHTS
0E7D          E064   CPX #100
0E7F          D004   BNE H4
0E81          CE74A4 DEC IDIR        ;RICHTUNG -
0E84          CA     DEX
0E85 H4       8E76A4 STX IOFFS
0E88          C8     INY
0E89          C00A   CPY #10
0E8B          F004   BEQ H5
0E8D          C014   CPY #20
0E8F          D00A   BNE H6
0E91 H5       CE77A4 DEC IDOT
0E94          AD77A4 LDA IDOT
0E97          C9FF   CMP #$FF
0E99          F020   BEQ H7
0E9B H6       20DC0E JSR PUNKT       ;EINE ZEILE AUSGEBEN
0E9E          201BEC JSR $EC1B       ;WAIT TILL TIME OUT
0EA1          20CE0E JSR H8
0EA4          C014   CPY #20
0EA6          F010   BEQ H9
0EA8          A980   LDA #$80        ;STARTE TIMER
0EAA          8D08A8 STA T2L         ;FUER 2.432 MS
```

```
0EAD            A909    LDA #$09
0EAF            8D09A8  STA T2H
0EB2            201BEC  JSR DE2
0EB5            4C500E  JMP H0
0EB8 H9         4C440E  JMP H1
0EBB H7         20CE0E  JSR H8
0EBE            A20A    LDX #10         ;FUER DIE 10 DRUCKSPALTEN
0EC0 H11        20A0FF  JSR PAT23       ;START/WAIT TI.
0EC3            CA      DEX
0EC4            10FA    BPL H11
0EC6            A9E1    LDA #$E1
0EC8            8D0CA8  STA PCR
0ECB            4CCBF0  JMP PINT        ;INIT VAR. FUER PRINT.
0ECE H8         A900    LDA #00
0ED0            8D01A8  STA DRAH        ;THERMAL EL. OFF
0ED3            AD00A8  LDA DRB
0ED6            29FC    AND #$FC        ;DONT CHANGE TAPE CONTR.
0ED8            8D00A8  STA DRB
0EDB            60      RTS
0EDC PUNKT      98      TYA
0EDD            48      PHA
0EDE            201C0F  JSR SET         ;BESTIMME BYTE UND BITMASKE
0EE1            A900    LDA #00
0EE3            8D7BA4  STA IBITU
0EE6            8D78A4  STA IOUTL
0EE9            8D79A4  STA IOUTU
0EEC            A901    LDA #01
0EEE            8D7AA4  STA IBITL
0EF1            AC76A4  LDY IOFFS       ;HOLE X-KOORD.
0EF4 PUNKT1     B1A0    LDA (BYTE),Y    ;HOLEN DES BESTIMMT. BYTES
0EF6            24A3    BIT YBIT        ;UEBERPR. MIT MASKE
0EF8            F012    BEQ NAUSG
0EFA AUSG       AD78A4  LDA IOUTL       ;AUSGABE DES PUNKTES
0EFD            0D7AA4  ORA IBITL       ;WENN BIT = 1
0F00            8D78A4  STA IOUTL
0F03            AD79A4  LDA IOUTU
0F06            0D7BA4  ORA IBITU
0F09            8D79A4  STA IOUTU
0F0C NAUSG      0E7AA4  ASL IBITL       ;KEINE AUSGABE
0F0F            2E7BA4  ROL IBITU       ;WENN BIT=0
0F12            98      TYA
0F13            38      SEC
0F14            E90A    SBC #10         ;Y = Y - 10
0F16            A8      TAY             ;NAECHSTE DRUCKKOPFSPALTE
0F17            B0DB    BCS PUNKT1      ;UEBERPRUEFEN OB ALLE
0F19            68      PLA             ;SPALTEN AUSGEGEBEN
0F1A            A8      TAY
0F1B            60      RTS
0F1C
0F1C SET        A909    LDA #TABAN      ;WANDELT X- UND Y-WERT IN
0F1E            85A1    STA BYTE+1      ;EINEN TABELLENWERT UM
0F20            AD77A4  LDA IDOT
0F23            4A      LSR A
0F24            4A      LSR A
0F25            4A      LSR A
0F26            85A0    STA BYTE        ;BYTE=INT(IDOT/8)
0F28            0A      ASL A
0F29            0A      ASL A
```

