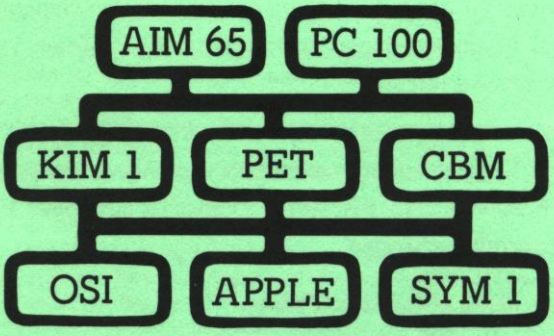
65xx MICRO MAG

COMPUTING · SOFTWARE · HOBBY

DM 7,80 Nr. 17 Februar 1981



Inhaltsverzeichnis

Analog / Digitalwandler

Mit dieser Analog/Digitalwandlerkarte können
6 Kanäle mit einer Genauigkeit von bis zu
10 Bit gewandelt werden.

Die Wandlungszeit hängt direkt von der geforderten Genauigkeit ab.
Außer den 6 Analogeingängen sind noch 10 Bit I/O über
den Frontstecker erreichbar.

Der Eingangsspannungsbereich der Analogkanäle geht von 0 bis 5 V.
Damit dieser Eingangsspannungsbereich vom Anwender an
besondere Probleme angepaßt werden kann, ist auf dieser Leiterplatte
ein Lochrasterteil vorhanden.

Preis: DM 324,-- zzgl. MWSt

Weiterhin lieferbar:

CPU-Karte NICO 65 mit 6502 CPU
CPU-Karte NiCO 69 mit 6809 CPU
Combo-Karte mit 8 k RAM und 2 VIAs
Buskarte
EPROM-Programmiergerät

ELEKTRONISCHE BAUGRUPPEN
MICROPROZESSOR-TECHNIK
Dohmann

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

PASCAL für CBM

Die strukturierte Programmiersprache PASCAL (nach K. Jensen und N. Wirth, Zürich) stößt bei den Betreibern der verschiedensten Systeme auf zunehmendes Interesse, wobei man allerdings bemerken muß, daß ihre Kenntnis wohl noch nicht sehr verbreitet ist. Man muß es daher sehr begrüßen, daß Commodore jetzt ein leistungsfähiges PASCAL für seine Rechner freigegeben hat. Wir benutzen die Gelegenheit, nicht nur diese Implementierung, sondern auch das Vokabular und die Datentypen dieser Sprache in einer allgemeineren Form für alle Leser vorzustellen. Zunächst jedoch eine maschinenmäßige Orientierung:

1. Die maschinenmäßige Umgebung für die Sprachimplementierung

Rechner wie PET und CBM melden sich beim Einschalten im BASIC-Modus, bei dem die im höheren Adreßbereich der Maschine angesiedelten Festwertspeicher benutzt werden. Man verläßt die Kommandoebene des BASIC durch Anweisungen wie SYS oder USR(.). In diesen Fällen bewegt man sich im allgemeinen in Programmabschnitten, die im etwa 31 KB großen Arbeitsspeicher (RAM) im Adreßbereich von etwa hex 033A bis 7FFF abgelegt sein mögen. Solche Programmabschnitte können außer Anwenderprogrammen in Maschinensprache natürlich auch Interpreter oder Kompiler einer anderen Sprache zusammen mit einer neuen Befehlsebene sein. So ist es bei PASCAL.

Eine solche Sprachimplementierung benutzt zwar die allgemeinen Dienstleistungen des Betriebssystems wie etwa zur Tastaturabfrage, zur Bedienung des Bildschirmes und anderer E/A-Einheiten. Bei konsequenter Programmierung verzichtet sie jedoch auf die Mitbenutzung von Routinen des vorhandenen BASIC-Interpreters. Man kann also sagen, daß unter PASCAL oder einer anderen zusätzlichen Sprache der Adreß- und Speicherbereich des eigentlichen BASIC entbehrlich, wenn nicht gar ein wenig hinderlich ist.

Solche Überlegungen führen z.B. beim APPLE III oder bei einer von Commodore erwarteten Weiterentwicklung dazu, daß man den Rechner nicht mehr mit den großen Festwertspeichern für nur eine Sprache ausrüstet, sondern ihn nur noch mit einem kleinen 'Urlader' versieht, der die Inbetriebnahme eines Diskettensystems erlaubt. Beim Systemstart entscheidet man sich also, welche Sprache oder welches lauffähige Programm man von der Diskette in die Maschine laden will. Damit wird ein Adreßraum von etwa 64000 Bytes als reiner Schreib-/Lesespeicher (RAM) verfügbar, die Hardware ist universeller nutzbar.

Man hat aber auch andere Konsequenzen gezogen: Mit ihren 16 Adreßbuspins kann eine 6502-CPU nur 64 K Adreßbereich direkt ansprechen. Für lange Programme, den Gebrauch höherer Sprachen und für die Präsenz vieler zusätzlicher Dienstleistungen (oft mit überschneidendem Adreßbereich) ist das oft nicht ausreichend. Der Ausweg liegt im 'memory banking', einer von der CPU über Interfacebausteine gesteuerten Zu- oder Abschaltung (Enable/Disable) ganzer Speicherbatterien in Blöcken von z.B. 24, 32 oder 48 KB. In diesen Blöcken mögen sich RAMs befinden und/oder auch Festwertspeicher (Sprachen). Einer 8-Bit-CPU werden derzeit schon Speicherbereiche bis etwa 128 KB zugänglich gemacht. Es ist aber abzusehen, daß man im Laufe der Zeit auch Megabyte-Speicher (in Blöcken) haben könnte. Der PC 1000 von Siemens war in diesem Sinne geplant. Hinsichtlich der tatsächlichen Ausnutzung werden die Erfordernisse des Anwenders, Speicherpreise und Energieverbrauch aber auch Fragen der Bearbeitungszeit mitsprechen.

2. PASCAL auf dem CBM

Das CBM-PASCAL residiert noch nicht in einer der aufgezeigten weiterweisenden Umgebungen, es residiert im 32 KB Arbeitsspeicher im Bereich bis hex 7FFF bei weiterer Anwesenheit des BASIC in den Festwertspeichern. Es wird in Form einer Diskette und eines zu ihrem Betrieb erforderlichen Sicherheits-ROM geliefert, zu einem Preise im Bereiche von DM 300,- (ohne Gewähr). Die begleitende Dokumentation mit ihren Beispielen ermöglicht eine unmittelbare

Inbetriebnahme. Es handelt sich um eine Implementierung der Transam Components Limited (TCL).
PASCAL ist eine Kompilersprache. Was heißt das? - Wir machen das am Gegensatz zum BASIC-Interpreter deutlich: Eine BASIC-Programmzeile wird mit nur geringen Veränderungen als Programm-Quelltext abgelegt (ab hex 0401 im CBM). Die Veränderungen gegenüber der eingetippten Zeile bestehen darin, daß für die 'reservierten' Anweisungsworte wie PRINT, FOR, TO usw. sogenannte 'tokens' abgelegt werden, also Befehlsverschlüsselungen in einem Byte. Hinzugefügt wird auch der Verweis auf die Beginnadresse der nächsten Anweisungszeile. Während der Ausführung eines Programmes muß das BASIC-Steuerprogramm die Befehlsbytes interpretieren. Viel Zeit wird weiterhin benötigt, um zunächst einmal den Speicherort der angesprochenen Varibalen zu finden. Dabei wird bei jeder Ansprache einer Variablen das Variablenverzeichnis immer wieder von vorne an durchgekämmt. Der Interpreter kann sich also den Ort von früher schon einmal benutzten Variablen nicht merken. Ähnliches gilt für GOTO, GOSUB, ON usw.. Auch die Syntaxprüfung, die Zulässigkeit der Formulierungen unter BASIC, findet erst zur Ausführungszeit statt (ggfs. Syntax Error).

Ein Kompiler ist demgegenüber ein Bearbeitungsprogramm für die Quelltexte einer Sprache, aus denen er ein ablauffähiges 'Objektprogramm' für die spätere Ausführung generiert. Der Kompiler übernimmt dabei nicht nur die Syntaxprüfung im Vorwege, sondern bewirkt auch eine Zuweisung der Speicheradressen für Prozeduren und Variablen im generierten Programm. Das kompilierte Programm ist damit näher an der Ausführung.

Die Umwandlung eines Quellenprogrammes in ein reines Maschinenprogramm würde zur wiederholten Ablage gleicher auszuführender Routinen und zu einem erheblichen Speicherbedarf führen. Man geht daher einen Mittelweg, indem man das Quellenprogramm in einen Zwischencode, den sog. P-Code umwandelt, der zur Ausführungszeit mit einem Minimum an Interpretation in den Aufruf ausführender Routinen verwandelt wird. In dieser Konstellation benötigt man zur Laufzeit also den P-Code sowie die in einfacher Menge abgelegten ausführenden Routinen als 'runtime package'.

In dieser Umgebung sind also, auch für den CBM-Compiler, folgende Phasen eines Programmes begrifflich zu trennen, auf die wir noch im einzelnen eingehen: Editierung des Quellentextes in PASCAL, Kompilierung und schließlich Ausführung.

3. Die Befehlsebenen des CBM-PASCAL

PASCAL wird mit LOAD"*",8 und mit RUN aktiviert. Man befindet sich danach auf der höchsten Befehlsebene. Von dort aus begibt man sich durch Eingabe von Schlüsselwörtern zu weiteren Ausführungsebenen, nämlich Quelltexteingabe, weitere Leistungen des Text-Editors, Diskettenbefehle, Kompilerbefehle, allgemeine Dienstleistungen.

3.1 Texteingabemodus

Durch Eingabe einer beliebigen ersten Zeilenummer und nachfolgenden Text gelangt man in den Texteingabemodus. Mit jedem RETURN wird eine um 10 erhöhte nächste Zeilennummer automatisch vorgegeben. Dieser Modus wird zur höheren Befehlsebene hin verlassen, indem man 2x RETURN betätigt.

3.2 Der Text-Editor

Die Editierungsmöglichkeiten des PET/CBM für BASIC-Programme sind bereits angenehm, unter PASCAL sind sie noch komfortabler. Auch hier gilt wieder 'what you see ist what you get'. Es wird also am Bildschirm mit INSERT und DELETE und mit den Cursorbewegungen editiert. Man kann in eine beliebige Zeile hineinfahren, ändern und die Zeile dann mit RETURN in denTextspeicher übernehmen. Der Quelltext wird wie bei BASIC in numerierten Zeilen angelegt. Eine bereits vorhandene Zeile wird gelöscht, indem man nur die Zeilennummer und RETURN eintastet. Wie bisher üblich, kann man Zeilen mit einer Zwischen-Nummer einschieben oder zu einer vorhandenen Zeilennummer einen neuen Text anlegen.

Daneben gibt es eine AUTO-Anweisung, mit der der Benutzer das Inkrement der Zeilennummern vorgeben kann. NUMBER erlaubt ein Renumerieren der Zeilen von ... bis Nummer (allerdings nicht zu kleineren Nummernbereichen hin). LIST und NEW arbeiten wie vom BASIC her bekannt. Neu ist DELETE, mit dem man eine Zeile oder von ... bis Zeilen Quelltext löschen kann. Angenehm sind die zeichenkettenorientierten Befehle FIND (schreibt alle Zeilen auf den Schirm, in denen der Suchstring vorkommt) und CHANGE, das alle Stellen, die dem Suchstring entsprechen, auf einen neuen String abändert und anzeigt. Mit UPPER oder LOWER wird der Zeichensatz umgeschaltet, BASIC führt zu BASIC zurück, BREAK zum TIM-Monitor.

3.3 Diskettenbefehle

Auf der PASCAL-Befehlsebene stehen weiterhin die üblichen verkürzten Diskettenbefehle des DOS-Supports zur Verfügung, das also nicht extra geladen werden muß.

Ein Quelltextprogramm wird mit dem Befehl PUT auf Diskette abgelegt. Mit GET kann es zurückgeladen werden.

3.4 Kompilerbefehle

Mit dem Laden von PASCAL befindet man sich im RESIDENT-Modus, d.h. der Kompiler ist ins RAM geladen (nur noch die Fehlermeldungen müssen ggfs. von Diskette nachgezogen werden). Der Befehl PRINT FRE(0) zeigt etwa 4861 für ein Quellprogramm noch freie Bytes an (Textspeicher). Für kleinere Programme reicht das aus. Bein residentem Quelltext und Kompiler bewirkt der Befehl 'R' eine Kompilierung mit sofort folgender Ausführung, wenn keine Fehler auftraten. Andernfalls werden die Fehlerorte und -arten angezeigt. Die dazu benötigten Nachrichten werden aus Platzgründen von der Diskette nachgezogen. Mit dem Befehl 'L' erhält man eine Fehleranzeige zusammen mit den Quelltextzeilen. 'P' erlaubt die Listung der Kompilierung auf dem Drucker.

DISK-Modus. Um längere Quelltexte anlegen zu können, wird mit DISK ein Textspeicher von 28925 Bytes bereitgestellt, wobei der Resident-Compiler überschrieben wird.

Ein solcher längerer auf Diskette abgelegter Quelltext (PUT) wird mit COMP (Name) kompiliert und als Name.OBJ als Objectfile auf Diskette abgelegt. Hierbei bestehen Listoptionen. Der Befehl EX (Name) führt zum Laden und zur Ausführung des benannten Objectfile (des kompilierten Programmes).

Der Befehl LINK erlaubt die Zusammenbindung verschiedener benannter objectfiles zu einem Gesamtprogramm als neues objectfile mit eigenem Namen. Daß hierbei besondere Standards eingehalten werden müssen, ist klar. Aber die gegebenen Möglichkeiten sind weitreichend.

Auch bei der Editierung eines Quelltextes gibt es Einbindungsmöglichkeiten für Textmodule, die sich bereits auf Diskette befinden. Weiterhin können mit LOCATE kleinere PASCAL-Programme erzeugt werden, die ohne das Vorhalten der Kompiler-Diskette lauffähig sind (sie werden unter BASIC (!) einfach mit RUN gestartet). Diesen Programmen wird das etwa 10 KB große runtime package angefügt. Sie sind damit z.B. auf einer 16 KB-Maschine ausführbar. Solche Programme sind wegen der einfachen Hantierung für ständige Anwendungen durch weniger geübte Mitarbeiter geeignet.

Der Befehl RESIDENT führt zum Wiederladen des residenten Kompilers, der durch DISK verlassen und aufgegeben wurde.

3.5 Allgemeine Dienstleistungen

Auf der höchsten Befehlsebene des PASCAL-Systems stehen folgende weitere Dienste zur Verfügung: Mit BREAK gelangt man auf die Befehlsebene des TIM-Monitors (entspr. SYS 1024 des BASIC). BASIC führt in den normalen BASIC-Modus zurück. Das DOS-Support wurde ebenfalls bereits erwähnt. Implementiert ist ferner das Tasten-REPEAT.