```
0F2A        0A      ASL A
0F2B        85A4    STA YBIT1
0F2D        AD77A4  LDA IDOT
0F30        38      SEC
0F31        E5A4    SBC YBIT1
0F33        85A4    STA YBIT1       ;BITZAHL IN YBIT1
0F35        A900    LDA #00
0F37        85A2    STA YBYTE
0F39 K3     A5A0    LDA BYTE        ;BERECHNUNG D. ANGESPR.
0F3B        F010    BEQ K2          ;GRUNDBYTES FUER
0F3D        A5A2    LDA YBYTE       ;DIE AKTUELLE ZEILE
0F3F        18      CLC             ;YE (YWERT/8)UM 100 BYTE
0F40        6964    ADC #100        ;HOEHER
0F42        85A2    STA YBYTE
0F44        9002    BCC K9
0F46        E6A1    INC BYTE+1
0F48 K9     C6A0    DEC BYTE
0F4A        4C390F  JMP K3
0F4D K2     A901    LDA #01         ;UMWANDELN DER BITZAHL
0F4F        85A3    STA YBIT        ;IN EINE MASKE
0F51 K1     A5A4    LDA YBIT1
0F53        F007    BEQ K4
0F55        06A3    ASL YBIT
0F57        C6A4    DEC YBIT1
0F59        4C510F  JMP K1
0F5C K4     A5A2    LDA YBYTE
0F5E        85A0    STA BYTE
0F60        60      RTS
```

#

Klaus Flesch, 7814 Breisach

# CBM-FORTH

Seit Anfang 1982 können nun auch die Besitzer eines CBM die Vorteile, die die Programmiersprache Forth bietet, nutzen. Die Firma Lowinski in Freiburg hat nämlich einen Forth-Compiler in ihr Programm aufgenommen. Der Autor hatte Gelegenheit, dieses Forth ausgiebig zu testen. Es ist auf dem Standard FIG-Forth, das ja auch im AIM 65 implementiert ist, aufgebaut. Da Forth bekanntlich eine erweiterbare Sprache ist, wurden zum Standard noch einige systemspezifische Erweiterungen hinzugefügt.

Diese Erweiterungen sind zum Beispiel die Möglichkeit, eine Commodore Diskettenstation an das Forth anzukoppeln. Gleichzeitig ist ein Editor und ein Assembler vorhanden. Im Editor können Programme (das heißt: Forth-Worte) entwickelt, verbessert und für den späteren Gebrauch abgespeichert werden. Im Assembler können Forth-Worte in der Maschinensprache des 6502 geschrieben werden. Dieser Assembler ist ebenso wie das Forth strukturiert gestaltet, d.h. die Programmentwicklung findet weitestgehend ohne Labels statt. Stattdessen gibt es Statements wie BEGIN - END oder REPEAT - UNTIL oder IF - THEN.Diese Ausdrücke sind PASCALgewohnten Programmierern sicherlich ein Begriff. Da sowohl im Forth wie auch im Assembler strukturiert gearbeitet wird, können Assemblerdefinitionen zuerst in Forth geschrieben und getestet werden, um dann später mit der gleichen Struktur in Maschinensprache übersetzt zu werden. Der Assembler arbeitet, wie auch das Forth, mit der etwas gewöhnungsbedürftigen umgekehrten polnischen Notation (UPN). Dazu zwei Beispiele:

| | |
|---|---|
| DATA LDA | LDA DATA |
| DATA )Y STA, | STA (DATA),Y |
| Forth | Normal |

Der Assembler enthält auch einige Macros, um 16-Bit-Zahlen verarbeiten zu können. Diese Macros sind auch durch eigene Befehle ergänzbar.

Der mitgelieferte Editor ist zeilenorientiert. Er verarbeitet sog. Screens, die 1 K-Byte groß sind. Dieses K-Byte ist in 16 Zeilen á 64 Zeichen aufgeteilt. Der Editor ermöglicht es nun,, nach Tippen des Wortes Editor Zeichen einzufügen, den Screen zu listen, Zeilen einzugeben und weitere Textmodifikationen auszuführen. Der Editorwortschatz wird deshalb erst nach dem Editorkommando aktiv. Er ist ein sog. Erweiterungsvokabular. Der Assembler ist ebenfalls unter einem neuen Vokabular zu erreichen. Die Aufteilung der Vokabularies ermöglicht es, Teile des Forth-Wortschatzes aus der Suche herauszunehmen.

Dies alles ist im Standardpaket enthalten. Gleichzeitig werden jedoch noch zwei weitere Bereiche des Forth optional erweitert. Die Floating-Point-Erweiterung ergänzt die normal nur vorhandene Integerrechenfähigkeit. Mit ihm kann alle aus dem BASIC bekannte Realarithmetik betrieben werden. Gleichzeitig können diese Funktionen natürlich zu neuen Funktionen zusammengestellt werden. Denkbar ist z.B. ein kleines Paket, das Imaginärzahlen verarbeitet.

Ein weiteres Ergänzungspaket umfaßt die Stringfunktionen, mit ihm kann man Zeichenketten verwenden. Möglich sind dabei alle Funktionen wie RIGHT$, LEFT$, MID$, CHR$. Zusätzlich sind jedoch auch Stringpointertypen verhanden, die theoretisch 32 K-Byte umfassen können. So ist es zum Beispiel möglich, einen String SCREEN zu definieren, der an der Adresse 32768 beginnt und eine Länge von 2000 Zeichen hat. Diesem String können nun andere Strings der gleichen Länge zugewiesen werden und natürlich auch umgekehrt. Die oben genannten Funktionen sind natürlich auch auf diesen etwas großen String anwendbar. Als Erweiterung können Strings um ein Zeichen nach links oder rechts verschoben werden. Man kann auch testen, ob ein Teilstring in einem anderen vorhanden ist. Dabei können auch mehrfache Übereinstimmungen gefunden werden. Weitere Möglichkeiten sind das Ersetzen eines Teilstrings durch einen anderen.

In der nachfolgenden Auslistung sehen wir den Befehlssatz eines voll ausgerüsteten CBM-Forth. Die invertierten Zeichen sind dabei Wörter, die auch in neuen Definitionen sofort ausgeführt und nicht kompiliert werden. ASSEMBLER, EDITOR, DISK, ?????? sind die Schlüsselwörter für verschiedene Vacabularies. Unter DISK befinden sich verschiedene Operationen, um Datenfiles zu eröffnen, BASIC-Files zu lesen und bei dem DOS 2.1 (4040 und 8050) relative Files zu erzeugen und zu lesen. Unter ?????? sind die Fehlermeldungen, die normalerweise auf der Diskette vorhanden sind, im RAM vorhanden.

VLIST des Anwenderprogrammes BREAKFORTH und ab TASK1 des FORTH

```
4CC7 BREAKFORTH          4CA4 DELAY      4C7A GAMECHK    4C4C BALL
4C1B BALLCHK   4BBD CLR  4B67 PCHK       4B4A BREAKC     4B3D -2
4B34 -1        4AB7 YCHK 4A6C XCHK       4A51 BOP        49FB PADDLE
49DC PSET      49BF PCLR 49B4 PBEG       489E INIT       488D LINE
4882 BEST      4877 XDIR 486C YDIR       4861 PPOS       4856 YPOS
484B XPOS      4840 SCORE 4834 SPVAR     4828 SPEED      4816 CLS
4800 CTE       47D6 PTC  47B6 :CASE      4798 RND        4778 RANDOMIZE

4768 RAND      4707 D?   468A DCLR       460B DSET       45EC ADDR
45E1 TASK1     45C5 PRINTOFF 4582 PRINTER 455E PPREL     4552 PSEC
4547 PRINTER#  4538 XEMIT 437F [illegible] 435E TST      433C :?
426F PNAME     423B [illegible]          4203 [illegible] 41DC [illegible]
41B5 TRANS     4187 CHF  4173 FNUMBER    4112 DFLOAT     4105 DSAV
40E4 DINTEGER  40B7 INTEGER 4094 (INTEG) 407C FLOAT      4055 (FLOAT)
4031 FPARRAY   4011 F/   4004 SQR(       3FF5 FPDROP     3FE2 F<>
3FD4 F=        3FC3 F<   3FB4 F>        3F90 FP?        3F7B (<>)
3F5E FPCON     3F38 Ft   3F11 (EXP)     3F03 SGN(       3EF6 RND(
3EE9 ATN(      3EDC TAN( 3ECF SIN(       3EC2 COS(       3EB5 EXP(
3EA8 INT(      3E9B ABS( 3E8E SQR(       3E81 LOG(       3E62 10P
3E56 F/        3E4B F-   3E40 F+        3E35 F*         3E0C 20P
3DE7 2PF       3DDD (N)  3DD3 JMPVEC     3DC5 (JMPV)     3DB4 FP.
3D7C (FP.)     3D51 S>P  3D2D [illegible] 3CFD (FPLIT)   3CCD FP!
```

```
3CA1 (FP)       3C96 FPT        3C80 FPDUP      3C68 FPROT      3C4C FPSWAP
3C3A ACC3       3C2A ACC2       3C1A ACC1       3C05 VAR%       3BDD FP@
3BC2 FP!        3BB4 SINGLEINP                  3B82 DOUBLEINP
3B64 GETSTR$    3ADE NUMBER$    3A96 FDIGIT     3A55 NOSPACE    39F5 ASCII$
39A4 $+         3973 REPLACE$   393E [illegible]  3925 ("!)       390C "!"
38DC MATCHES$   38C9 P1         3890 NEXT-MATCH                 3847 INPUT$
3836 (INPUT)    37FA $!         3789 MATCH$     377C M3         375F [illegible]
374A ($<-)      373B M2         372E M1         36C3 EQ$        3671 LEFT$
3655 RIGHT$     3640 CHR$       3635 CHR        35F3 LROT$      35A9 RROT$
3543 MID$       3526 $POINTER   3505 $STRING    34F1 CLEAR-STACK
34B3 POP        347B PUSH       345E STACK      33B4 VLIST      33A0 KORROUT
3388 PRINTOFF   3365 PRINTER    3340 PPREL      3334 PRBOTH     3327 XEMIT
32E4 .S         32D1 PICK       32C4 V          32BA P          32B0 0
32A6 L          328A GRANGE     2A23 CODE       2A12 [illegible]
269A [illegible]  2667 LINE       2648 TEXT       262F [illegible]  261B DISKLOAD
25FC SD1        25E2 SD0        25CE DISKVER    25BE DISKSAVE   2593 DISKERR
2581 UNTALK     2570 DISKCHAR   2557 SECADRS    2545 TALK       2530 SETDISK
2510 DSELECT    24C8 DATAINT    24B6 DFNA$      24A0 DISKCOM    2161 [illegible]
2156 DTYP       210D SYSSIZE    20F9 CASS2      20E7 CASS1      20CD VERIFY
208E SYSSAVE    2068 CASSAVE    204A MEMSET     201D CASSLOAD   2004 SETDEVICE#

1FC7 NAMIEEE    1F69 NAMSET     1F34 STR$       1F1F SCR#       1ED2 $VARIABLE

1E8A DIM()      1E4A [illegible]  1E3F DEVICE#    1E0D SYS        1DC5 DUMP
1D6B VLIST      1D59 -RVS       1D48 RVS        1CB6 MON        1C5A TRIAD
1C25 INDEX      1BE5 LIST       1BD6 ?          1BCA .          1BBA .R
1BAB D.         1B88 D.R        1B70 #S         1B48 #          1B33 SIGN
1B1A #>         1B0B <#         1AEE SPACES     1ADD [illegible]  1ABB [illegible]
1AA4 [illegible]  1A8B [illegible]  1A74 [illegible]  1A66 [illegible]  1A52 [illegible]
1A3C [illegible]  1A26 [illegible]  1A13 [illegible]  1A08 [illegible]  19ED [illegible]
19DB [illegible]  19C9 BACK       195E FORGET     1945 [illegible]  193A R/W
18C6 [illegible]  188E LOAD       1841 MESSAGE    182B CASFX      180C CASSA
17E1 CASLO      17CE BYE        17A1 MTYPE      1761 PERROR     173A ARRAY
1728 .LINE      1706 (LINE)     16D8 FLUSH      1678 BLOCK      1630 BUFFER
161B DR1        160B DR0        15F7 EMPTY-BUFFERS              15D1 UPDATE
15A8 +BUF       159D PREV       1592 USE        1576 M/MOD      1564 *./
1553 *./MOD     1543 MOD        1533 /          1523 /MOD       1514 *
14EE M/         14D1 M*         14BA MAX        14A2 MIN        1494 DABS
1485 ABS        1473 D+-        1461 +-         1452 S->D       1411 COLD
13DD ABORT      13AE QUIT       139E [illegible]  138E DEFINITIONS
1376 [illegible]  1346 VOCABULARY                 132E IMMEDIATE
12E0 INTERPRET                  12B9 ?STACK     129E [illegible]  1281 [illegible]
1265 [illegible]                1201 CREATE     11CD ID.        1198 ERROR
118A (ABORT)    1158 -FIND      1102 NUMBER     10B7 (NUMBER)   1088 UPPER
103B WORD       1029 PAD        1011 HOLD       1002 BLANKS      FF1 ERASE
 FD1 FILL        F92 [illegible]  F7B QUERY      F0A EXPECT      ED8 [illegible]
 EBF (.") [illegible]  E8C -TRAILING [illegible]    E5E TYPE [illegible]    E4B COUNT [illegible]
 E19 DOES>      E09 <BUILDS     DEF [illegible]  DD9 (;CODE)    DC4 DECIMAL
 DAF HEX        D9E SMUDGE      D8A ]           D7C [illegible]  D66 COMPILE
 D4D ?LOADING   D31 ?CSP        D1F ?PAIRS      D09 ?EXEC       CF2 ?COMP
 CD8 ?ERROR     CC5 !CSP        CB1 PFA         C9C NFA         C8E CFA
 C7F LFA        C6F LATEST      C4B TRAVERSE    C34 -DUP        C25 SPACE
 C11 ROT        C03 >           BE6 <           BDA U<          BCD =
 BC1 -          BB1 C,          BA0 ,           B94 ALLOT       B84 HERE
 B75 2+         B68 1+          B60 HLD         B57 R#          B4F CSP
 B46 FLD        B3D DPL         B34 BASE        B2A STATE       B1F CURRENT
 B12 CONTEXT    B05 OFFSET      AF9 SCR         AF0 OUT         AE7 IN
 ADF BLK        AD6 VOC-LINK    AC8 DP          AC0 FENCE       AB5 WARNING
 AA8 WIDTH      A9D TIB         A8D +ORIGIN     A7F B/SCR       A73 B/BUF
 A67 LIMIT      A5B FIRST       A4F C/L         A45 BL          A3C 3
 A34 2          A2C 1           A24 0           A0B USER        9F2 VARIABLE
 9D2 CONSTANT   9B9 [illegible]  98C [illegible]  97F C!          967 !
 958 C@         943 @           934 TOGGLE      913 +!          904 DUP
 8EC SWAP       8E3 DROP        8D2 OVER        8BB DMINUS      8A2 MINUS
 87C D+         863 +           852 0<          83D 0=          827 R
 816 R>         804 >R          7E9 LEAVE       7D6 ;S          7C2 RP!
 7B2 SP!        7A3 SP@         78D XOR         777 OR          762 AND
 725 U/         6ED U*          6C8 CMOVE       6BE CR          6B7 ?TERMINAL