Die Befehle HEX und DECIMAL dienen der Umrechnung zwischen den Zahlensystmen. Aus dem Befehlssatz des BASIC sind erreichbar: Lesen und Setzen der Systemuhr TI$, ?FRE(0), Status ST, ferner die Befehle OPEN, CLOSE, PRINT, PRINT , POKE, SYS, FOR und LET.

Soweit die Einbettung des CBM-PASCAL in das System. Von einer Sprachimplementierung ist zu verlangen, daß sie bequem und sicher zu handhaben ist. Diese Voraussetzungen sind hier erfüllt. Die Kommando-Ebene ist vielseitig, ihre Editierungsmöglichkeiten sind komfortabel. Das Hantieren mit Textfiles, Objectfiles und mit dem Compiler wird nach dem Studium der Unterlagen und nach einigen Übungen schnell beherrscht. Dieser Teil des Artikels möge weitere Klarstellungen vermittelt haben.

4. Die Sprache PASCAL

4.1 Programmstrukturen

Nach dieser Beschreibung des problemlosen 'roundabout' sollen hier nun einige wichtige Merkmale des PASCAL und seine Erweiterungen in der CBM-Version dargestellt werden, um dem noch wenig vorbereiteten Leser einen ersten Begriff zu geben.

PASCAL ist eine strukturierte Sprache, d.h. es kommt nicht nur auf den richtigen Sprachgebrauch (Syntax), sondern auch auf die Reihenfolge der Anweisungen an, so wie wir es z.B. auch vom COBOL her kennen. In Vereinfachung ist zu sagen: Jedem Programm und auch jedem Programmteil (Prozedur, Funktion) ist ein Erklärungsteil voranzustellen, in dem die dort benutzten Variablen mit Namen und Variablentyp erklärt werden. Das Gleiche gilt für Konstanten (mit Wertzuweisung) und für Parameter, die einer Prozedur (geblocktes Unterprogramm) von seinem Aufrufer her übermittelt werden. Einen solchen notwendigen Erklärungsteil kennen wir in ähnlicher Form ja auch von der Assemblerprogrammierung her. - Ebenso muß eine wo immer aufgerufene Prozedur oder Funktion bereits weiter vorne im Programmtext programmiert worden sein. Das führt dazu, daß das die Ausführung kontrollierende Hauptprogramm am Schluß des Programmtextes steht.

Dieser Aufbau zieht nach sich, daß Variable für die jeweiligen Prozeduren 'lokal' sind, sie können nur von ihnen erreicht und benutzt werden und sie können nach Prozedurabschluß aufgegeben werden. Nur die Variablen des Hauptprogrammes sind 'global', sie können von überall her erreicht werden, es sei denn, eine Prozedur verwendet ihren Namen auch 'lokal'.

Der Ausführungsteil von Programmen und Programmsegmenten wird immer durch das Konstrukt BEGIN ... END nach außen begrenzt. Dazwischen liegen die auszuführenden Anweisungen. Ein solches Konstrukt wird BLOCK genannt. Ein GOTO ist nur ganz begrenzt möglich.

BASIC erlaubt ein 'Freistilprogrammieren'. Es ist dem Benutzer unbenommen, Variable an jeder Programmstelle einzuführen und zu benutzen, seine DATA legt er oft an den Schluß des Programmes, ebenso seine Unterprogramme. Dazwischen kann er beliebig hin- und herspringen. Das geht nicht unter PASCAL, wo alles in weiter oben stehenden Programmteilen bereits erklärt sein muß.

Höhere Sprachen haben natürlich auch viele Gemeinsamkeiten. PASCAL erlaubt zunächst einmal die Verwendung symbolischer Namen für Programmabschnitte, Variable, Konstante und Parameter. Die arithmetischen und vergleichenden Operationen werden praktisch gleich formuliert. Wertzuweisungen erfolgen mit ':=' und Typzuweisungen mit ':'. Das Semikolon (;) dient als Begrenzer für Statements (wie in BASIC der Doppelpunkt oder RETURN, gefolgt von einer neuen Zeilennummer). Die zur Verfügung gestellten höheren mathematischen Funktionen entsprechen denen z.B. des BASIC. Funktionsargumente werden wie üblich in Klammern geschrieben. Das CBM-PASCAL hat 9 signifikante Ergebnisziffern und deckt den Wertebereich von 1E-38 bis 1E+38 ab.

Die Besonderheiten des PASCAL liegen in den Steuerungsanweisungen und in den weiteren zur Verfügung gestellten Funktionen und Datentypen.

4.2 Steuerungsanweisungen

Die bekannte Zählschleife formuliert man in PASCAL z.B. als FOR I :=1 TO 5 DO ..(Ausführungsteil), wobei zuvor erklärt sein muß: VAR I : INTEGER.

Die Bedingungsschleife formuliert man REPEAT ... UNTIL (Endbedingung). Zwischen diesem Konstrukt liegen die auszuführenden Anweisungen.

Bedingungsabfragen erfolgen mit IF (Bedingung) ... THEN (auszuführende Anweisung/en), auch mit einem zweiten Statement-Teil: ELSE DO für den nicht zu IF gehörenden Entscheidungsast.

Das CASE-Statement ist vergleichbar mit der Formulierung ON (Eingangsbedingung) DO .... Im DO-Teil stehen auszuführende Anweisungen, nicht etwa ein GOTO. Auf das CASE-Statement folgt eine Fall-Liste mit je einer formulierten Eingangsbedingung und nachfolgendem Anweisungsteil. Wenn man will, ist dieses Statement eine übersichtliche Entscheidungs- und Anweisungstabelle.

Das Statement WHILE (Bed. noch nicht erfüllt) DO stellt die Bedingungsabfrage schon vor die erste mögliche Ausführung (bei REPEAT ... UNTIL steht sie nach jeder Ausführung), so daß eine Ausführung ggfs. gar nicht mehr erfolgen muß, wenn die Bedingung zu Beginn schon erfüllt ist. - Der Umgang mit diesen Steuerungsanweisungen dürfte nicht weiter schwerfallen.

4.3 E/A-Anweisungen

Mit den lesenden und schreibenden Anweisungen und ihrer Formatierung wollen wir uns hier nicht im einzelnen befassen. Die Bedienung der Systemeinheiten Tastatur, Bildschirm und von an den IEEE-Bus angeschlossenen Einheiten mit Erkennung von Zeilenende und File-Ende sind durch die Gruppe der READ- und WRITE-Anweisungen abgedckt. Im Laufe der Zeit wird man sicherlich auch Dateiverwaltungssystme für CBM sehen, die nicht nur einen sequentiellen Zugriff ermöglichen.

4.4 Variablentypen und ihre besonderen Anweisungen

Die von PASCAL akzeptierten Grundtypen sind Ganzzahlen, Gleitkommazahlen, ASCII-Bytes, logisch wahr/unwahr (Integer, Real, Char, Boolean). Daneben können unbeschränkt vom Anwender zu definierende Variablentypen geführt werden, z.B. die Aufzählungstypen. Ein Beispiel: TYPE WOCHENTAG =(Montag, Dienstag, ...,Sonntag). Aus dieser Aufzählungsmenge läßt sich ein 'subrange' bilden, z.B. TYPE Arbeitstage=(Montag, ...,Freitag).Für diese Aufzählungstypen hält PASCAL die Funktionen ORD (PLatznummer in der Aufzählung), SUCC (Ansprache des Folgeelementes) und PRED (vorhergehendes Element) Zur Verfügung.

Der Typ CHAR (Character) hat die Zeichenlänge von 1 Byte. Die Funktion ORD (Character) liefert eine Ganzzahl zwischen 0 und 255 entsprechend dem ASCII-Code des Zeichens ab. In der Umkehrung erzeugt CHR(Zahl) ein ASCII-Zeichen, ähnlich wie in BASIC.

Während man Literale, in Anführungsstriche eingeschlossene Strings also, mit einer Anweisung WRITE('Literal') drucken kann, müssen Stringvariable in dieser Implementierung wie auch im Standard-PASCAL zunächst erklärt werden, z.B. TYPE STRING= PACKED ARRAY (1..80) OF CHAR; VAR WORD:STRING. Der erste Teil erklärt Strings als Zeichenketten bis zu 80 Zeichen. Danach kann die Variable WORD als vom Typ STRING erklärt werden. Eine solche Erklärung könnte sich für weitereStringvariableanschließen. Damit stehen dem Anwender auch weitere Stringoperationen zur Verfügung.

Die Wahrheitsvariablen vom Typ BOOLEAN nehmen die Werte TRUE oder FALSE im Gefolge von vorhergehenden Operationen an. Sie werden weiterhin durch die Funktionen EVEN oder ODD (gerade Zahl, ungerade Zahl) gesetzt. Ausdrücke in Bedingungsabfragen (true oder false) können ferner durch logisch AND, OR und NOT verknüpft werden.

Interessant sind die Variablen vom Typ SET (eine Menge von Elementen), in denen bis zu 127 Elemente eines Grundtyps aufgezählt werden (bei Zahlen auch in verkürzter Notierung). Für sie gibt es Operationen der Mengenlehre, ob ein Element IN einer Menge ist oder zugleich in zwei Mengen enthalten ist, ob die Menge 1 eine Untermenge von 2 ist, ob die Mengen gleich oder ungleich sind, und auch eine Berechnung der Trennmenge ist möglich. Mit diesen Operationen sind Untersuchungen möglich, die häufigen aber lästigen Zuordnungarbeiten des Alltagslebens entsprechen und die sonst auch etwas umständlich zu programmieren sind.

Zu den höherentwickleten Datenstrukturen gehören die Datensätze, RECORDs, in denen Variable oder auch andere RECORDS oder ARRAYs durchaus unterschiedlichen Grundtyps zu

einem umfassenden Datenkonstrukt zusammengefaßt werden. Hierzu gehört die Operation WITH (item of RECORD) DO. Es ist also eine direkte Ansprache des im RECORD enthaltenen Datenelementes möglich.

ARRAYs: Variable können in mehreren Dimensionen dimensioniert werden und dürfen als Elemente beliebige, auch anwenderdefinierte Datentypen enthalten, ebenso andere arrays als Elemente.

Datensätze, Arrays und Files können als optimiert zu packen erklärt werden. Dazu gehören die Prozeduren PACK und UNPACK. - Pointervariable nehmen dynamisch den Speicheradreßwert von Variablen an.

4.5 Erweiterungen gegenüber Standard-PASCAL

In der voraufgehenden Beschreibung sind schon zahlreiche Erweiterungen gegenüber dem Entwurf von Jensen/Wirth enthalten. Erwähnenswert sind weiterhin: Hexadezimale Konstante sind zugelassen, ebenso der Zugriff in die Maschine mit PEEK, POKE, ORIGIN, GETKEY, HEX lesen und schreiben. Bitmanipulationen mit logisch AND, OR, EXOR und NOT sind in 16 Bit Breite möglich, ferner Verschiebeoperationen in 16 Bit Breite (entsprechend ASL, LSR). Implementiert ist ferner ein Zufallszahlengenerator, Zugriff zur Systemuhr und Anschluß an Programme in Maschinensprache, entsprechend dem SYS.

5. Zusammenfassung

PASCAL ist für die meisten Leser dieser Zeitschrift eine neue Sprache und eine neue Möglichkeit. Sie wird für die so verbreiteten CBM-Systeme nunmehr preiswert angeboten und erhält damit die Chance, Allgemeingut zu werden. - Die Ausführungen mögen dargelegt haben, daß der Umgang mit Text-Editor und Kompiler denkbar einfach ist. In die Strukturierung dieser Sprache findet man sich hinein und erhält damit ein reiches Vokabular programmatischer Ausdrucksmöglichkeit. Der Anwender kann eigene Variablentypen erklären, Aufzählungstypen, Sets und höhere Strukturen wie Records und Arrays mit Elementen unterschiedlicher Grundtypen. Er kann mit ihnen außer arithmetischen, logischen und stringorientierten Operationen Verknüpfungen entsprechend der Mengenlehre vollziehen und damit in knapp formulierte Wahrheitsräume vordringen, die bisher mühsam auszuprogrammieren waren.

Innerhalb der geforderten Erklärungsfolgen ist PASCAL logisch und diszipliniert. Mitte 1982 werden wir uns wahrscheinlich nicht mehr darüber unterhalten, ob man PASCAL implementieren oder erlernen sollte. Wir werden dann wohl eher abwägen, welche bequemen Lösungen wir in BASIC, welche schnell ausführenden wir in Assembler und welche gründlich überdachten wir in PASCAL schreiben werden. - Ein gutes deutsches Lehrbuch zu dieser Sprache scheint noch nötig? Das TCL-PASCAL bietet mit seinen Erweiterungen gegenüber dem Standard ein leistungsfähiges Interface zur Maschine, der 6502. In diesem Sinne darf man im Laufe der Zeit auf Beiträge aus der Leserschaft hoffen, wie sie ja auch schon für die verschiedenen BASICs zur Veröffentlichung vorgelegt wurden.

R.L. □

Kleinanzeigen

Suche Drucker-Motor für CBM 3023. Horst Brettin, Tel.: 030 - 218-6615/16/17 (d) bzw. 685 53 44

1 Binder Matrixdrucker BM 132 neuwertiges Vorführmodell, 132 Schreibstellen, 180 Zch/Sek. IEC-Bus, PET-kompatibel 40 % unter Neupreis. Andere Schnittstellen auf Anfrage. Entwicklungsbüro Krickl, Schwenningen, Tel. 07720 - 33 875.

Suche zwecks Gedanken- und Programmaustausch Besitzer eines CBM mit Computhink oder CV-Floppy, der ebenfalls Maschinensprache programmiert. Habe PASCAL, Spitzenassembler und in Kürze Textverarbeitungsprogramm (16 K). Klaus Flesch, Badbergstr. 32, 7818 Oberbergen im Kaiserstuhl.

□

Dipl.-Math. A. Quint, Wiesbaden-Erbenheim

Kaufmännisches Rechnen auf dem CBM

Bei kaufmännischen Anwendungen benötigt man nur folgende Operationen: +, -, *, /. Manchmal benötigt man noch die Wurzelfunktion, z.B. bei Annuitätenberechnungen.

Bei Rechnern wie dem CBM werden Zahlen als Gleitkommazahlen gespeichert, hier in 5 Bytes. Davon dienen 4 Bytes der Speicherung der Mantisse und 1 Byte der Speicherung des Exponenten. In den 4 Bytes kannn man maximal 9 Ziffern speichern, DM 9.999.999,99 wäre also der größte darstellbare Betrag. Bei Beträgen über 10 Mio. DM erfolgt die Darstellung in exponentieller Form. Damit verschwinden dann die Pfennige und möglicherweise weitere geringwertige Stellen. In technischen Anwendungen ist diese Form der Darstellung mehr als ausreichend, da dort die Größe der Zahl und die ersten signifikanten Ziffern am interessantesten sind. Außerdem lassen sich die genannten Operationen mit Gleitkommazahlen leicht programmieren und die entsprechenden Algorithmen sind schnell.