 6A9 KEY        6A1 EMIT        662 ENCLOSE     5FC (FIND)      5CB DIGIT
 5C1 I          59F (DO)        568 (+LOOP)     538 (LOOP)      517 0BRANCH
 4ED BRANCH     4A7 EXECUTE     484 CLIT        42E LIT
```

Um einen Einblick in das Programmieren mit Forth zu geben, wurde ein Programm adaptiert, das viele Leser sicher aus BASIC-Versionen kennen. Es handelt sich um BREAKTHROUGH, hier BREAKFORTH genannt. Dafür wurde zuerst ein kleines Hilfspaket entwickelt, um in der Y-Achse eine doppelte Auflösung zu erreichen. Die Befehle setzen oder löschen einen Punkt. Sie errechnen sich aus den schon vorhandenen Bildinformationen, welches Zeichen beim Setzen oder Löschen eines Halbpunktes nun in das Bildschirm-RAM gesetzt werden muß. Mit einer CASE-Anweisung wäre dieses einfacher zu lösen gewesen. Diese Anweisung kann jedoch eingeführt werden. Ein Beispiel dazu ist in den Forth Dimensions II/3, Seite 37 beschrieben und erläutert worden. Eine Hilfskonstruktion, die ähnliches ausführt, wird auf Screen 112 definiert.Sie führt danach eine Operation durch, die durch die Zahl auf dem Stack bestimmt wird. Eine 1 führt das erste Wort aus, eine 4 das vierte. Aus diesem Grund müssen später die Zahlen -1 und -2 als Worte in der Directory definiert werden. Gleichzeitig werden noch einige bescheidene Zufallszahlendefinitionen eingeführt. Sie verwenden als Zufallselement den Inhalt von Timer 1 der VIA 6522. Gleichzeitig werden noch Cursor-Bewegungsfunktionen wie PTC (Put Cursor) und CTE (Clear to End of Line) und CLS (Clear Screen) definiert. Dabei sieht man auch, wie einfach Maschinensprache zur Definition verwendet werden kann. Zur Arbeitsweise von CTE ist zu sagen, daß es sich die Spalteninformation aus der Speicherzelle $C6 in das Y-Register lädt und dann die Zeile bis zur nächsten Spalte mit Leerzeichen löscht.

Doch nun zur Beschreibung des Programmes:

Es lädt sich zuerst die beiden besprochenen Module ein und initialisiert den Zufallsgenerator. Danach werden einige Variablen definiert. Die Funktion LINE löscht eine Zeile. INIT fragt die vom Spieler einzugebenden Informationen ab und zeichnet das Grundspielfeld mit der Umrandung und dem inneren Block. Das wird durch das Statement 33088 400 224 FILL erreicht. PCLR löscht den Paddle und PSET setzt ihn wieder. Das Befehlswort PADDLE überprüft den Tastatureingabecode und bewegt den Paddle beim Tasten der 1 nach links, jedoch höchstens bis zur Position 2. Mit der Taste 3 wird er nach rechts bewegt. BOP gibt einen Piepslaut aus. XCHK und YCHK prüfen, ob der Ball in X- oder in Y-Position eine Bande berührt hat. Wenn dies der Fall war, wird die X- und Y-Richtung entsprechend geändert. PCHK prüft, ob der Ball unten aus dem Spielfeld gelangt ist. Wenn er dabei auf den Paddle gelangt, wird er reflektiert. Der Reflektionswinkel wird dabei so bestimmt, daß er an den Enden spitz und in der Mitte stumpf ist. Das wird über die Konstruktion :CASE erreicht. Diese wandelt die Eingabe 0 bis 7 in die Ausgabe -2 -1 -1 -1 1 1 1 1 2 um. CLR löscht nach Auftreffen des Balles den bestimmten Abschnitt. An BALLCHK sieht man nun die modulare Weise, in der neue anwendungsbezogene Wörter entstehen. Es berechnet zuerst die neue X- und Y-Position des Balles und führt dann die vorher definierten Befehle XCHK, YCHK und PCHK aus. Es fragt dann ab, ob an dieser neuen Position ein Punkt vorhanden ist. Wenn ja, wird dieser gelöscht. Da der Befehl PCHK ein boolsches Flag auf den Stack legt, bleibt dieser auch vorhanden. BALL befindet sich noch ein Modul höher. Es löscht zuerst die alte Ballposition, führt BALLCHK aus und setzt den Ball, falls er nicht außerhalb des Feldes geraten ist. GAMECHK läßt läßt bei einem Punktestand von n*1800 Punkten einen neuen inneren Block entstehen. Das Wort DELAY wird nun gebraucht, um dem menschlichen Partner die Möglichkeit einer Rekation zu geben. Dies zeigt auch einen weiteren Vorteil des Forth auf: Es ist nämlich, da es compiliert wird, um einiges schneller als das normale BASIC. Im Endwort BREAKFORTH werden nun alle diese definierten Wörter zusammengefügt. Es enthält die Hauptschleife, die für jeden Ball einmal durchlaufen wird.

Die Forthprogramme sind in der typischen Forth-Struktur, den Screens aufgelistet. – Informationen über das 250 DM kostende Forth-Paket sind bei der Firma Wolfgang Lowinski Computerservice, Gallwitzstr. 10, 7800 Freiburg zu erhalten. Der Autor dieses Beitrages ist gerade dabei, eine Informationssammlung über Forth zusammenzusuchen. Er wäre für Anregungen und Informationen dankbar (Anschrift: Neutorstr. 9, 7814 Breisach).

```
SCR # 111
  0 FORTH DEFINITIONS HEX ( DSET USW. )
  1 : ADDR ( X Y --- ADDR YMOD2 ) 2 /MOD 50 * ROT + 8000 + ;
  2 : DSET ADDR SWAP OVER C@ DUP 62 = IF DROP IF 62 ELSE E0 THEN
  3                          ELSE DUP E2 = IF DROP IF E0 ELSE E2 THEN
  4                               ELSE E0 = IF DROP E0 ELSE
  5                        IF 62 ELSE E2 THEN THEN THEN THEN
  6 SWAP C! ; ( X Y --- )  ( SETZT PUNKT )
  7 : DCLR ADDR SWAP OVER C@ DUP 62 = IF DROP IF 60 ELSE 62 THEN
  8                          ELSE DUP E2 = IF DROP IF E2 ELSE 60 THEN
  9                               ELSE E0 = IF IF E2 ELSE 62 THEN ELSE
 10                       DROP 60 THEN THEN THEN
 11 SWAP C! ; ( X Y --- ) ( LOESCHT PUNKT )
 12 : D? ADDR C@ DUP E0 = IF DROP DROP 1 ELSE DUP 62 = IF DROP IF 1
 13                          ELSE   0 THEN ELSE E2 = IF IF 0 ELSE 1
 14                         THEN ELSE DROP 0 THEN THEN THEN ;
 15 ( X Y --- B TRUE WENN PUNKT GESETZT ) ;S


SCR # 112
  0 FORTH DEFINITIONS ( RANDOMS ETC )
  1 12345 VARIABLE RAND : RANDOMIZE 95135 59460 @ + RAND @ MOD
  2 RAND ! ; : RND 59460 @ RAND +! RAND @ MOD ABS ;
  3
  4 : :CASE <BUILDS !CSP ] SMUDGE DOES> SWAP 2 * + @ EXECUTE ;
  5
  6 HEX : PTC C6 C! DUP D8 C! 50 * 8000 + C4 ! ;
  7 CODE CTE ( CLEARS LINE TO END ) C6 LDY, BL # LDA, BEGIN,
  8 C4 )Y STA, INY, 50 # CPY, 0= UNTIL, NEXT JMP,
  9
 10 FORTH DECIMAL
 11 : CLS 147 EMIT ; ;S
 12
 13
 14
 15


SCR # 113
  0 DECIMAL : TASK1 ; 111 LOAD 112 LOAD RANDOMIZE
  1 0 VARIABLE SPEED 0 VARIABLE SPVAR 0 VARIABLE SCORE
  2 0 VARIABLE XPOS  0 VARIABLE YPOS  2 VARIABLE PPOS
  3 1 VARIABLE YDIR  1 VARIABLE XDIR  0 VARIABLE BEST
  4 : LINE   0 PTC CTE ;
  5 : INIT   CLS 0 LINE ." SPEED ( 1 - 10, 1 IS FASTEST ) "
  6          KEY 48 - 1 MAX 10 MIN 10 * SPEED !
  7          0 LINE ." NUMBER OF BALLS DESIRED " KEY 48 -
  8          CLS 80 0 DO I 3 DSET I 4 DSET LOOP
  9          48 3 DO 0 I DSET 79 I DSET 1 I DSET 78 I DSET LOOP
 10         33088 400 224 FILL 0 SCORE !
 11         0 LINE ." BREAKFORTH IN CBM-FORTH  SCORE: 0     BEST: "
 12         BEST ? 0 60 PTC ." |ALL: " ;
 13 ( ----- #BALLS  )
 14 -->
 15
```

```
SCR # 114
  0 34608 CONSTANT PBEG : PCLR  ( --- ) PPOS @ PBEG + 8 96 FILL ;
  1 : PSET  ( --- )  PPOS @ PBEG + 8 226 FILL ;
  2 : PADDLE  ( --- )
  3   151 C@ 177 = IF PCLR -1 PPOS @ + 2 MAX PPOS ! PSET THEN
  4   151 C@ 179 = IF PCLR  1 PPOS @ + 70 MIN PPOS ! PSET THEN
  5 ;
  6 : BOP ( --- )  6 231 C! 7 EMIT ;
  7 : XCHK ( --- )
  8   XPOS @  2 < IF XDIR @ MINUS XDIR ! 2 XPOS ! BOP THEN
  9   XPOS @ 77 > IF XDIR @ MINUS XDIR ! 77 XPOS ! BOP THEN
 10 ;
 11 : YCHK ( --- )   YPOS @ 5 < IF 1 YDIR ! 5 YPOS !
 12                       1 SPVAR ! BOP THEN
 13   YPOS @ 23 < IF SPVAR @ 4 MIN SPVAR ! THEN
 14   YPOS @ 19 < IF SPVAR @ 3 MIN SPVAR ! THEN
 15   YPOS @ 15 < IF SPVAR @ 2 MIN SPVAR ! THEN ; -->


SCR # 115
  0 -1 CONSTANT -1 -2 CONSTANT -2
  1 :CASE BREAKC ( N --- )  -2 -1 -1 -1 1 1 1 2 ;
  2 : PCHK 0 YPOS @ 46 >  ( --- B WENN BALL AUSER SPIEL )
  3   IF 46 YPOS ! XPOS @ PPOS @ - DUP -1 > OVER 8 < AND
  4      IF -1 YDIR ! BOP
  5      BREAKC XDIR ! ELSE DROP 1+ THEN THEN ;
  6 : CLR XPOS @ 2 - 124 AND 2+ DUP 4 + SWAP DO I YPOS @ DCLR LOOP
  7   YPOS @ 27 - ABS SCORE +! 0 32 PTC SCORE ? BOP
  8   YDIR @ MINUS YDIR ! ;  ( --- )
  9 : BALLCHK ( --- B ) YDIR @ YPOS +! XDIR @ XPOS +! XCHK YCHK PCHK
 10   XPOS @ YPOS @ D? IF CLR THEN ;
 11 : BALL ( B --- B ) XPOS @ YPOS @ DCLR BALLCHK DUP 0=
 12        IF XPOS @ YPOS @ DSET THEN PSET ;
 13  -->
 14
 15


SCR # 116
  0 : GAMECHK ( --- ) SCORE @ 1800 MOD 0= IF 33088 400 224 FILL THEN
  1 ;
  2 : DELAY ( --- ) SPEED @ SPVAR @ * 0 DO LOOP ;
  3
  4 : BREAKFORTH ( --- ) BEGIN INIT 0 PSET DO
  5       2000 SPEED @ / 0 DO DELAY PADDLE LOOP
  6       0 66 PTC I 1+ . 5 SPVAR ! 2 RND 1 = IF 1 ELSE -1 THEN
  7       XDIR ! 1 YDIR ! 72 RND 2 + XPOS ! 29 YPOS !
  8       BEGIN 3 0 DO PADDLE LOOP BALL GAMECHK DELAY END
  9       LOOP SCORE @ BEST @ MAX BEST ! 8 18 PTC
 10       ." _UN |AME AGAIN " 0 158 C! KEY 74 = 0= UNTIL ;
 11 BREAKFORTH ;S
 12
 13
 14
 15