Bei einem Buchhaltungsprogramm scheidet eine Darstellung der Zahlen in dieser Form völlig aus. Eine Zahlenangabe in exponentieller Form ohne exakte Angabe des Pfennigbetrages ist in einem solchen Programm nicht zu verwenden. Außerdem treten bei Operationen mit Gleitkommazahlen kleinere Fehler auf, aber gerade bei einem Buchhaltungsprogramm müssen die Pfennigbeträge stimmen. Und die größte exakt darstellbare Zahl ist bei Firmen mittlerer Größe bereits zu klein, weil häufig ein Umsatz über 10 Mio. DM auftritt.

Als Konsequenz muß man die entsprechenden Operationen selbst programmieren. Bei kaufmännischen Anwendungen treten Beträge mit 2 Stellen nach dem Komma (Pfennige) auf. Der Vorteil der Eigenbau-Routinen ist die Korrektheit bis auf die letzte Stelle. Damit stimmt dann auch der vielbeschworene Pfennig in der Buchhaltung wieder. Außerdem kann man die Routinen auf eine beliebige Stellenzahl auslegen. Ein Nachteil ist, daß sie langsamer als die vorhandenen Routinen sind. Man kann dies abmildern, indem man die Arbeitsvorgänge aufspaltet, in einen Teil, der die Anwesenheit des Anwenders erfordert (interaktiver Teil, z.B. Eingabe der Bewegungen) und in einen Teil, der ohne seine Anwesenheit laufen kann (Batch-Verarbeitung, z.B. Verbuchung der Bewegungen und Ausdruck des Journals).

Die üblichen Operationen in der Buchhaltung sind + und -. Multiplikation wird nur benötigt für die Berechnung der Mehrwertsteuer aus einem Betrag. Die Addition und die Subtraktion sollen aus den obengenannten Gründen als eigene Routinen programmiert werden.

Festlegungen: Eingabewerte stehen in X1$ und X2$, das Ergebnis wird zurückgegeben in AA$, also symbolisch: AA$ = X1$ + X2$.

Für die Subtraktion: Eingabewerte stehen in Y1$ und Y2$, das Ergebnis steht dann in AA$, also symbolisch: AA$ = Y1$ - Y2$.
Bei dieser Operation darf kein negatives Ergebnis entstehen, was auch bei einer Buchhaltung nicht vorkommen dürfte.

Die BASIC-Programme:

```
9400 REM
9450 X1=LEN(X1$):X2=LEN(X2$)
9500 MA=X1:IFMA<X2THENMA=X2
9550 IFINT(MA/2)*2=MATHENMA=MA+1
9600 IFLEN(X1$)<MATHENX1$=" "+X1$:GOTO9600
9650 IFLEN(X2$)<MATHENX2$=" "+X2$:GOTO9650
9700 CA=0:AA$="":X3$=""
9750 FORJ9=1TOMA-1STEP2
9800 J8=MA-J9:IFJ9>1THENJ8=J8-1
```

```
9850 X1=VAL(MID$(X1$,J8,2))
9900 X2=VAL(MID$(X2$,J8,2))
9950 X3=X1+X2+CA
10000 CA=0:IFX3>=100THENX3=X3-100:CA=1
10050 X3$=STR$(INT(X3+1E-5)):X3$=MID$(X3$,2)
10100 IFLEN(X3$)=1THENX3$="0"+X3$
10150 IFJ9=1THENX3$="."+X3$
10200 AA$=X3$+AA$:NEXTJ9
10250 IFCA=1THENAA$="1"+AA$
10300 IFASC(AA$)=48THENAA$=MID$(AA$,2):GOTO10300
10350 IFLEFT$(AA$,1)="."THENAA$="0"+AA$
10400 RETURN
10450 REM
10500 Y1=LEN(Y1$):Y2=LEN(Y2$)
10550 MA=Y1:IFMA<Y2THENMA=Y2
10600 IFINT(MA/2)*2=MATHENMA=MA+1
10650 IFLEN(Y1$)<MATHENY1$=" "+Y1$:GOTO10650
10700 IFLEN(Y2$)<MATHENY2$=" "+Y2$:GOTO10700
10750 CA=0:Y3$="":AA$=""
10800 FORJ9=1TOMA-1STEP2
10850 J8=MA-J9:IFJ9>1THENJ8=J8-1
10900 Y1=VAL(MID$(Y1$,J8,2))
10950 Y2=VAL(MID$(Y2$,J8,2))
11000 Y3=100+Y1-Y2-CA
11050 CA=1:IFY3>=100THENY3=Y3-100:CA=0
11100 Y3$=MID$(STR$(INT(Y3+1E-5)),2)
11150 IFLEN(Y3$)=1THENY3$="0"+Y3$
11200 IFJ9=1THENY3$="."+Y3$
11250 AA$=Y3$+AA$:NEXTJ9
11300 IFASC(AA$)=48THENAA$=MID$(AA$,2):GOTO11300
11350 IFLEFT$(AA$,1)="."THENAA$="0"+AA$
11400 RETURN
```

Beschreibung der Addition (Zeilennummern 9400-10400): Die Strings werden durch Blanks auf gleiche Länge gebracht. Da immer Paare von Zahlen verarbeitet werden, muß die Länge der Zeichenfolgen wegen des Punktes (=Dezimalkomma) immer ungerade sein. Die Strings werden von hinten nach vorne durchgearbeitet (Variable J8).Es werden immer Paare von Zahlen addiert (also Zahlen von 00 bis 99). Dabei entsteht möglicherweise ein Übertrag (Zahl größer oder gleich 100; Carry; Variable CA). Danach wird das Ergebnis in den String AA $umgewandelt. Falls am Ende der Addition noch ein Carry vorhanden ist, muß noch eine '1' an den String AA $ vorn angehängt werden (Beispiel: '99.00' + '5.00' ergibt '1'. + '04.00').

Beschreibung der Subtraktion (Zeilennummern 10450-11400): Das Programm ist analog zur Addition aufgebaut. Bei der Subtraktion ist allerding das Carry=1, wo bei der Addition galt Carry=0.

Beide Routinen benötigen natürlich mehr Zeit. Die Addition von Zahlen mit etwa 6 Stellen benötigt ca. 1/2 Sekunde. Da Tausende solcher Operationen vorkommen, lag es nahe, diese in Assembler zu schreiben. Da die Addition die am häufigsten gebrauchte Operation ist, kann man sich dabei auf diese beschränken.

Festlegungen: Bei einem 8-Bit-Prozessor können natürlich keine Paare, sondern nur einzelne Ziffern addiert werden. - Die Übergabevariablen X1$, X2$ und AA$ müssen als erste Variable definiert werden, also

```
(ZEILEN-NR.) CLR:X1$="":X2$="":AA$=""
```

```
027A  A5 2A     LDA  2A          BEGINN VON X2$ NACH $01 FF.
027C  69 07     ADC  #07
027E  85 01     STA  01
0280  A5 2B     LDA  2B
0282  69 00     ADC  #00
0284  85 02     STA  02
0286  A0 03     LDY  #03         STARTADRESSE DER STRINGS
0288  B1 2A     LDA  (2A),Y      X1$ NACH $56 FF.
028A  85 56     STA  56          X2$ NACH $58 FF.
028C  B1 01     LDA  (01),Y
028E  85 58     STA  58
0290  C8        INY
0291  B1 2A     LDA  (2A),Y
0293  85 57     STA  57          X1$: HI
0295  B1 01     LDA  (01),Y
0297  85 59     STA  59          X2$: HI
0299  A0 02     LDY  #02
029B  B1 2A     LDA  (2A),Y      LAENGE VON X1$ SPEICHERN
029D  8D 82 03  STA  0382
02A0  B1 01     LDA  (01),Y      LAENGE VON X2$ SPEICHERN
02A2  8D 83 03  STA  0383
02A5  A9 00     LDA  #00
02A7  8D 81 03  STA  0381        CARRY INITIALISIEREN
02AA  A2 12     LDX  #12         ERSTE ADDITION:
02AC  20 7A 03  JSR  037A        LETZTE STELLE IST PFENNIGE
02AF  20 43 03  JSR  0343
02B2  20 7A 03  JSR  037A        ZWEITE ADDITION:
02B5  20 43 03  JSR  0343        VORLETZTE STELLE = GROSCHEN
02B8  20 7A 03  JSR  037A        ZAEHLER VERMINDERN UND
02BB  A9 2E     LDA  #2E         PUNKT ABSPEICHERN
02BD  9D 87 03  STA  0387,X
02C0  CA        DEX
02C1  CE 82 03  DEC  0382        SCHLEIFE UEBER MARK-BETRAAGE
02C4  30 0B     BMI  02D1        TEST AUF ENDE DER STRINGS
02C6  CE 83 03  DEC  0383
02C9  30 2A     BMI  02F5
02CB  20 43 03  JSR  0343        ADDIEREN
02CE  4C C1 02  JMP  02C1        WEITER IN DER SCHLEIFE
02D1  AC 83 03  LDY  0383        LAANGE X1$ ≤ LANGE X2$
02D4  88        DEY
02D5  10 0A     BPL  02E1        TEST AUF =
02D7  AD 81 03  LDA  0381
02DA  C9 00     CMP  #00
02DC  F0 37     BEQ  0315
02DE  4C 1C 03  JMP  031C
02E1  18        CLC
02E2  B1 58     LDA  (58),Y      ZU DEN RESTLICHEN STELLEN VON
02E4  20 51 03  JSR  0351        X2$ WIRD CARRY ADDIERT
02E7  88        DEY
02E8  30 2B     BMI  0315
02EA  B1 58     LDA  (58),Y
02EC  20 51 03  JSR  0351
02EF  88        DEY
02F0  10 F8     BPL  02EA
02F2  4C 15 03  JMP  0315
02F5  AC 82 03  LDY  0382        'ANGE X1$ > LANGE X2$
02F8  10 0A     BPL  0304
02FA  AD 81 03  LDA  0381
```

```
02FD  C9 00     CMP  #00
02FF  F0 14     BEQ  0315
0301  4C 1C 03  JMP  031C
0304  18        CLC
0305  B1 56     LDA  (56),Y      ZU DEN RESTLICHEN STELLEN VON
0307  20 51 03  JSR  0351        X1$ WIRD CARRY ADDIERT
030A  88        DEY
030B  30 08     BMI  0315
030D  B1 56     LDA  (56),Y
030F  20 51 03  JSR  0351
0312  88        DEY
0313  10 F8     BPL  030D
0315  A9 00     LDA  #00
0317  CD 81 03  CMP  0381        FALLS AM ENDE ALLER ADDITIONEN
031A  F0 06     BEQ  0322        CARRY=1, MUSS DER STRING
031C  A9 31     LDA  #31         MIT 1 ANFANGEN
031E  9D 87 03  STA  0387,X      OPERATION:"1" + AA$
0321  CA        DEX
0322  18        CLC
0323  A0 11     LDY  #11         STARTADRESSE DES ERGEBNISSTRINGS
0325  8A        TXA              UNTER AA$ ABSPEICHERN
0326  69 03     ADC  #03
0328  6D 85 03  ADC  0385
032B  91 2A     STA  (2A),Y      LOW
032D  C8        INY
032E  AD 86 03  LDA  0386
0331  69 00     ADC  #00
0333  91 2A     STA  (2A),Y      HIGH
0335  8E 84 03  STX  0384
0338  A9 12     LDA  #12
033A  38        SEC
033B  ED 84 03  SBC  0384
033E  A0 10     LDY  #10
0340  91 2A     STA  (2A),Y      LÄNGE DES ERGEBNISSTRINGS
0342  60        RTS              UNTER AA$ ABSPEICHERN
0343  AC 82 03  LDY  0382        UNTERPROGRAMM ZUR ADDITION
0346  B1 56     LDA  (56),Y      ZWEIER ZIFFERN, DIE IN ASCII
0348  AC 83 03  LDY  0383        DARGESTELLT SIND
034B  18        CLC
034C  71 58     ADC  (58),Y      ZWEITE ZIFFER ADDIEREN
034E  38        SEC
034F  E9 30     SBC  #30         $30 ="0" IN ASCII ABZIEHEN
0351  18        CLC              ROUTINE ZUR ADDITION DES CARRY
0352  6D 81 03  ADC  0381
0355  8D 84 03  STA  0384        ERGEBNIS ZWISCHENSPEICHERN
0358  A9 00     LDA  #00         CARRY ZURÜCKSETZEN
035A  8D 81 03  STA  0381
035D  A9 39     LDA  #39         TEST AUF CARRY
035F  CD 84 03  CMP  0384
0362  B0 0E     BCS  0372
0364  A9 01     LDA  #01         FALLS ÜBERLAUF: CARRY=1
0366  8D 81 03  STA  0381
0369  AD 84 03  LDA  0384
036C  38        SBC              UND 10 ABZIEHEN
036D  E9 0A     SBC  #0A
036F  4C 75 03  JMP  0375
```

```
0372  AD 84 03  LDA   0384          ERGEBNIS VOM ZWISCHENSPEICHER HOLEN
0375  9D 87 03  STA   0387,X       UND IM ERGEBNISSTRING ABSPEICHERN
0378  CA        DEX
0379  60        RTS
037A  CE 82 03  DEC 0382          ZÄHLER AUF DIE ZU ADDIERENDE STELLE
037D  CE 83 03  DEC 0383          VERMINDERN
0380  60        RTS
0381  00                          CARRY
0382  00                          LÄANGE DES STRINGS X1$
0383  00                          LÄNGE DES STRINGS X2$
0384  00                          ZWISCHENSPEICHER
0385  87 03                       ADRESSE DES BEGINNS DER SPEICHERUNG
                                  DES ERGEBNISSTRINGS
```

Das Programm erlaubt im Ergebnis Beträge bis zu 18 Stellen = 10^{15} DM. Es ersetzt die Zeilen 9400-10400 durch folgende Anweisungen:

```
SYS 634
AA  = AA  + ""
RETURN.
```

Durch die mittlere Anweisung wird die Zeichenfolge, die das Ergebnis enthält, aus dem von der Assemblerroutine angelegten Bereich herauskopiert und damit vor dem Überschreiben beim nächsten Aufruf gerettet. Addiert man 8-stellige Beträge, so benötigt man mit dieser Routine etwa die doppelte Zeit wie die eingebaute Gleitkommaaddition.

Eine Alternative besteht im Packen der Zeichenfolgen, d.h. in der Darstellung einer Ziffer in 4 Bit und damit in der Darstellung von 2 Ziffern in einem Byte. Da der Prozessor die dezimale Addition und Subtraktion mit gepackten Zahlen kennt, könnte man das Problem damit effektiv angehen. Es wäre dann aber notwendig, eine PACK-Routine zu schreiben, da die Zeichenfolgen erst gepackt, dann verarbeitet und vor dem Ausdruck wieder in ASCII-Folgen umgewandelt werden müssen.

(Anmerkung des Herausgebers: Ein dezimal rechnendes kaufmännisches Unterprogrammpaket wurde aus der Feder von Michael Zimmermann bereits in den Heften ab Nr. 2 als 'ASP, Advanced Subroutine Package' abgedruckt. Die im Zusammenhang interessierenden Abschnitte finden sich in den Heften 2-6, also auch im Buch 1-6. Was hierzu bisher fehlt, ist die Einbindung des ASP in das BASIC von CBM und z.B. auch AIM 65. Der Herausgeber ist gerne bereit, eine solche Synthese aus den vorliegenden Vorschlägen zu veröffentlichen. Es sei weiterhin darauf verwiesen, daß mit dem SYS-Befehl weitere Parameter übergeben werden können, wie z.B. in diesem Heft im Artikel '1/4 Grafik in Maschinensprache' von H. Brettin beschrieben.) ¤

Buchbesprechung

mikrocomputer anwendungen, Sonderheft 33 der FUNKSCHAU, Franzis-Verlag, München 1980, 82 Seiten, DM 15,60. Neben Programmen für HP- und TI-Taschenrechner enthält dieses Sonderheft schwerpunktmäßig Software für PET/CBM und für AIM 65/PC 100. Für die Commodores sind folgende Artikel erwähnenswert: Plotten mit dem CBM-Drucker, PET als Logikanalysator, PET erzeugt DATA-Zeilen, RESTORE-N-Befehl, Eingabe und Auslesen von Funktionen bei PET/CBM. Als besonders nützlich mögen sich die großen Tabellen 'ROM und RAM bei PET und CBM' erweisen.
Für den AIM 65 sind u.a. abgedruckt:Single Step im BASIC, Formatierte Assemblerausgabe auf Baudot-Fern-Schreiber, Sprache aus dem Computer, Fernschreiber als AIM-Drucker, RENUMBER, RESTORE Line Number, Auto-Start in BASIC, Kansas-City Tonbandformat für AIM. - Allgemeinere Artikel behandeln virtuelle Mehrfachtabellen in BASIC, den Leerstring im Microsoft-BASIC. In der Summe: Viel Information, viele Programme und Anregungen, sehr zu empfehlen. ¤

Peter Trübger, Hamburg

Grafik-Zugriff bei CBM-Busy

Viele Programmierer vermissen bei den Business-Tastaturen der Commodore-Rechner die Möglichkeit, alle Grafikzeichen per Tastendruck zu erzeugen. Bildschirmmasken lassen sich nur umständlich herstellen. Nachfolgend ist eine kleine Routine beschrieben, mit der 31 Grafikzeichen zusätzlich direkt erreicht werden können.

Der IRQ wird mit SYS 826 so versetzt, daß bei jedem Interrupt zunächst die Programmadresse 'START' auf den Stack gepushed wird. Kehrt der Rechner vom Interrupt zurück, so wird das Programm abgearbeitet. Dieser Kunstgriff erlaubt eine gezielte Veränderung im Tastaturbuffer. Die ESC-Taste fungiert als AUS-/EIN-Schalter für den Grafiksatz. Die Cursor-Steuerung arbeitet normal, wie auch die Groß- und Kleinschreibung.

Die Abfrage des Tastaturbuffers nach jedem Interrupt eröffnet speziell bei BASIC 4 reizvolle Möglichkeiten für das Editieren. Man könnte die ESC-Taste als Vortaste für die Editierfunktionen Scroll up, Delete line, Insert line usw. benutzen. BASIC 4 hat im $E000-ROM noch viel Platz für diese Änderungen.

```
                    0010 ;GRAFIK FÜR CBM-BUSY
                    0020          .LS
                    0030          .CE
                    0040          .OS
                    0050 ;-----------------------
                    0060 FLAG     .DE $00
                    0070 CINV     .DE $90
                    0080 ;-----------------------
                    0090 KEYBUF   .DE $026F          ;TASTENBUFFER
                    0100 KEY      .DE $E455          ;INTERRUPT
                    0110 PREND    .DE $E600          ;EXIT INTERRUPT
                    0120 ;-----------------------
                    0130          .BA $033A
033A- 78            0140 IRQSET   SEI
033B- A9 45         0150          LDA §L,NEUIRQ
033D- A2 03         0160          LDX §H,NEUIRQ
033F- 85 90         0170          STA *CINV
0341- 86 91         0180          STX *CINV+1
0343- 58            0190          CLI
0344- 60            0200          RTS
0345- A9 03         0210 NEUIRQ   LDA §H,START
0347- 48            0220          PHA
0348- A9 52         0230          LDA §L,START
034A- 48            0240          PHA
034B- 08            0250          PHP
034C- 08            0260          PHP
034D- 08            0270          PHP
034E- 08            0280          PHP
034F- 4C 55 E4      0290          JMP KEY
0352- AD 6F 02      0300 START    LDA KEYBUF
0355- C9 1B         0310          CMP §$1B           ;ESC-TASTE?
0357- D0 08         0320          BNE WEITER
0359- AA            0330          TAX
035A- A5 00         0340          LDA *FLAG
035C- 49 80         0350          EOR §$80
035E- 85 00         0360          STA *FLAG          ;FLAG FLIP-FLOP
0360- 8A            0370          TXA
```

```
0361- A6 00     0380 WEITER     LDX *FLAG
0363- E8        0390            INX
0364- 30 0A     0400            BMI END
0366- C9 20     0410            CMP §$20
0368- 90 06     0420            BCC END
036A- C9 40     0430            CMP §$40
036C- B0 02     0440            BCS END
036E- 69 80     0450            ADC §$80            ;GRAFIK!
0370- 8D 6F 02  0460 END        STA KEYBUF
0373- 4C 00 E6  0470            JMP PREND
                0480            .EN
END OF MAE PASS!


 ---  LABEL FILE:  ---

CINV =0090              END =0370               FLAG =0000
IRQSET =033A            KEY =E455               KEYBUF =026F
NEUIRQ =0345            PREND =E600             START =0352
WEITER =0361
//0000,0376,0376
```

Horst Brettin, Berlin 44

1/4-Grafik für CBM 3001

Das Programm Lissajous-Figuren dient der Demonstration der Maschinenroutine 1/4-Grafik. Es ist so angelegt, daß beide Teile gemeinsam geladen werden können. Die Maschinenroutine wird mit SYS 933,X,Y aufgerufen. Dabei gibt X (0...79) die Position der Schreibstelle als Abstand vom linken und Y (0...49) den Abstand vom oberen Bildschirmrand an. Wenn die Werte für X bzw. Y außerhalb des in Klammern jeweils angegebenen Bereiches liegen, so kommt die Fehlermeldung 'ILLEGAL QUANTITY ERROR'.

Wie aus der Zeile 180 des BASIC-Listings zu sehen, können für X und Y auch ganze numerische Ausdrücke eingesetzt werden. Diese Ausdrücke werden von der Routine 54988 (hex D6CC) des Interpreters ausgewertet.Sie prüft als erstes, ob ein Komma folgt und übergibt ein Byte im X-Register der CPU. - Ab 59208 (hex E748) hat das Betriebssystem eine Tabelle der Low-Bytes der Bildschirmanfänge. Die High-Bytes stehen in der Zeropage ab 224 (hex E0), hier hat aber das Bit 7 eine Sonderstellung, es wird deshalb hier mit ORA# $80 immer auf 1 gesetzt. - Die Tabelle von 1002 bis 1017 (hex 03EA-03F9) enthält die verwendeten Grafikzeichen in einer ausgeklügelten Reihenfolge.

Das BASIC-Programm enthält keine Besonderheiten. Der Aufbau der Lissajous-Figur kann durch Tastendruck gestoppt werden. Auf einen weiteren Tastendruck fährt das Programm fort oder, falls ein 'N' gedrückt wurde, beginnt es von vorn. Das Programm stoppt nicht, wenn die Figur fertig ist!

```
100 REM (C) BY HORST BRETTIN
110 PRINT"□          LISSAJOUS - FIGUREN■":INPUT"FREQUENZVERHAELTNIS";V
120 PRINT TAB(14);:INPUT"PHASE";P:P=P*π/180
130 X=0:IF V>30 THEN DX=π/720:GOTO 170
```

```
140 IF V>10 THEN DX=π/450:GOTO 170
150 IF V>2 THEN DX=π/360:GOTO 170
160 DX=π/180
170 PRINT"□
180 SYS 933,40.5+39*SIN(V*X+P),25.5-24*SIN(X):X=X+DX
190 GET A$:IF A$="" THEN 180
200 WAIT158,1:GET A$:IF A$="N" THEN 110
210 GOTO 180
```

```
03A5:  20 CC D6   JSR  $D6CC      GETBYTE  X
03A8:  E0 50      CPX  #$50       ≥80 ?
03AA:  B0 3C      BCS  $03E8
03AC:  86 DD      STX  $DD        RETTEN
03AE:  20 CC D6   JSR  $D6CC      GETBYTE Y
03B1:  E0 32      CPX  #$32       ≥50 ?
03B3:  B0 33      BCS  $03E8
03B5:  8A         TXA
03B6:  4A         LSR             HALBIEREN
03B7:  AA         TAX
03B8:  A9 01      LDA  #$01       OPERATOR VORBEREITEN
03BA:  90 02      BCC  $03BE      UND ERZEUGEN
03BC:  0A         ASL
03BD:  0A         ASL
03BE:  46 DD      LSR  $DD        HALBIEREN
03C0:  90 01      BCC  $03C3
03C2:  0A         ASL
03C3:  A4 DD      LDY  $DD        X INS Y-REG
03C5:  85 DD      STA  $DD        OPERATOR ABLAGEN
03C7:  BD 48 E7   LDA  $E748,X    L-BYTE DER BILDSCHIRMZEILE
03CA:  85 11      STA  $11
03CC:  A9 80      LDA  #$80
03CE:  15 E0      ORA  $E0,X      H-BYTE DER BILDSCHIRMZEILE
03D0:  85 12      STA  $12
03D2:  A2 0F      LDX  #$0F
03D4:  B1 11      LDA  ($11),Y    ALTES GRAFIKZEICHEN AUS BILD-
03D6:  DD EA 03   CMP  $03EA,X    SCHIRM IN TABELLE SUCHEN
03D9:  F0 03      BEQ  $03DE
03DB:  CA         DEX
03DC:  D0 F8      BNE  $03D6
03DE:  8A         TXA             ALS OPERATOR IN DEN AKKU
03DF:  05 DD      ORA  $DD        MIT ABGELEGTEM OPERATOR
03E1:  AA         TAX             VERKNÜPFEN
03E2:  BD EA 03   LDA  $03EA,X    NEUES GRAFIKZEICHEN AUS TABELLE
03E5:  91 11      STA  ($11),Y    AUF DEN BILDSCHIRM
03E7:  60         RTS
03E8:  4C 23 D1   JMP  $D123      ILLEGAL QUANTITY ERROR
```

□

Tastentest für CBM 3032. Der in Heft 16 auf Seite 53 abgedruckte Artikel mit diesem Namen stammt ebenfalls aus der Feder von Horst Brettin, Berlin. Der Herausgeber bittet das versehentliche Fortlassen des Namens dort zu entschuldigen.

Dr. A. Schnell, Aachen

AIM 65 mit 'fremder' Systemsoftware

Beschrieben wird eine Kombination des Rockwell AIM 65 mit Systemsoftware von Ohio Scientific, welche nach einigen Änderungen ein sehr leistungsfähiges Disk System mit Druckerausgabe ergibt. An Hardware wird für ein solches System folgendes benötigt: Neben dem AIM 65 ein externer Speicher von mindestens 20 K Byte, eine Floppy-Steuerung, ein Bootstrapprogramm und eine OSI Floppy Disk mit Systemdiskette. Zur Abrundung ist ein über DILINK gesteuertes Videointerface noch sehr nützlich, aber nicht unbedingt erforderlich.

Interface zur Floppy Disk

Die Hardware dieses Interfaces wurde so einfach wie möglich gehalten und entspricht in der Funktion dem 610 Board von OSI. Ein Selbstbau ist problemlos möglich, da Justierarbeiten sehr leicht durchführbar sind. Abb. 1 zeigt die Schaltung schematisch. Es wird ein PIA 6820 (auch 6520) eingesetzt, welcher die Steuerung der Floppy übernimmt, ein ACIA 6850 ist für die serielle Datenübertragung zuständig. Neben diesen beiden Bausteinen, die über Adreß- und Datenbus nach geeigneter Dekodierung und Pufferung im Adreßbereich C0xx angesprochen werden, werden nur noch einige Zeitglieder für die seriellen Daten benötigt. Abb. 2 zeigt die vollständige Schaltung. Sie wird parallel zur Speichererweiterung am Speichererweiterungsstecker angeschlossen.

Die Einstellung der Widerstände R1, R2, R3 und R4 ist relativ unkritisch. Ausgehend von der Mittelstellung müssen die an den Testpunkten TP1 bis TP4 mit einem Oszilloskop gemessenen Kurvenformen der Abb. 3 entsprechen (bei Datenausgabe bzw. in der Einlesephase, die ja mit dem Bootstrapprogramm bewirkt wird). Diese Abbildung erklärt auch die Art der seriellen Datenübertragung, bei der Takt und Datenstrom kombiniert werden. Abb. 4 zeigt die Belegung des Steckers zur OSI Floppy. Dies ist ein gewöhnliches Minifloppylaufwerk, so daß unter Umständen auch ein anderes Fabrikat benutzt werden kann. Allerdings beinhaltet das Laufwerk noch einen sogenannten Dataseparator, der bei den meisten Minifloppylaufwerken als Option angeboten wird und der für ein Separieren von Takt und Daten bei den gelesenen Informationen sorgt. Mehr Hardware wird nicht benötigt.

Systemsoftware

Die zum OSI Floppydisklaufwerk mitgelieferte Systemdiskette enthält neben dem Disk Operating System DOS das Microsoft-BASIC, verschiedene Utility-Programme, Assembler, Relocator und einen Extended Monitor. Diese Software muß allerdings erst einmal in den Speicherbereich des AIM 65 übertragen werden, damit sie die Kontrolle über den AIM 65 übernehmen kann. Dazu wird ein kurzes Maschinenprogramm benötigt, ein sog. Bootstrap Loader. Dieser kann z.B. von der Kassette in den oberen RAM-Bereich geladen und vom AIM Monitor aus gestartet werden. Besser ist es jedoch, dieses kurze Programm im EPROM zu haben, das in die für den Assembler vorgesehene ROM-Fassung gesteckt wird. So wird im Monitor Mode des AIM 65 nach Drücken der Taste N zu der Adresse D000 gesprungen und, wenn alles klappt, das System geladen. PROG.1 ist das kommentierte Bootstrap Programmlisting, hier für das EPROM ab Adresse D000 wiedergegeben.

Bis jetzt sind allerdings erst die Anfangsschwierigkeiten überwunden, man darf nämlich nicht erwarten, daß die OSI Software ohne weiteres mit der AIM Hardware zusammenarbeitet. Ein glücklicher Umstand ist es allerdings, daß sich die einzelnen Speicherbereiche nicht gegenseitig überschneiden. So wird von der OSI Software der Speicherbereich des AIM Monitors nicht benötigt und die Floppy I/O-Adressen (C0xx) werden vom AIM Monitor nicht verwendet. Die I/O-Adressen des AIM Monitors stören wiederum nicht die OSI Software.