KLAUS FLESCH 25 JANUAR 1982                                        #
```

# Software-Besprechung

**ISAM für den CBM**

*Seit einem Jahr ist beim Autor dieses Artikels ein Datenhaltungsprogramm nach MSAM-Art in Betrieb, das auf der ISAM-Unterstützungssoftware der Fa. Creative Software beruht (s.a. Artikel 'ISAM, ein Dateityp' in diesem Heft). Die ISAM-Routine (2 KBytes in Maschinensprache) ist in erster Line als Softwareunterstüzung für den BASIC-Programmierer gedacht, der sich mit den Problemen der Datenhaltung beschäftigt. Unter dieser Voraussetzung entsprach sie ganz den Erwartungen. - Man erhält die Software auf einer Diskette. Herzstück ist die o.a. ISAM-Routine, die als BASIC-Programm geladen und mit RUN an die Obergrenze des RAM-Speichers geschoben wird. Weiterhin befinden sich auf der Diskette (jeweils in BASIC) ein ISAM-Monitor und ein Adreßprogramm mit bereits eingerichteter Datei. Mit dem ISAM-Monitor können alle Datei-Befehle direkt und im Dialog ausgeführt werden. Er dient zum Testen der ISAM-Befehle und ist hilfreich bei der ISAM-Programmentwicklung sowie beim 'Reparieren' von ISAM-Dateien. Die wichtigsten Befehle aus der Sicht des Anwenders sind:*

| | |
|---|---|
| *CREATE* | *legt eine neue ISAM-Datei an* |
| *OPEN* | *öffnet eine ISAM-Datei zum Lesen und Schreiben* |
| *READ* | *liest Schlüssel und DAten aus einer Datei* |
| *WRITE* | *schreibt Schlüssel und Daten in eine Datei* |
| *READNEXT* | *liest Schlüssel und Daten sequentiell* |
| *DELETE* | *löscht Schlüssel und Daten in einer Datei* |
| *CLOSE* | *schließt eine ISAM-Datei* |

*Die Befehle sind als Einsprungsadressen in die ISAM-Routine aus BASIC definiert. Die Parameterübergabe erfolgt vor bzw. nach dem ISAM-SYS-Befehl mit einem analogen BASIC-Befehl. Wenn man die SYS-Adressen als Variable definiert (dies ist jedoch nicht bei allen Befehlen möglich), und die Variablennamen entsprechend ihrer Funktion mnemotechnisch abkürzt (z.B. SYS(OPN); SYS(RNK)), dann läßt sich ISAM ganz gut in BASIC-Programme einbinden.*

*Das ISAM-Handbuch, in leicht verständlichem Englisch geschrieben, ist mit 53 Seiten relativ umfangreich und läßt aus der Sicht des Anwenders keine Fragen offen. - Fehler wurden während des einjährigen Betriebes nicht festgestellt. Auch erscheint der Kaufpreis von rund 100 Dollars nicht unangemessen. ISAM gibt es in verschiedenen Ausführungen auf die jeweilige CBM-Rechner-Floppy-Kombination bezogen. Alles in allem bietet diese Software die Grundlage für eine sehr komfortable Datenhaltung, der nur durch die physikalischen Gegebenheiten der Diskette Grenzen gesetzt sind.*

*Technische Daten*

| | |
|---|---|
| *ISAM-Routine* | *2 KBytes hex 7000-77FF bei der 32 KB-Version* |
| *ISAM-Dateipuffer* | *2 KBytes hex 7800-7FFF* |
| *Anzahl der gleichzeitig offenen ISAM-Dateien: max. 5* | |
| *Zugriffszeit* | *liegt in der Größenordnung des Direktzugriffes bei der Floppy* |
| *Key* | *max. 24 Zeichen (Wertebereich hex 20 bis 5F)* |
| *Datensatz (Key + Record): max. 247 Zeichen (Wertebereich der Record-Zeichen hex 00 bis FF)* | |
| *Indexblöcke* | *werden nach einem modifizierten Prinzip des binären Suchbaumes angelegt* |
| *Preis ohne Versand: 99,95 Dollars* | |
| *Hersteller* | *Creative Software, PO BOX 4030, Mountain View, CA 94040, USA.* |

*Wolfgang Seer*

**POWER Rom PAC - ein weiteres Werkzeug für den Programmierer der Commodore-Rechner.** *Entwickelt wurde es von Brad Templeton, Professional Software Inc., USA.*

*POWER erweitert den vorhandenen BASIC-Editor, dient somit der Programmentwicklung und nicht dem Bereitstellen zusätzlicher Befehle. Es ist in dreifacher Hinsicht bemerkenswert:*

*Die Default-Parameter bei den Befehlen wie auch die Befehlsoptionen selbst sind praxisorientiert und ausgefeilt.*

*Im Handbuch werden eine Reihe nützlicher POWER-Routinen erklärt und die Einsprungsadressen für die Mitbenutzung durch den Assembler-Programmierer angegeben.*

*Ferner werden Hinweise für die Implementierung eigener Editor-Befehle gegeben.*

*Wie bei anderer Software ähnlicher Art gilt auch hier: Nicht die Anzahl der Befehle macht's, sondern deren Leistungsfähigkeit und Nützlichkeit ist entscheidend. POWER kann zu jeder Zeit ein- und ausgeschaltet werden, so daß die CHARGET-Routine in den Urzustand zurückgesetzt werden kann. Neben diesem General-Ein-Aus können eine Reihe von Befehlsmodi aktiviert und desaktiviert werden.*

*Nach dem 'Anspringen' von POWER (die Charget-Routine zeigt danach auf POWER) werden die Cursor-Tasten mit einer sehr schnellen Repeatfunktion versehen. Als ebenfalls sehr angenehm wird vom Autor das Bildschirmscrolling für BASIC-Programme empfunden. Nur durch einen Druck auf die Cursor-Taste läuft das Programm vorwärts oder rückwärts über den Bildschirm. Automatische Zeilennumerierung mit oder ohne Parameter (AUTO) sowie das Löschen von Zeilen (DELETE) ist eine Selbstverständlichkeit. AUTO ohne Zeilennummer positioniert sich automatisch auf die letzte Zeilennummer + Default-Increment 10. - Nicht ganz so selbstverständlich ist bei der Neunumerierung (RENUMBER) die Möglichkeit des teilweisen Neunumerierens, wobei die neue Startadresse beliebig gewählt werden kann. Einleuchtende Einschränkung: Der neu zu numerierende Block darf mit seinen Anfangs- und Endnummern nicht mit den Teilen des Programmes kollidieren, die unverändert bleiben sollen.*

*Besonders vielfältig in ihren Möglichkeiten und Optionen sind die FIND- und REPLACE-Funktionen. Durch eine Reihe von Steuerzeichen (Meta-Characters), die auch ein- und ausgeschaltet werden können, lassen sich ähnlich wie beim 'Pattern Matching' des CBM-DOS Suchstringmasken bilden. Strings, Text nach REM-Befehlen und BASIC-codetext lassen sich unabhängig voneinander durchsuchen. Mit den Befehl SELECT KEYWORD wird jeder Taste ein vorgegebenes BASIC-Keyword zugeordnet. Im SHIFT-Mode kann dann bei Tastendruck das vollständige Keyword ausgegeben werden. Andererseits kann man aber auch jeder Taste bis zu 80 Zeichen Text einschließlich Anführungszeichen zuordnen. Sie werden als REM-Zeilen besonderer Art im Programmtext abgelegt, einer Taste zugeordnet und im SHIFT-Mode ausgegeben. Dies geht soweit (Option), daß man durch Tastendruck ein internes Unterprogramm unmittelbar ausführen lassen kann.*

*Beim TRACE sind mehrere Optionen möglich. Sie reichen von der Angabe der Zeilennummer über komplettes BASIC-Zeilen-List bis hin zur im Augenblick behandelten Variablen mit Inhaltsangabe. Durch Niederdrücken entsprechender Tasten kann beim TRACE zwischen Dauer-Trace, Single-Step und Unterdrücken des Trace-Outputs gewählt werden. - Ein DUMP zeigt alle Variablen (ohne Feldvariable) mit ihrem Inhalt. Auch Stringvariable werden so dargestellt, so daß sie ohne Aufwand in Variablennamen oder Inhalt geändert werden können. - Beim Execute-Befehl (XEC) wird von der Tastatur als Input-Device auf das im vorangehenden Open-Befehl genannte umgesteuert (z.B. Floppy). Wie bei der Tastatur werden auch hier alle eingehenden Zeichen auf dem Bildschirm ausgegeben und jeder Zeile ein Carriage Return zugefügt. Hauptanwendung dieses Befehls ist das Verschmelzen (Merge) zweier BASIC-Programme. Das einzumischende Programm sollte sich dann als sequentielles Datenfile im ASCII-Mode (keine Tokens) auf der Diskette befinden. Mit dem XEC-Befehl erfolgt ein Einmischen des Files über den Bildschirm in der Art 'What you see is what you get'.*