Was passiert denn nun eigentlich, wenn die Hardware vorhanden ist, das Bootstrapprogramm geladen oder im EPROM ist und wenn nach dem Einschalten das Bootstrapprogramm z.B.

durch die Taste N angesprungen wird? Zunächst initialisiert der Bootstrapprogramm das Floppy Disk-Laufwerk und den PIA- und ACIA-Baustein auf der Interfaceplatine. Dann wird von der Floppy das Disk Operating System DOS geladen und im DOS eine Routine angesprungen, die das Microsoft-BASIC von der Floppy lädt. Dies macht sich akustisch durch verschiedene Diskettenoperationen bemerkbar. Danach wird automatisch das Programm BEXEC* geladen und ausgeführt. Dieses BASIC-Programm erwartet eine Eingabe und deshalb wird das nicht vorhandene OSI Keyboard an der Adresse DF00 abgefragt. Da es nicht vorhanden ist, wartet das Programm beliebig lange in einer Warteschleife. Es passiert also nichts mehr, das Display bleibt dunkel, und es erfolgt keine Reaktion auf Tastendrücke am AIM.

Glücklicherweise ist aber der AIM Monitor durch einen RESET wieder erreichbar, die OSI Systemsoftware wird hierbei nicht gelöscht. Nach dem RESET hat man also wieder volle Kontrolle über den AIM und das Disk Operating System. Vom AIM Monitor kann man jetzt also die Ein- und Ausgaberoutinen des DOS und des BASIC so verändern, daß als Input das AIM Keyboard abgefragt wird und als Ausgabe das AIM Display, der AIM Drucker oder ein angeschlossenes Videointerface bedient wird. Dazu muß man allerdings wissen, wo die bewußten Routinen abgespeichert sind. Mit dem AIM Disassembler ließ sich dies mit viel Geduld herausfinden. Es ergab sich, daß in einer Input-/Outputtabelle die Anfangsroutinen verschiedener Systemroutinen abgespeichert sind, die das OSI Keyboard, das OSI Display, die seriellen Ausgänge usw. bedienen. Diese Adressen brauchten lediglich so verändert zu werden, daß sie auf eigene I/O Routinen oder die entsprechenden Routinen im AIM Monitor zeigen. Nach diesen Änderungen läßt sich dann die Kontrolle an das OSI Betriebssystem zurückgeben über *=2A51 und G/. Nun reagiert in der Tat das 'fremde' DOS auf die Hardware des AIM, und man kann mit dem System arbeiten.

Dies wäre nun eine recht umständliche Prozedur, wenn sie nach jedem Neuladen des Systems erneut vollzogen werden müßte. Am einfachsten wäre es, wenn die Änderungen der OSI Software automatisch durch das BEXEC* Programm nach dem Systemladen durch entsprechende POKEs erfolgen könnte. Dies ist in der Tat möglich, da ja BEXEC* als erstes Programm automatisch ausgeführt wird, bevor irgendeine Referenz zu einer I/O-Adresse gemacht wird. Über entsprechende DATA-Anweisungen und POKEs in den OSI-Systembereich führt dann das BASIC-Programm die notwendigen Änderungen selbst aus. Als Benutzer merkt man nichts von diesem Prozeß; man schließt die Floppy mit ihrem Interface an, drückt N und nach dem Ladeprozeß meldet das System sich mit einem Ausdruck gemäß Abb. 5 auf dem AIM-Drucker, wobei gleich noch das Directory der Disk mit ausgedruckt werden kann.

Die Programme, die mit dem AIM-Kleinassembler zum ersten Mal eingetippt werden müssen, und die veränderten I/O-Vektoren sind in entsprechenden POKEs und DATA Anweisungen im Programm BEXEC* enthalten, welches mit PUT "BEXEC*" das ursprüngliche BEXEC* auf der Systemfloppy überschreibt. PROG.2 ist ein Listing dieses Programmes mit dem beschriebenen System auf dem AIM-Drucker mit LIST#5.

Wenn alle diese Änderungen ausgeführt worden sind, hat man mit dem OSI Microsoft BASIC und dem AIM hervorragende I/O-Möglichkeiten. Mit PRINT#2 wird der Bildschirm über ein seriell angesteuertes Videointerface angesprochen, PRINT#3 druckt nur auf dem AIM-Display, während PRINT#4 auf dem AIM-Drucker ausgibt. PRINT# 5 und PRINT#6 schreiben auf die Floppy. Der Input erfolgt bislang nur von der Floppy, dem AIM-Keyboard oder einem reservierten Speicherbereich. Weitere I/O-Möglichkeiten sind möglich und denkbar, z.B. Input oder Output von der Cassette im AIM- oder KIM-Format.

Da der BASIC-Interpreter von 0200 bis 2085 hex abgespeichert ist, läßt sich seine Arbeitsweise im STEP-Mode beobachten. Da er im RAM-Bereich liegt, läßt sich durch Änderung einzelner Speicherstellen die Wirkung einer Änderung auf den Ablauf sehr schön studieren. Auf diese Weise ist eine Analyse des Microsoft BASIC möglich.

(Hinweis des Herausgebers: Es sollte eigentlich doch auch möglich sein, eine dem AIM angepaßte Systemdiskette zu schaffen? Dieser Artikel eröffnet nicht nur weitere Floppy-Möglichkeiten für den AIM, sondern ist zugleich eine schöne Demonstration, daß man die Software anderer 6502-Systeme benutzen kann, wenn man die E/A-Routinen anpaßt.)

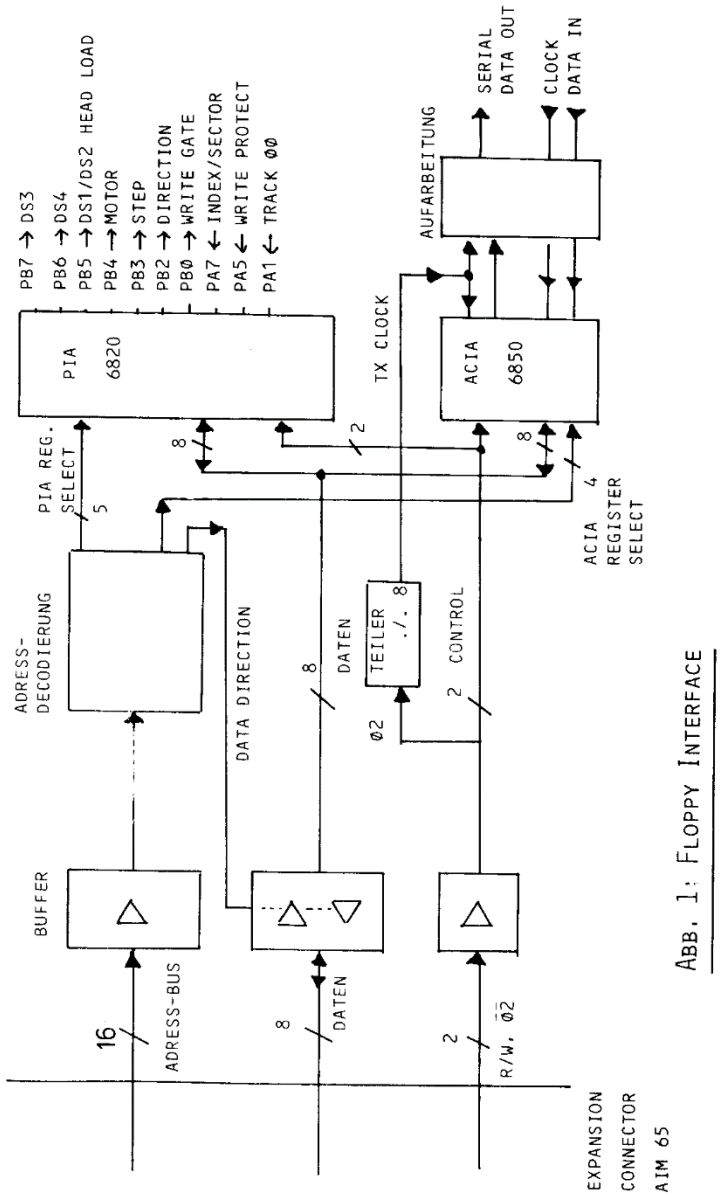


ABB. 1: FLOPPY INTERFACE

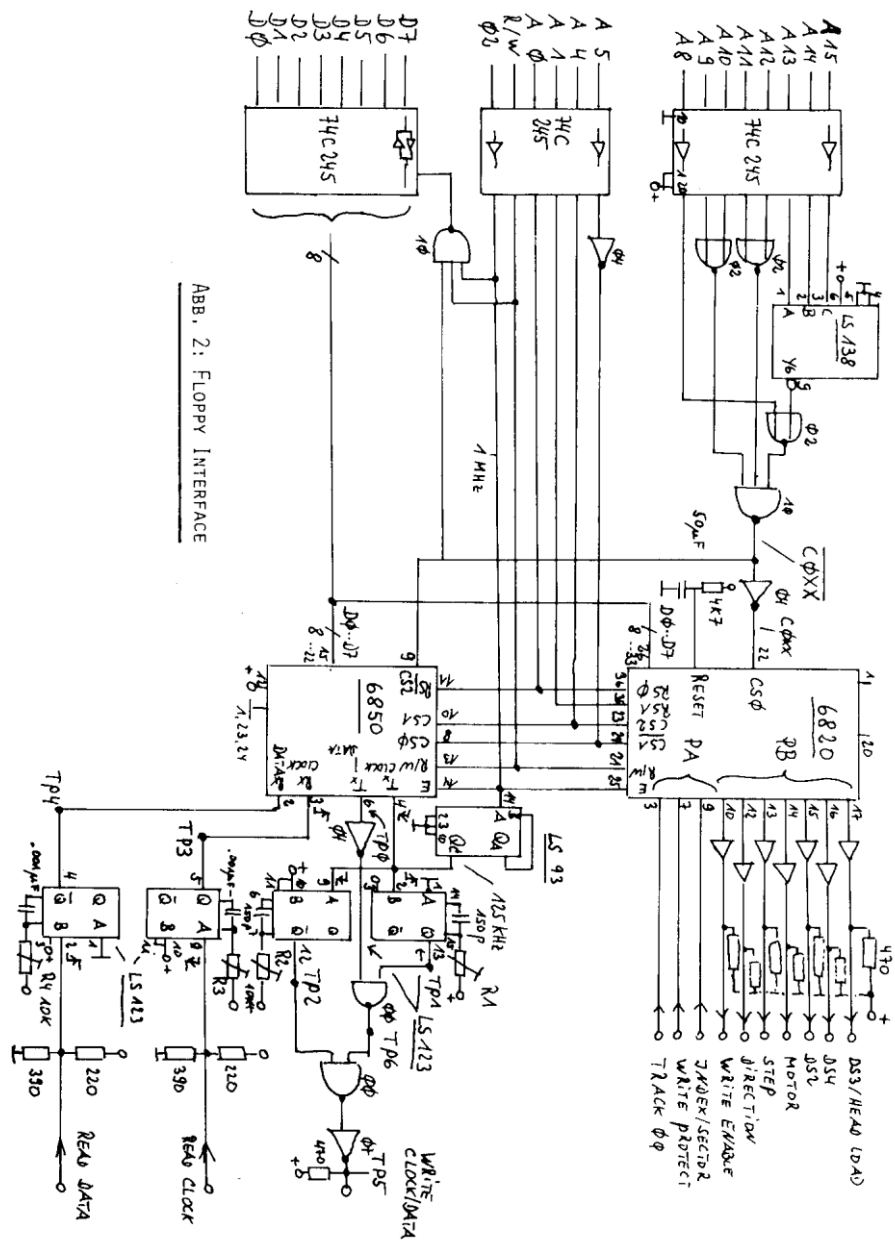


ABB. 2: FLOPPY INTERFACE

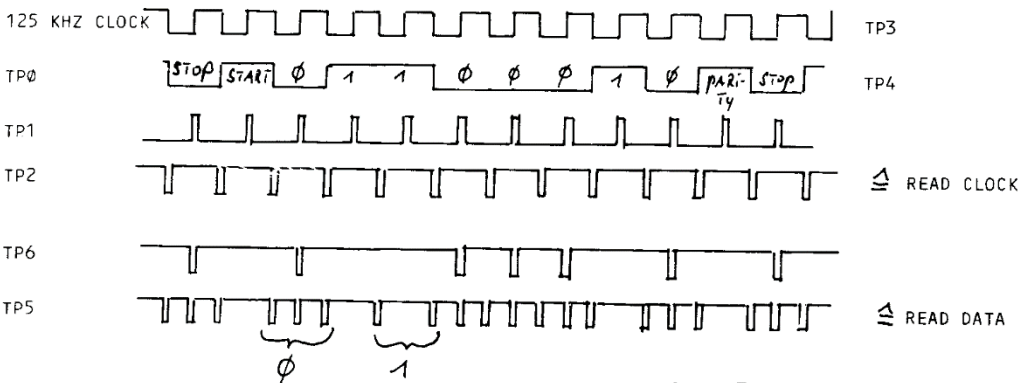


ABB. 3: IMPULSDIAGRAMME

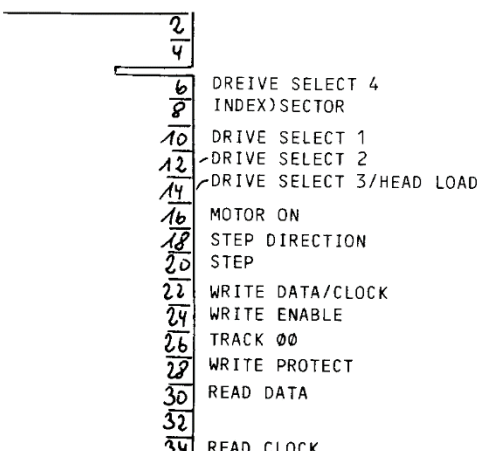


ABB. 4: STECKERBELEGUNG DER OSI-FLOPPY

PROG.1 BOOTSTRAP LOADER

```
D000 20 JSR D00C
D003 6C JMP (00FD)
D006 20 JSR D00C
D009 6C JMP (FFFC)

D00C A0 LDY #00
D00E 8C STY C001
D011 8C STY C000      INITIALISIERUNG
D014 A2 LDX #04       PIA
D016 8E STX C001
D019 8C STY C003
D01C 88 DEY
D01D 8C STY C002
D020 8E STX C003
D023 8C STY C002
D026 A9 LDA #FB
D028 D0 BNE D033
D02A A9 LDA #02
D02C 2C BIT C000
D02F F0 BEQ D04D
D031 A9 LDA #FF
D033 8D STA C002
D036 20 JSR D0A5
D039 29 AND #F7
D03B 8D STA C002
D03E 20 JSR D0A5
D041 09 ORA #08
D043 8D STA C002
D046 A2 LDX #18
D048 20 JSR D091
D04B F0 BEQ D02A
D04D A2 LDX #7F
D04F 8E STX C002
D052 20 JSR D091
D055 AD LAD C000
D058 30 BMI D055

D05A AD LDA C000      WARTET AUF
D05D 10 BPL D05A      INDEXHOLE
D05F A9 LDA #03
D061 8D STA C010      INITIALISIERUNG
D064 A9 LDA #58       ACIA
D066 8D STA C010
D069 20 JSR D09C
D06C 85 STA FE        STARTADRESSE
D06E AA TAX           HIGH
D06F 20 JSR D09C      STARTADRESSE LOW
D072 85 STA FD
D074 20 JSR D09C
D077 85 STA FF        ANZAHL PAGES
D079 A0 LDY #00
D07B 20 JSR D09C
D07E 91 STA (FD),Y    IN MEMORY
D080 C8 INY           LADEN
D081 D0 BNE D07B
D083 E6 INC FE
D085 C6 DEC FF
D087 D0 BNE D07B
D089 86 STX FE
D08B A9 LDA $FF
D08D 8D STA C002      PIA RESET
D090 60 RTS
D091 A0 LDY #F8
D093 88 DEY
D094 D0 BNE D093      VERZOGERUNG
D096 55 EOR FF,X
D098 CA DEX
D099 D0 BNE D091
D09B 60 RTS

D09C AD LDA C010
D09F 4A LSR .A
D0A0 90 BCC D09C      BYTE VON
D0A2 AD LDA C011      ACIA LESEN
D0A5 60 RTS
```

PROG.2

```
1 REM DIRECTORY UTILITY, BEXEC:

2 POKE8993,2:POKE8994,2                  I/O FLAG

3 POKE9522,174:POKE9523,37               INPUT VECTOR #2
4 POKE741,76:POKE750,78                  NEW, LIST ENABLE
5 POKE2893,55:POKE2894,8                 REDO FROM START ENABLE
6 POKE268,76:POKE269,81:POKE270,42       F1 VECTOR = STOP TASTE

7 POKE8981,183:POKE8982,37:POKE8983,192:POKE8984,37   OUTPUT VECTOR #3
                                                      UND #4
8 POKE42001,0                            PRINTER OFF
12 DATA41,127,201,13,240,249,201,10
13 DATA208,5,32,168,238
14 DATA169,13,32,0,240,76,168,238        DEFAULT OUTPUT #2 (VIDEO)
16 DATA32,60,233,201,127,208,2,169,95,96          INPUT #2 (KEYBOARD)
20 FORI=9625TO9655:READD:POKEI,D:NEXT
22 DATA41,127,201,10,240,218,76,5,239
```

```
 23 DATA162,128,142,17,164,41,127,201,13,240,204,201,10  OUTPUT #3 (DISPLAY)
24 DATA208,2,169,13,32,0,240,162,0,142,17,164,96         #4 (PRINTER)

 26 FORI=9656TO9690:READD:POKEI,D:NEXT

 27 DATA96,127,141,128,164,173,130,164,42,176,244,56,169,0,2   F1= STOP TASTE
28 FORI=2073TO2087:READD:POKEI,D:NEXT

 29 POKE2073,169
 30 NF=0:PN=11897
 50 DEF FNA(X)=10*INT(X/16)+X-16*INT(X/16)
 80 DV=2:Y=2:X=PEEK(8994)
 10000 REM                                DISK DIRECTORY
 10010 REM PRINT A DIRECTORY OUT
 10020 REM
 10030 PRINT #DV : PRINT #DV,"OS-65D VERSION 3.0"    NACH OUTPUT
 10035 PRINT #DV," -- DIRECTORY --" : PRINT #DV
 10040 PRINT#DV,"FILE     TRACK RANGE"
 10050 PRINT#DV,"--------------------"
 10060 DISK ! "CA 2E79=12,1"
 10070 GOSUB 11000
 10080 DISK ! "CA 2E79=12,2"
 10090 GOSUB 11000
 10130 PRINT#DV:PRINT#DV,NF;"ENTRIES FREE OUT OF 64";
10140 END
 11000 REM
 11010 REM READ DIRECTORY OUT OF BUFFER INTO ARRAYS
 11020 REM
 11040 FOR I=PN TO PN+248 STEP 8
 11050 IF PEEK(I)=35 THEN NF=NF+1 : GOTO 11130
 11060 N$=""
 11070 FOR J=I TO I+5
 11080 N$=N$+CHR$(PEEK(J))
 11090 NEXT J
 11100 PRINT#DV,N$;TAB(11);FNA(PEEK(I+6));TAB(15);"-";
11110 PRINT#DV,TAB(16);FNA(PEEK(I+7))
 11130 NEXT I
 11140 RETURN
READY.
```

```
OS-65D VERSION 3.0
 -- DIRECTORY --
FILE      TRACK RANGE
--------------------
OS65D3      0  - 12
BEXEC*      14 - 14
CHANGE      15 - 16
CREATE      17 - 19
DELETE      20 - 20
DUMMY       26 - 29
RENAME      25 - 25

 57 ENTRIES FREE OUT
 OF 64
```

Abb. 5

```
<K7=2599
/14
2599 29 AND #7F
259B C9 CMP #0D
259D F0 BEQ 2598
259F C9 CMP #0A
25A1 D0 BNE 25A8
25A3 20 JSR D1B7
25A6 A9 LDA #0D
25A8 20 JSR F000
25AB 4C JMP D1B7

25AE 20 JSR E93C
25B1 C9 CMP #7F      #2
25B3 D0 BNE 25B7     KEYBOARD
25B5 A9 LDA #5F
25B7 60 RTS
```

```
<K>*=25B8
/16
25B8 29 AND #7F      #3
25BA C9 CMP #0A      DISPLAY
25BC F0 BEQ 2598
25BE 4C JMP EF05

25C1 A2 LDX #80
25C3 8E STX A411     #4
25C6 29 AND #7F      PRINTER
25C8 C9 CMP #0D
25CA F0 BEQ 2598
25CC C9 CMP #0A
25CE D0 BNE 25D2
25D0 A9 LDA #0D
25D2 20 JSR F000
25D5 A2 LDX #00
25D7 8E STX A411
25DA 60 RTS                  □
```

Dr. Andreas Joss, CH-3012 Bern

FASTLINK (AIM65)

In Heft 16 dieser Zeitschrift, Seite 33 ff. stellte Herr Dr. Joss 'Ein schnelles und sicheres Bandformat für AIM 65' vor. Es benutzt eine geblockte Aufzeichnungsart und eine Schreibgeschwindigkeit von 8000 Baud. Bei einigen Tape-Decks funktioniert es ohne Änderungen an der Hardware, für andere wurden Schaltungsvorschläge gemacht. Mit FASTLINK findet nun eine Anbindung des Text-Editors und des AIM-BASIC an dieses bemerkenswert schnelle Bandformat statt nicht jedoch, wie vom Herausgeber irrtümlich vermerkt, an den Assembler. - Ein weiterer Artikel behandelt in diesem Zusammenhang 'Die (Un-) Zuverlässigkeit von Kassettenspeichern'. (d. Hrsg.).

```
0000          ;ONE PART DUMPS THE EDITOR USING FAST.

0000          ;AFTER RELOADING THE DUMP WITH FASTLOAD, THE EDITOR

0000          ;CAN BE REENTERED WITH 'T'.
0000          ;(THE EDITOR WILL BE INITALIZED TO THE SAME LOCATION

0000          ; AS IT WAS BEFORE DUMP !)
0000

0000          ;THE ROUTINE IS CALLED BY JSR EFAST
0000

0000          ---------------------

0000
0000          ;THE OTHER PART IS CALLED BY BASIC VIA USR(W) OR
0000          ; WITH 'END:#$9100' FROM THE GWK-BASIC-EXPANSION.
0000
0000          ;THIS ROUTINE STORES THE INITIALIZED BASIC-PROGRAMM

0000          ;WITH OR WITHOUT THE VARIABLES AND STRINGS DEPENDING
0000          ;ON THE ANSWER TO "A?" (ALL?)
0000
0000          ;AFTER THE PROGRAM HAS BEEN LOADED WITH FASTLOAD, BASIC IS

0000          ;REENTERED AND THE PROGRAM CAN BE RESTARTED WITH "GOTO XX"

0000          ;(XX = LINE NUMBER), WITHOUT LOOSING THE VARIABLES.
0000 CLR          =$EB44                    0000
0000 OUTPUT       =$E97A                    0000 SNSTRT       =$9455
0000 RED1         =$FE96                    0000 SNSYNC       =$9473
0000 HEX          =$EA7D                    0000 SENDA        =$9537
0000 EQUAL        =$E7D8                    0000 SNDPRT       =$9484
0000 FNAM         =$E8A2                    0000
0000 ADDRS1       =$F910                    0000 SECCNT       =$F8
0000 BLANK        =$E83E                    0000 SECFAC       =$F9
0000 QM           =$E7D4                    0000 MORFLG       =$FC
0000                                        0000 BLKCNT       =$FD
0000 ADDR         =$A41C                    0000
0000 NAME         =$A42E                    0000 ALLFLG       =$F4
0000 DRB          =$A800                    0000
```

```
0000                *=$90D5
90D5 EFAST    20EE8 JSR BLANK      ; DUMP THE EDITOR
90D8          208991 JSR NSTART
90DB          A959   LDA #'Y
90DD          85FC   STA MORFLG
90DF          A2DF   LDX #$DF
90E1          A000   LDY #0
90E3          207691 JSR XYADS1

90E6          A2E6   LDX #$E6
90E8          207C91 JSR STXY      ; DUMP EDITOR-ZERO-PAGE-REG.

90EB          208391 JSR SNXXRT
90EE          A6E3   LDX $E3
90F0          A4E4   LDY $E4
90F2          207691 JSR XYADS1
90F5          A94E   LDA #'N
90F7          85FC   STA MORFLG
90F9          A6E1   LDX $E1
90FB          A4E2   LDY $E2
90FD          4C6791 JMP FINISH    ; DUMP EDITOR-TEXT
9100
9100 BFAST    2044EB JSR CLR       ; DUMP BASIC
9103          20EE8 JSR BLANK
9106          A941   LDA #'A       ; ALL?
9108          207AE9 JSR OUTPUT
910B          20D4E7 JSR QM
910E          20EE8 JSR BLANK
9111          2096FE JSR RED1
9114          85F4   STA ALLFLG
9116          20EE8 JSR BLANK
9119          208991 JSR NSTART
911C          A959   LDA #'Y
911E          85FC   STA MORFLG
9120          A200   LDX #0
9122          A000   LDY #0
9124          207C91 JSR STXY
9127          2010F9 JSR ADDRS1
912A          A2DE   LDX #$DE
912C          207C91 JSR STXY      ; DUMP 00-DE(ZERO-PAGE
912F          208391 JSR SNXXRT
9132          A200   LDX #0        ; DUMP 200-...
9134          A002   LDY #2
9136          207691 JSR XYADS1
9139          A5F4   LDA ALLFLG
913B          C959   CMP #'Y'      ; Y OR J : SEND WHOLE BASIC

913D          F00F   BEQ WHOLE
913F          C94A   CMP #'J
9141          F00B   BEQ WHOLE

9143          A94E   LDA #'N
9145          85FC   STA MORFLG
9147          A675   LDX $75

9149          A476   LDY $76       ; SEND PROGRAMM ONLY

914B          4C6791 JMP FINISH
```

```
914E WHOLE    A679    LDX $79
9150          A47A    LDY $7A

9152          207C91  JSR STXY
9155          208391  JSR SNXXRT      ; SEND PROGRAMM AND VARIABLES
9158          A94E    LDA #'N
915A          85FC    STA MORFLG
915C          A67B    LDX $7B

915E          A47C    LDY $7C

9160          207691  JSR XYADS1
9163          A67F    LDX $7F
9165          A480    LDY $80         ; SEND STRINGS

9167 FINISH   207C91  JSR STXY
916A          208391  JSR SNXXRT
916D
916D TAPON    AD00A8  LDA DRB         ; TURN TAPE 1 ON
9170          0910    ORA #%00010000
9172          8D00A8  STA DRB

9175          60      RTS
9176
9176 XYADS1   207C91  JSR STXY
9179          4C10F9  JMP ADDRS1
917C
917C STXY     8E1CA4  STX ADDR
917F          8C1DA4  STY ADDR+1
9182          60      RTS
9183
9183 SNXXRT   205594  JSR SNSTRT
9186          4C8494  JMP SNDPRT
9189
9189 NSTART
9189 ASKS     A953    LDA #'S'        ; START SIMILAR LIKE FASTDUMP
918B          207AE9  JSR OUTPUT
918E          20D8E7  JSR EQUAL
9191          2096FE  JSR RED1
9194          207DEA  JSR HEX
9197          B0F0    BCS ASKS
9199          85F9    STA SECFAC
919B          85F8    STA SECCNT
919D          A201    LDX #1
919F          20A2E8  JSR FNAM
91A2          205594  JSR SNSTRT
91A5 SNDNM    207394  JSR SNSYNC
91A8          A93A    LDA #':
91AA          203795  JSR SENDA
91AD          A000    LDY #0
91AF DNMLP    B92EA4  LDA NAME,Y
91B2          203795  JSR SENDA
91B5          C8      INY
91B6          C005    CPY #5
91B8          D0F5    BNE DNMLP
91BA          A232    LDX #50
91BC          C6F8    DEC SECCNT
91BE          D0E5    BNE SNDNM

91C0          A296    LDX #150
91C2          207394  JSR SNSYNC
91C5          A900    LDA #0
91C7          85FD    STA BLKCNT
91C9          60      RTS
```

Dipl.-Ing. U. Kornnagel, Ing. grad. G. Krohn und Ing. grad. P. Bach

Datenaustausch zwischen zwei Mikroprozessorsystemen (3)

3. Datenaustausch über ICE-Bus

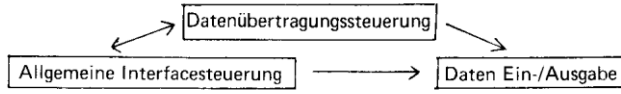
Dieser Bus wurde im Jahre 1976 national und international genormt. Er zeichnet sich durch seine Flexibilität im Datenaustausch verschiedener Intelligenzen aus. Es können bis zu 16 Geräte im Steckdosenprinzip verbunden werden. Die einzelnen Geräte werden über Busleitungen gesteuert und können in beliebiger Folge als TALKER und LISTENER arbeiten.

3.1. Der Bus-Aufbau

Der Bus läßt sich in drei Funktionsgruppen aufteilen:

1. Datenübertragungssteuerung,
2. Datenleitungen,
3. Allgemeine Interfacesteuerung.

Der Bus hat folgende Ablaufhierarchie:



3.1.1 Die Datenübertragungssteuerung

Sie besteht aus 3 Leitungen, nämlich:

1. DAV-Leitung, Data Valid (Daten gültig). Sie wird auf Low gesetzt, wenn gültige Daten an den 8 Datenleitungen anliegen.
2. NRFD-Leitung, Not Ready For Data (nicht aufnahmebereit). Über diese Leitungen werden alle Geräte im Wired-AND miteinander verbunden. Sie geht auf Low, wenn mindestens ein Gerät nicht aufnahmebereit ist.
3. NDAC-Leitung, No Data Accepted (keine Daten übernommen). Diese Leitung wird auf Low gesetzt, wenn alle angesprochenen Geräte die Daten übernommen haben.