*Bei aktiviertem POWER werden Fehler zur RUN-Zeit über den Befehl WHY mit aufgesetztem Cursor angezeigt. - Soweit die Beschreibung der meisten und sicher wichtigsten Befehle. - Das Handbuch in deutscher Übersetzung ist umfangreich und führt in humorvoller Weise in die Anwendungen von POWER ein. Am Schluß findet man eine Reihe von Einsprungadressen in POWER-Routinen mit Erläuterungen. Ferner ermuntert das Handbuch jeden, eigene individuelle Befehlsroutinen unter Benutzung von POWER zu erstellen und die Befehlstabelle dazu an POWER anzuhängen. Neben anderen ist hierfür ein USER-Vector im 2. Cassettenpuffer vorgesehen.*

*Technische Daten*

| | |
|---|---|
| *Software* | *4 KBytes im ROM (hex 9000-9FFF)* |
| *Preis* | *DM 195,-- (unverb.)* |
| *Vertrieb* | *VIP Software GmbH, 5100 Aachen* |

*Wolfgang Seer*

# Editorial

*Die zahlreichen in diesem Heft enthaltenen Besprechungen geben ihm einen Schwerpunkt in der qualitativen Aussage, während in anderen Heften zumeist Lösungsvorschläge für die Programmierung enthalten waren. Ein so reichhaltiger Bewertungsteil ist gar nicht so einfach zusammenzustellen, denn er setzt voraus, daß die Geräte oder die Software beschafft und eingehend ausprobiert wurden. Die Autoren bilden sich erst danach ein eigenes von etwaiger Lieferantenwerbung unabhängiges Urteil. Diese Art der Berichterstattung ist im Blätterwald durchaus nicht selbverständlich, wie man leider immer einmal wieder feststellen muß. - Bewertungen sind notwendig, um im Wettbewerb des Angebotes auf gute und weniger gute Leistungen hinzuweisen, und zwar zum Vorteil des eventuellen Käufers, der damit die Kriterien für seine Auswahl findet. Es wäre daher zu begrüßen, wenn sich auch künftig viele Autoren für die Besprechung von Hard- und Software fänden, denn ein großer Überblick läßt sich nur im gemeinsamen Bemühen erzielen.*

*Mit diesem Heft kommt auch ein Erhebungsbogen für die bereits angekündigte* **Leserumfrage** *zum Versand. Der Herausgeber bittet alle Leser höflich um ihre Mitwirkung und um möglichst umgehende Absendung ihrer Antworten als Brief.*

*Es geht darum, die künftigen mehrheitlichen und auch minderheitlichen Informationsinteressen der Leser besser kennenzulernen, als das im Zusammenhang mit Telefonaten und gelegentlichen brieflichen Äußerungen möglich ist. Die große Lesertreue seit Anfang an ist sicher ein gutes Indiz, daß in der Aufbauphase, als es nur wenige andere Veröffentlichungen gab, viel nützliche Information vermittelt werden konnte. Es sind aber sicher auch Lücken geblieben, die es zu schließen gilt. Und andererseits ist das Angebot der Computermöglichkeiten, Sprachen und Betriebssysteme für alle Systemfamilien fast unübersehbar groß geworden, so daß es gilt, die möglichen Trends zu erkennen. Um die Öffnung dieser Zeitschrift für weitere Systeme anzuzeigen (der 6809 wurde bereits etwas aufgebaut), ist beabsichtigt, sie sehr bald auf einfach 'MICRO MAG' umzubenennen, wie sie ja meistenteils sowieso schon genannt wird.*

*Auch in die laufende Ausgabe konnten nicht alle vorliegenden Arbeiten aufgenommen werden.Die Autoren werden noch um etwas Geduld gebeten. Wir haben demnächst: Disassembler für 6809, in BASIC geschrieben, und einen in 6809 Assembler mit umfangreichen Erläuterungen zur Programmierungstechnik. Angesagt ist ein BASIC-Compactor und aus erfahrener Hand ein Bericht zur Programmierung mit PL/65 u.a.m.. Gerne werden künftig auch Beispiele und Hinweise für den VC-20 entgegengenommen, ebenso für den AIM 65/40.*

OOOΔΔΔOOOΔΔΔOOOΔΔΔOOOΔΔΔOOOΔΔΔOOOΔΔΔOOOΔΔΔOOOΔΔΔOOOΔΔΔOOO

**Anzeigen**

**Suche FOCAL-Interpreter für CBM.** H. Brettin, Tel. 030-685 53 44 (priv.) oder 030-218 6616(die)

**CBM-2/3/4/8001 Speicherbelegung,** 630 Adr. 20 DM * Beschreibung von ROM-Routinen für reelle Arithmetik, Tape- und IEEE-Bus-I/O, Parameterübergaben an Maschinenprogramme etc. 25 DM. * Zus. 73 Seiten A4 für 35 DM * Katalog kostenlos * H. J. Koch Liegnitzer Str. 8, 3008 Garbsen 8.

**! AIM 65 PASCAL taugt nichts !**
**ohne deutsches Handbuch ....**

Umfang 100 Seiten, Plastikeinband. Buchpreis DM 26,-- (tw-Anrechnung). ROMs haben wir natürlich auch. **6502-Platine aus MC 2/82** DM 73,--. Regge, 2800 Bremen, Fesenfeld 57, Telefon 0421-71 114 (auch abends).

Kleinanzeigen kosten DM 10,- für 2 Zeilen. Betrag bitte bei Bestellung überweisen oder in Form von Briefmarken beifügen.

# Aus der Branche - Produkte

**Rockwell International** erhielt von National Panasonic in Japan einen Großauftrag für die Lieferung der 6502 CPU zur Lieferung ab 1982. Sie soll in den bereits seit längerem angekündigten HHC (Hand Held Computer) eingebaut werden.

**Instant Pascal für den AIM 65** war bei Abschluß dieses Heftes Anfang Februar 1982 noch nicht lieferbar. Ein deutsches Anwenderhandbuch ist bei H.J. Regge, Fesenfeld57, 2800 Bremen beziehbar

**Für CBM der Serien 3001 bis 8001** bietet die Fa. Rossmöller in Bonn, Kaiserstraße 24 eine Reihe von Erweiterungen an: EPROM-Programmier- und Löschgerät, 3- oder 16-fach umschaltbare ROM-Boxen, 4K RAM-Karte zum Stecken auf ROM-Sockel, COMBASIC (kommerzielle 4K Erweiterung), hochauflösende Grafik nebst Plot-Programm dazu.

**Interfaces für die Schreibmaschine Olivetti P30/35** werden jetzt von der Firma MICCON, Wallensteinstraße 146 in 8500 Nürnberg 80 angeboten. Bei der Schreibmaschine handelt es sich um ein elektronisches Gerät mit Typenrad zum interessanten Preis von etwa DM 1250,-. Es werden Interfaces für die Schnittstellen V24, 20 mA und IEEE 488 geliefert. Dabei gibt es eine Version mit eigener CPU und Pufferspeicher, die an jeden Rechner anzuschließen ist. Besitzer eines AIM 65/PC 100 können die Schreibmaschine jedoch auch über die User-VIA 6522 betreiben, wenn sie ein verändertes E-ROM zusammen mit einem einfacheren Interface benutzen (ca. DM 255,-) Die Tastatur des AIM wird dadurch in die Lage versetzt, Kleinschreibung, deutsche Sonderzeichen, die übrigen Sonderzeichen der Schreibmaschine und Kontrollcodes entgegenzunehmen und im Text-Editor zu speichern.

**Commodore führt ab 9. Februar 1982 verschiedene Trainings-Kurse durch,** und zwar BASIC für Anfänger, BASIC für Fortgeschrittene, Arbeiten mit der Floppy, PASCAL, Assembler, IEEE-Workshop. Anmeldungen/Auskünfte: Commodore Büromaschinen GmbH, z.Hd. Herrn Berlipp, Dornhofstraße 38, 6078 Neu-Isenburg, Tel.: 061 02 - 249 - 132.

**Für die speicherprogrammierbare Steuerungstechnik (SPS)** steht ein Lehrprogramm für die innerbetriebliche Ausbildung auf CBM 3016/32 und 4016/32 bei der Firma Dipl.-Ing. Wulf Dietmar Hein, Kollenrodtstr. 17 in 3000 Hannover 1 zur Verfügung (Tel.: 0511 - 62 80 80). Es nennt sich LOG 1.0, bringt die logischen Funktionen auf den Bildschirm, hat Zeitglieder und Zähler/Teiler.

**SYSTEMHELP für CBM der Serien 8000 und 4000** ist ein großes Paket zur Unterstützung des Programmierers (2x4 KB EPROM) der Firma Hard + Soft GmbH.Spitzwegstraße 42 in 8580 Bayreuth (T.: 0921 - 6 88 77). Außer den üblichen Erweiterungen mit AUTO, DELETE, FIND usw. und Floppy-Kurzbefehlen erlaubt es ohne Programmzerstörung ein Listen von Programmen oder Daten von der Floppy auf den Bildschirm oder einen Drucker. Programme können verschlüsselt werden, man kann Soft-Keys für ganze Befehlsfolgen anlegen, es gibt Uhrbefehle, ein universaler String-Sort ist implementiert. Die Erstellung von Masken ist komfortabel gelöst, daneben gibt es eine Fülle von Bildschirmbefehlen.