3.1.2 Die Datenleitungen

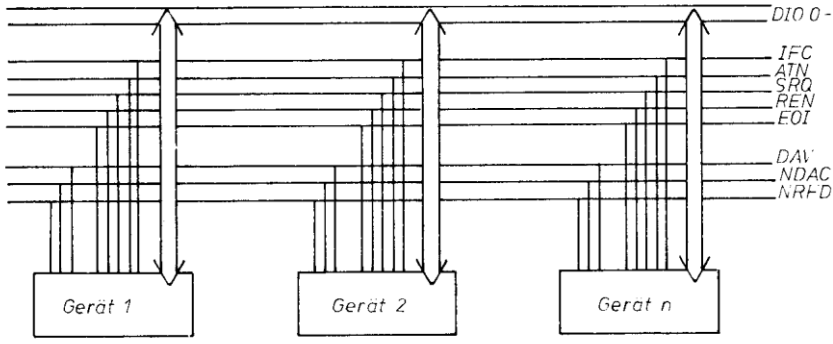
Es handelt sich um 8 Leitungen, die sowohl Daten als auch Befehle übermitteln können, Data Input Output (DIO 0 bis 7).

3.1.3. Die Interfacesteuerleitungen

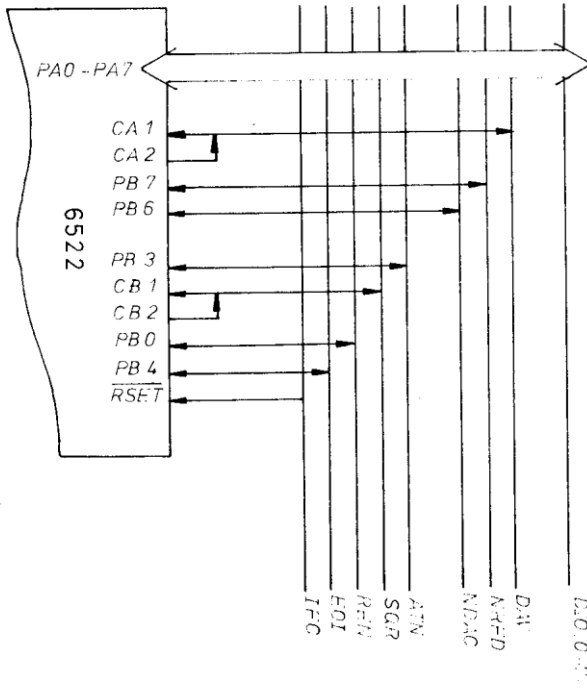
Hierbei handelt es sich um 5 Leitungen, mit deren Hilfe die Geräte gesteuert werden

1. IFC, Interface Clear (Schnittstelle zurücksetzen). Die Leitung dient dazu, bei den am Bus angeschlossenen Geräten einen Reset auszulösen.
2. ATN, Attention (Achtung). Diese Leitung zeigt an, daß an den Datenleitungen jetzt Geräteadressen für die angewählten Geräte anstehen.
3. SRQ, Service Request (Bedienungsruf). Diese Leitung zeigt an, daß ein Gerät von der Zentraleinheit bedient werden möchte.
4. REN, Remote Enable (Fernsteuerung freigeben). Mit dieser Leitung kann die Fernsteuerung über den Bus ein- bzw. ausgeschaltet werden (Lokalbetrieb).
5. EOI, End of Identification (Ende der Kennung). Dieses Signal gibt das Ende der Übertragung an.

3 2 Anschluß der Geräte an den Bus



3.3 ICE-Bus mit VIA

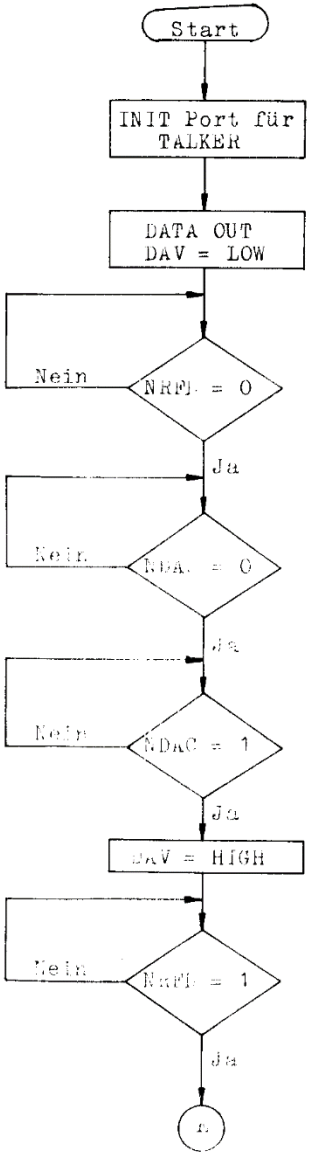


PAØ-PA7 = DIO (Ø-7)
DAV_{IN} = CA1
DAV_{OUT} = CA2
NRFD = PB7
NDAC = PB6
ATN = PB3
SRQ_{IN} = CB1
SRQ_{OUT} = CB2
REN = PBØ
EOI = PB1

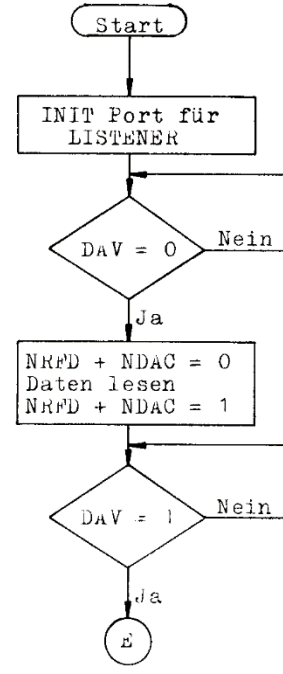
Als Bus-Selekt können PB4 und PB5 verwendet werden.

3.4 Flußdiagramm

TALKER



LISTENER



3.5 Das ICE-Handshake

3.5.1 Der Talker

Im ersten Schritt wird das VIA als TALKER vorbereitet. Danach werden die Daten am Port bereitgestellt, indem DAV auf Low gelegt wird. Durch logisch '0' an NRFD und NDAC wird angezeigt ob die Daten von den LISTENERs der angeschlossenen Geräte übernommen wurden. Wenn alle Geräte die Daten übernommen haben (dann logisch '1' an NDAC), wird durch steigende Flanke an DAV angezeigt, daß die Datenübertragung beendet ist. Danach wartet der TALKER bis alle Geräte für das nächste Byte bereit sind (NRFD='High').

3.5.2 Der LISTENER

Es werden alle Ports für LISTENER-Betrieb vorbereitet. Danach wartet der LISTENER, daß die Leitung DAV auf '0' gezogen wird (gültige Daten liegen jetzt am Port an). Die Leitungen NRFD und NDAC werden von ihm auf Low gesetzt und somit die Daten übernommen. Anschließend werden NDAC und NRFD auf 'High' gelegt. Nach der Übernahme wartet der LIS TENER auf 'HIGH' an DAV um sicherzustellen, daß die Übertragung der Daten abgeschlossen ist.

3.6 Programm

```
TALKER

        ;INIT PORT FÜR TALKER
TALK    LDA #$FF
        STA DDRA
        LDA #$Ø
        STA DDRB

        ;DATENAUSGABE UND DAV LOW
        LDA BUFFER
        STA DRA      ;IN DEN PORT
        LDA PCR
        AND #$FD
        STA PCR
        ;NRFD + NDAC = LOW
        LDA #$FF
LOP1    BIT DRB
        BPL LOP1
        BVS LOP1
        ;DAV = HIGH
        LDA PCR
        ORA #$2
        STA PCR
        ;NRFD = 1
LOP2    BIT DRB
        BMI LOP2
        RTS
```

```
LISTENER

        ;INIT PORT FÜR LISTENER
LISTN   LDA #$Ø
        STA DDRA    ;F. DATENRICHTUNG
        LDA #$FF
        STA DDRB

        ;WARTE BIS CA1=LOW
        LDA #2
LOP1    BIT IFR     ;INTERRUPT FLAG-REG
        BEQ LOP1
        ;NRFD + NDAC=LOW
        LDA #$Ø
        STA DRB
        ;DATEN LESEN
        LDA DRA
        STA BUFFER
        LDA PCR
        ORA #1
        STA PCR
        ;NRFD + NDAC = HIGH
        LDA #$FF
        STA DRB
        ;WARTE, BIS DAV=HIGH
        LDA #2
LOP2    BIT IFR
        BEQ LOP2
        LDA PCR
        AND #$FE
        STA PCR
        RTS                          ¤
```

Dr. Fritz Mayer-Lindenberg, Bielefeld

Rechnerkopplung mit Interrupt

In diesem Aufsatz beschreibe ich ein einfaches Verfahren zur Koppelung von zwei oder mehreren Rechnern (Prozessoren), welches an einer unten näher beschriebenen, auch für sich interessanten Hardwarekonfiguration erprobt wurde. Dabei wird, wie in Bild 1 gezeigt, der erste Mi-Mikrocomputer, der mit entsprechender Peripherie (Tastatur, Bildschirm) ausgerüstet ist, als Terminal für den zweiten verwendet. Der zweite wird also über den ersten programmiert und gesteuert. Vom ersten werden nur die Programme zum Senden und Empfangen von Daten ausgenutzt, nicht aber eventuelle weitere Ein/Ausgabemöglichkeiten oder Rechenleistungen.

Von paralleler Rechenleistung kann daher hierbei nicht gesprochen werden. Diese wird nun aber durch die zweite, gestrichelt eingezeichnete Verbindung zwischen den Rechnern ermöglicht, die einen Ausgang des zweiten mit dem Interrupteingang des ersten verbindet. Wird diese im Laufe eines Programmes durch den zweiten Prozessor aktiviert, so verläß der erste das Terminalprogramm und springt in sein Interruptprogramm. Mit diesem empfängt er Daten des zweiten Prozessors, die anders als im Terminalbetrieb weiterverarbeitet werden. Das Interruptprogramm kann auch durch diese Daten umgeschaltet werden, aus ihm können auch wieder Daten oder Steuersignale an den zweiten Prozessor abgesandt werden. Auf diese Weise läßt sich eine echte Zusammenarbeit der beiden Rechner programmieren.

Dieses Verfahren habe ich an einem Mikrocomputer vom Typ AIM 65 zur Anwendung gebracht der über Tri-State-Puffer (parallelen) Zugang zu einem äußeren Adreß- und Datenbus hat. An diesen ist ein statischer RAM-Speicher angeschlossen, siehe Bild 2. Soweit stellt diese Konfiguration eine statische Speichererweiterung für den AIM 65 mit gepufferten Bussen dar. Da der äußere Bus aber abtrennbar ist, kann er auch von einem beliebigen anderen, asynchron zum AIM 65 arbeitenden Mikroprozessor verwendet werden, der in der oben beschriebenen Weise durch den AIM 65 gesteuert wird. Man erhält dadurch die Möglichkeit, andere Mikroprozessoren in Hard- und Software kennenzulernen, wobei viele Funktionen des AIM-Monitors eingesetzt werden können, da der statische Speicher des äußeren Prozessors dem AIM ja zugänglich gemacht werden kann. - Als äußeren Prozessor betreibe ich den SC/MP von National mit dem NIBL-BASIC-Interpreter, der für einfache Steuerfunktionen recht geeignet ist. Eine modernere Alternative hierzu wäre der Einchipcomputer 8073. Der Ausgang F2 des SC/MP steuert den Interrupteingang des AIM 65. Das Interruptprogramm verzweigt an eine vom SC/MP im statischen RAM abgelegte Adresse.

Während des Terminalbetriebes werden die Daten zwischen den beiden Rechnern seriell übertragen, um den Ein- und Ausgabeanforderungen des NIBL-Interpreters gerecht zu werden. Einem Ausgangstor des AIM 65 werden Steuersignale S1, S2 und S3 entnommen, von denen S1 dem AIM 65 Zugang zum äußeren Bus verschafft und dabei den SC/MP anhält. S2 ist an den Reseteingang des SC/MP geführt. S3 steuert eine Schreibschutzlogik für einen der beiden 4 K-Byte-Blöcke des statischen RAMs. Letzterer wird durch den SC/MP als Adreßbereich 2xxx angesprochen. Falls der SC/MP mit dem NIBL-Interpreter nach Aufhebung des Reset hier einen ROM-Bereich mit einem BASIC-Programm vorfindet, so wird dieses als erstes ausgeführt.

Der AIM 65 hat hierdurch 'periphere Intelligenz' erhalten, und Maschinenprogramme des AIM 65 können aus einem parallel abgearbeiteten BASIC-Programm heraus aufgerufen werden. Der zusätzliche Aufwand gegenüber einer statischen Speichererweiterung kann dabei als gering bezeichnet werden.

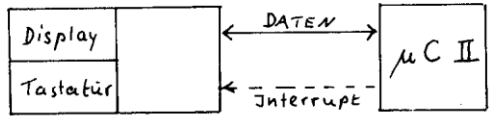


Bild 1

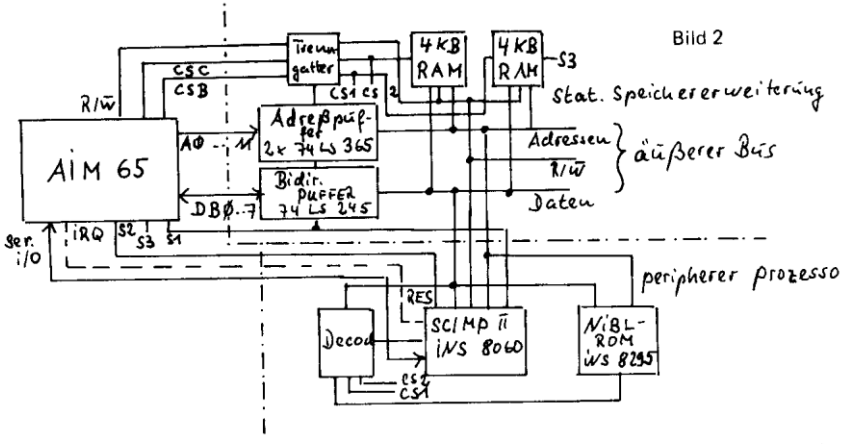


Bild 2

Dr. Andreas Joss, CH-3012 Bern

Die (Un-) Zuverlässigkeit von Kassettenspeichern

Einleitung

Wohl die meisten Computerbenützer haben schon erlebt, daß ihr Bandgerät ein File nicht mehr lesen wollte. Die Gründe, die dazu führen, sind vielfältig. Faktoren, die mitspielen sind (s.a. MICRO MAG Nr. 16, S. 36 ff):

- Aufnahmeelektronik
- Tonkopf (Stellung, Verschmutzung, Magnetisierung, Abnützung)
- Bandlauf (Tonkopfkontakt, Bandführung, Gleichlauf)
- Kassette, Bandmaterial
  - Aussteuerbarkeit allgemein
  - Höhenaussteuerbarkeit
  - Zustand (Aussetzer, mechanische Schäden)
  - Alter (Aging, Verlieren der Magnetisierung)
- Wiedergabeelektronik
- Aufnahmeformat (Baud-Rate, Codierungsart).

Der vorliegende Artikel möchte versuchen, etwas Klarheit zu schaffen, ist aber wegen des beschränkten Materials eher als Erfahrungsbericht denn als eine wissenschaftliche Studie aufzunehmen.