**Interfaces für die elektronischen Schreibmaschinen Olivetti ET 121, 201, 221 und 231** werden seit einiger Zeit ebenfalls von der Fa. Hard + Soft GmbH in Bayreuth geliefert. Alle Funktionen dieser Schreibmaschinen können auch vom Computer angesteuert werden, sogar auch Proportionalschrift mit Blocksatz, Leistungen, die man sonst nur bei wesentlich teureren Textsystemen findet. Ein solches Interface ist beim Herausgeber seit einigen Monaten störungsfrei im Einsatz und soll demnächst eingehender besprochen werden.

## Programmier-Workshops 6502 und 6809

**Der Herausgeber führt am 5./6. März für 6502 bzw. am 12./13. März 1982 für 6809 (jeweils Freitag/Sonnabend) wiederum Intensiv-Schulungen für die Assembler- und Interface-Programmierung durch. In den vergangenen Jahren wurden bereits viele Kurse dieser Art erfolgreich durchgeführt. Ort: Ahrensburg bei Hamburg. Kosten je Teilnehmer DM 540,-- + MwSt. Sonderprospekt und (frühzeitige!) Anmeldung beim Herausgeber.**

# DUO Plott Interface für MX80 F/T

**und COMMODORE Rechner 30/40/80**

**Paralleles IEEE 488 - Interface:**

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

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

# Umrüstsatz MX80 F/T auf MX82F/T

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

Preis incl. MwSt. DM 250,--

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

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


## Stellberg Computer-Systeme

COMMODORE EPSON C' ITOH SOFTWARE INTERFACE

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

# BEPCO Computer-Entwicklungssystem

**im Euroformat**

| | |
|---|---|
| **CPU-Karte** 6502, ACIA 6551, VIA 6522, RAM/EPROM on Board<br>V24/20 mA schaltbar, DMA | 498,- DM |
| **40 KB EPROM** Betriebssystemkarte | 199,- DM |
| **Floppy Controller 5,25"/8"** single/double dens.<br>mit Treiber-Softwarelisting (6502) | 849,- DM |
| **64 KB RAM-Karte** (4116), statischer Betrieb<br>Refresh on Board | 660,- DM |

Gehäuse mit Busplatine (21x12x25 cm), Analogwandler 8-Kanal,
Video Interface, Netzteile, ASCII-Tastatur, Logikanalysator,
2fach Kassetteninterface i.V., alle EUROFORMAT
Bus-kompatible FELTRON/ELEKTOR, gesockelt, bestückt, geprüft

**SOFTWARE** Basic, Focal, Editor, Assembler, Forth, Pascal

**Auch weiterhin MZ80K, MZ80B,** SOFTWARE-UNTERSTÜTZUNG
alle Listings (teilw. kommentiert), eigene Betriebsprogramme

## Niehus Hobby Electronik

2320 Plön
Johannisstraße 7
Postfach 189
Telefon 045 22 - 27 42

# Systemerweiterung für AIM 65 / PC 100

## auf Europakarten

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

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

792,- + MWSt = 894,96 DM

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

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

**ANALOG E/A** 8 Bit, max. 80 kHz Abtastrate, mit Aliasingfiltern und sample- and hold, E/A unabhängig voneinander

370,- + MWSt = 418,10 DM

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

180,- + MWST = 203,40 DM

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

240,- + MWSt = 271,20 DM

**Neu 32 KB CMOS-RAM,** statisch 200 ns, für alle 8-Bit-CPUs geeignet,
DM 884,- + MwSt = 998,92 DM

Die Alternative zum Nadeldrucker bietet vieles, was bisher nur die großen elektronischen Büromaschinen leisten. **Elektronische Typenrad-Schreibmaschine Praxis 35/2 mit integrierter Schnittstelle,** 29 verschiedene Schriftarten, 3 Schriftgrößen (10, 12, 15 Zeichen/Zoll), 10 Zeichen Korrekturspeicher (list off und cover up), Carbon- und Textilfarbbandcassetten, Centronix-Schnittstelle, geschwindigkeitsoptimert mit Nutzung des internen Schreibpuffers. Prospekt anfordern! Wir sind Olivetti-Händler und bieten vollständigen Service!

| | | |
|---|---|---|
| Preise: | Praxis 35 ohne Interface, inkl. MwSt | 1340,- DM |
| | Praxis 35 mit Centronix-Interface, inkl. MwSt | 1898,- DM |
| **EPROM-Löschlampe** mit E27-Schraubsockel | | 65,- DM |
| **Philips MDCR-Laufwerk,** inkl. MwSt | | 370,- DM |


DIPL.-ING. HORST NEUDECKER

INGENIEURBÜRO FÜR MESSTECHNIK

**Mehringplatz 13**
**1000 Berlin 61**

**Tel.: 030 - 251 20 00**
**030 - 262 45 18**

# Video-Interface für AIM 65 / PC 100

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

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

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

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

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

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

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

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

**32 KB RAM-Modul zum direkten Einstecken in PC 100/AIM 65.** Die Steckerleiste mit dem Systembus bleibt für externe Zusätze frei, die ROM-Sockel werden nicht verdeckt. Preis für 32 KB DM 692,- + MwSt = DM 781,96
teilbestückt mit 16 KB DM 426,- + MwSt = DM 481,38

**Neuerscheinungen der Europakartenserie:** 256 K-RAM (5 Volt), AIM 65-kompatibler 6502A-Single-Eurocard-Computer, PASCAL-ROMkarte.
Bitte Unterlagen anfordern!

DIPL.-ING. HORST NEUDECKER

INGENIEURBÜRO FÜR MESSTECHNIK

# 65xx MICRO MAG

## COMPUTING · SOFTWARE · HOBBY

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

65xx MICRO MAG erscheint zweimonatlich, jeweils Mitte Februar, April usw..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 26 Dollar. Abonnements laufen bis auf Widerruf mit Kündigungsmöglichkeit bis zu zwei Wochen vor deren Ablauf.

**Nachliefermöglichkeiten:** Hefte 1-13 sollten als Buch (s. Anzeige unten) nachbezogen werden. Solange Vorrat reicht, können auch noch einzelne Hefte der Nos. 7-13 geliefert werden. Hefte 14-22 sind unbeschränkt nachlieferbar.

Private Besteller werden um Überweisung/Scheck (auch Auslandsschecks) zusammen mit der Bestellung gebeten. Konto Roland Löhr, Nr. 654 70-2-2 Postscheckamt Hamburg, BLZ 200 100 20.

## Leser-Service des Herausgebers

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

**Thermokopf (Printerplatte) für AIM 65 und PC 100**
mit Anleitung für den leichten Einbau. Auffrischung des Druckes DM 25,--

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

**Das Buch 7-13 des 65xx MICRO MAG** neu !
Sammelband ohne Anzeigen, ca. 340 Seiten, geb., enthält neben etwa 20 allgemeinen Darstellungen 25 Artikel und Programme für CBM/PET sowie 35 für AIM 65/PC 100. Eine Fundgrube für die fortschreitende Systemnutzung DM 42,--

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

**Programming & Interfacing the 6502 with Experiments**
von Marvin L. de Jong, ca. 410 S., engl., viele Schaltungsvorschläge und die Programme für den Betrieb der Interfaces dazu, didaktisch gegliedert DM 59,--

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

**AIM 65 Laboratory Manual And Study Guide**
von L. J. Scanlon (jr.). ca. 185 Seiten, engl. Lehrbuch mit vielen Übungen zur Benutzung und Programmierung des AIM. Sehr geeignet für Anfänger DM 26,--

**Forth User's Guide**
Handbuch der Firma Rockwell für ihr Fig-FORTH zum AIM 65, ca. 300 S., engl.. Erklärung des Befehlssatzes und der E/A, die im vorläufigen Handbuch noch fehlte. Geeignet auch für andere Betreiber eines Fig-FORTH DM 24,--

Vorstehende Preise sind Endpreise einschl. Verp. und Porto. Nachnahme: +DM 1,50
**Workshops für 6502 und 6809: Beachten Sie bitte die Hinweise im Heftinneren.**