Material und Methode

Untersucht wurden die Faktoren Elektronik, Bandmaterial, Alterung. Zwei Bandgeräte wurden benützt mit 8000 Baud bei normaler Bandgeschwindigkeit. Das Bandformat sowie die Elektronik sind in Heft 16, S. 33 ff. im Detail beschrieben.

Hier nochmal die Besonderheiten der Bandgeräteelektronik:

	Aufnahmeelektronik	Wiedergabeelektronik
T 1	Gleichstrom durch Aufnahmekopf (Schaltbild s. Micromag Nr. 16, S. 36)	Phasengetreue, aufwendige Elektronik mit Schmitt-Trigger mit Hysterese (Schaltbild s. Micromag Nr. 16, S. 38)
T 2	normale HF-Vormagnetisierung	Schmitt-Trigger nach 1. Transistorstufe (Schaltbild s. Micromag Nr. 16, S. 37)

17 Bandkassetten wurden verwendet. Alle waren vorher nie bespielt und wurden in zwei Geschäften eingekauft. Drei Sorten wurden in beiden Geschäften gekauft. Preise und Marken sind aus Tabelle 1 ersichtlich.

Auf 16 Kassetten wurden je 10.0000 Bytes hex 16 (Sync-Bytes) mit T1 und 1.000 Bytes hex 26 mit T2 aufgenommen. Diese Aufnahmen wurden auf den entsprechenden Geräten nach verschiedenen Zeiten abgespielt und die Verluste berechnet.

Vorversuche hatten ergeben, daß Band 6 und 13 als einzige auf T2 (mit normaler Aufnahmeelektronik) brauchbare Resultate ergaben. Deshalb wurden auf den Bändern 6 und 13 je 10x weitere 10.000 Bytes mit verschiedenen Bitmustern analysiert, Aufnahme und Wiedergabe auf T2.

Auf den Bändern 6, 13 und 17 wurde schließlich im FAST-Format (s. Heft 16) mit dreimaliger Widerholung jedes Blocks der gesamte ROM-Bereich des AIM von Adresse B000 bis FFFF in Blocks je 256 Bytes gedumpt (240 Blocks, 3x 20K Byte). Mit T1 wurden Band 6 und 17 bespielt, mit T2 Band 6 und 13. Die Aufnahmen von T1 wurden auch auf T2 abgespielt (Tabelle 3).

Die Alterung wurde künstlich beschleunigt, mit erhöhten Temperaturen. Nach der Faustregel, daß chemische Prozesse bei jeweils 10^0 Celsius erhöhter Temperatur doppelt so schnell ablaufen, wurden die Bänder zuerst einige Tage bei 20^0 Celsius, dann 4 Tage bei 50^0 C (8-fache Alterung) und ca. 6 Wochen bei 60^0 C (16-fache Alterung) im Klimaschrank gelagert und jeweils nur einige Stunden zum Abspielen entfernt. - Der Übersichtlichkeit wegen wurden die Resultate stark reduziert zusammengefaßt.

Resultate

Tabelle 2: Aufnahme und Wiedergabe auf T2, verschiedene Bitmuster

	Band 6, je 10'000 Bytes			Band 13, je 10'000 Bytes		
Bitmuster	Anfang	5 Monate	2 Jahre	Anfang	5 Monate	2 Jahre
00010110	+++	++	--	+++	+++	++
00100110	+++	+	--	+++	+++	+++
00000010	+++	+	--	+++	+++	+++
11111110	+++	+++	+	+++	+++	+++
00101100	+++	+	--	+++	+++	+++
11101001	+++	+++	+	+++	+++	+++
00001111	+++	+++	-	+++	+++	+++
00011101	+++	-	--	+++	+++	+
01010100	+++	-	--	+++	+++	++
01010101	+	---	---	+++	+++	--

Tabelle 1: Gesamtliste und Resultate des 1. Versuchs

Band	Typ	Preis (Fr.)	10'000 Bytes ∅ 16 (Sync) Tape 1 → Tape 1 Anfang	nach 5 Monaten	nach 2 Jahren	1'000 Bytes ∅ 26 Tape 2 → Tape 2 Anfang	nach 5 Monaten	nach Jahr
1	Hitachi Low Noise C60	2.30	+++	+++	+++	---	---	--
2	" " " "	"	+++	+++	+++	---	---	--
3	" UDC 46	3.50	+++	+++	+++	---	---	--
4	" "	"	+++	+++	+++	---	---	--
5	" UDC 60	4.80	+++	+++	+++	---	---	--
6	" UDC 60 EX	5.50	+++	+++	+++	++	++	
7	" UDC 60 ER	"	+++	+	++	---	---	--
8	Maxell UDXLI C60	"	+++	++	+	---	---	--
9	Agfa Superferro 60+6	3.80	-	---	---	---	---	--
10	" Ferrocolor 60+6	2.50	+++	-	-	---	---	--
11	" " "	"	+++	+++	+	---	---	--
12	Scotch High Energy C45	3.30	+++	++	++	---	---	--
13	" Metafine C46	8.--	+++	++	+	++	++	
14	Memorex MRX30xide C60	5.--	+++	+	+	---	---	--
15	Philips Ferro C60	3.--	+++	-	--	---	---	--
16	BASF Ferrosuper LH C60	3.90	-	---	---	---	---	--
17	Maxell UDC 90	6.--	——— nicht getestet ———					

Wertung für Tabellen 1 und 2:

+++ = bis 0.05 % Verlust
++ = bis 0.1 % Verlust
+ = bis 0.5 % Verlust
0 = bis 1 % Verlust
- = bis 5 % Verlust
-- = bis 20 % Verlust
--- = über 20 % Verlust

Tabelle 3: Anzahl falsch gelesener Blocks
(total 240 Blocks je 256 Bytes Daten sowie Sync am Anfang und Checksummen)

	Aufnahme Wiedergabe	Anfang	1 Monat	5 Mte	1 Jahr	2 Jahre
Band 6	T 2 ——→ T 2	21	58	73	228	220
	T 1 ——→ T 1	0	0	0	0	0
	T 1 ——→ T 2	0	0	0	0	0
Band 13	T 2 ——→ T 2	0	0	0	6	10
Band 17	T 1 ——→ T 1	0	0	0	0	4
	T 1 ——→ T 2	0	0	0	0	19

Diskussion

Aus Tabelle 1 geht hervor, daß mit einigen Ausnahmen alle Bänder auf T1 gute Resultate erzielten. Auf T2 hingegen schnitten nur 2 Bänder gut ab: Band 6 war das einzige Normalband auf CrO2-Position (Hitachi UDC60EX), Band 13 war das einzige Metallband (Scotch Beide Sorten hatten eine gute Höhenaussteuerbarkeit gemeinsam. Die modifizierte Aufnahme-

elektronik schien eine weitaus bessere Magnetisierung der Bänder zu haben.

Aus Tabelle 2 geht hervor, daß, falls eine normale Aufnahmeelektronik wie bei T2 verwendet wurde, das Metallband überlegen war; ferner, daß verschiedene Bitmuster nicht gleiche Probleme stellten. Speziell eine Folge von 0101... stellte hohe Anforderungen an die Elektronik und wurde durch jedes RC-Glied verzerrt.

Tabelle 3 zeigt, daß für eine derart dichte Datenaufzeichnung (8000 Baud) mit der modifizierten Aufnahmeelektronik, kombiniert mit einem gut höhenaussteuerbaren Band (CrO2-Position) beste Dauerresultate erzielt werden konnten.

Aus den vorliegenden Daten geht nicht der Einfluß von Geschwindigkeitsschwankungen hervor. Dieser war jedoch nach Auffassung des Autors vernachlässigbar, da selbst eine Variation von 2% bei einem Verhältnis von beinahe 1:2 in der Modulation, wie dies von den meisten Bandformaten verwendet wird, eine geringe Rolle spielte.

Schlußfolgerungen

Beim Einkauf von Kassetten zur Datenaufzeichnung schien sich eine Bandsorte mit guter Höhenaussteuerung zu lohnen (in CrO2-Position). Ferner lohnte sich der in dieser Zeitschrift Nr. 16, S. 36 allgemein gültig beschriebene Umbau der Aufnahmeelektronik. Die auf der darauffolgenden Seite beschriebene Wiedergabeelektronik lohnte sich in 2. Priorität, wegen der Vermeidung von zu vielen RC-Gliedern.
Das Metallband war überlegen mit konventioneller Elektronik, fiel jedoch zurück bei der modifizierten Aufnahmeelektronik.

¤

Dirk Sanders, 6100 Darmstadt

Komplette BASIC-Statements durch CTRL und Tastendruck

Das Monitorprogramm des AIM erzeugt in Verbindung mit der CTRL-Taste eine alphabetisch geordnete Reihe von besonderen ASCII-Zeichen. CTRL/A entspricht z.B. hex 00, CTRL/Z dem Wert hex 1A. Eine Auflistung der CTRL-Codes ist in 65xx MICRO MAG, Heft 14, S. 16 zu finden.

Gedacht sind diese Codes vor allem als Steuerzeichen für Video-Interfaces, fast alle lassen sich aber auch in einem Programm zur Erzeugung von kompletten BASIC-Wörtern verwenden. Durchsucht man das BASIC Interpreter-ROM, so findet sich ab Adresse hex B090 die komplette Auflistung der BASIC-Befehle in Textform. Für sie kann also auf eine eigene Tabelle verzichtet werden. Im letzten Byte eines jeden Wortes ist Bit 7 gesetzt, um das Wortende zu kennzeichnen. Eine Auflistung dieses Bereiches ist weiter unten abgedruckt.

Es bietet sich an, ein Programm mit Tabelle zu erstellen, das zu jedem CTRL-Code das entsprechende Wort aufruft und im Eingabepuffer des BASIC-Interpreters (hex 14-50) ablegt. Ein gewisser Komfort ist erreicht, er läßt sich jedoch noch steigern: Für einige Statements wäre der Vorbefehl PRINT (?) sowie der Nachsatz 'Klammer auf, Null, Klammer zu' wünschenswert. Absoluten Komfort bietet ein nachgesetztes RETURN, z.B. PRINT FRE(0) RETURN. Schwer zu erreichen ist dies nicht:

Eine zweite Tabelle speichert die Arbeitsanweisungen. Bitweise werden hier für jeden Befehl die Zusatzfunktionen programmiert und später im Programm abgefragt. Eine Verdichtung von zwei Arbeitsanweisungen auf ein Byte schien in dieser Tabelle nicht lohnenswert, da die gesparten Bytes doch wieder von der Seitentrennung aufgebraucht würden. Auf diese Art bleibt auch Raum für eigene Erweiterungsideen.

Zum Programmablauf und zur Bedienung:

Das Programm ist im Speicherbereich so angeordnet, daß die ATN-Funktion, falls notwendig, noch im Bereich bis hex 0FFF (für 4 K RAM) abgespeichert werden kann (Achtung: Im BASIC Reference Manual steht ATN nicht im höchstmöglichen Bereich; nicht vergessen, die Adresse im Programm selbst zu ändern). Der BASIC-Speicherbereich ist auf 3830 zu begrenzen. Durch F3 wird der DILINK-Vektor auf CONTROL BASIC umgeschaltet. Dieser 'neue' Vektor kann auf Band gespeichert werden. Nun ist es sogar möglich, den Bereich von hex 0EF6 bis 0F00 bei Speichermangel dem BASIC zuzuordnen. Wird der Inhalt von 7F und 80 auch auf Band gespeichert, so kann BASIC bequem mit RETURN initialisiert werden. CONTROL BASIC ist dann allerdings erst nach erfolgter Initialisierung zu laden.

Bemerkungen zum Listing: In X befindet sich am Schluß der Wert für den Anzeigezeiger, dieser wird vom BASIC in Speicherstelle hex 11 kopiert. Ist der Akku bei Rückkehr ins BASIC =00, so erfolgt RETURN. Hex FF als Arbeitsanweisung reserviert den Code für andere Anwendungen. Zur Auswahl der Worte: 23 verschiedene CTRL-Tasten sind nutzbar. Die Auswahl erfolgte nach der Wortlänge (4 Tasten) und kann leicht durch die Änderung der Wortadresse auf eigene Bedürfnisse geändert werden.

Funktionszuweisung zu den Tasten

A	TAN(	G	GOTO	M	---	S	SIN(
B	RETURN	H	LIST ↑	N	NEXT	T	THEN
C	COS(	I	INPUT	O	?PEEK(	U	GOSUB
D	SAVE ↑	J	---	P	POKE	V	CONT
E	PEEK(	K	LIST	Q	SQR(	W	LEFT$(
F	?FRE(Ø) ↑	L	LOAD ↑	R	RUN ↑	X	MID$(
Y	RIGHT$(						

↑ BEDEUTET RETURN

```
ØEF6 A2 LDX #Ø1       NEUE ADRESSE NACH DILINK BRINGEN
ØEF8 8E STX A4Ø6
ØEFB A2 LDX #ØF
ØEFD 8E STX A4Ø7
ØF00 60 RTS
ØF01 C9 CMP #1A       BUCHSTABE? ASCII ≥ $1A
ØF03 BØ BCS ØF09
ØF05 C9 CMP #ØD       RETURN?
ØF07 DØ BNE ØF0C
ØF09 4C JMP ØF05      JA, IN ANZEIGE
ØF0C A8 TAY
ØF0D BE LDX ØF79,Y    ARBEITSANWEISUNG
ØF10 EØ CPX #FF       WENN $FF, NICHT BEARBEITEN, Z.B. TV-STEUERZEICHEN
ØF12 FØ BEQ ØF09
ØF14 8A TXA           ANZEIGE, CTRL IN A
ØF15 AE LDX A415      ANZEIGEZEIGER
ØF18 CE DEC A416      CTRL-ZEICHEN IN DRUCKPUFFER LOSCHEN
ØF18 CE DEC A416
ØF1B 48 PHA
ØF1C C9 CMP #FF         UM N-FLAG NACH A ZU STELLEN
ØF1E 1Ø BPL ØF25
ØF2Ø A9 LDA #3F         "?" FÜR PRINT
ØF22 2Ø JSR ØF57        - AUSGABE -
ØF25 B9 LDA ØF6Ø,Y      WORTADRESSE IM BASIC ROM
```

```
ØF28 A8 TAY
ØF29 B9 LDA BØ9Ø,Y      LADE BASIC-WORT
ØF2C 48 PHA
ØF2D 29 AND #7F         BIT 7 LÖSCHEN
ØF2F 2Ø JSR ØF57        - AUSGABE -
ØF32 C8 INY
ØF33 68 PLA
ØF34 1Ø BPL ØF29        BIT 7 = WORTENDE
ØF36 68 PLA
ØF37 ØA ASL .A          NÄCHSTE ANWEISUNG
ØF38 A8 TAY
ØF39 1Ø BPL ØF4Ø
ØF3B A9 LDA #28         "("
ØF3D 2Ø JSR ØF57        - AUSGABE -
ØF4Ø 98 TYA
ØF41 ØA ASL .A          NÄCHSTE ANWEISUNG
ØF42 A8 TAY
ØF43 1Ø BPL ØF4F
ØF45 A9 LDA #3Ø       ASCII "Ø"
ØF47 2Ø JSR ØF57      - AUSGABE -
ØF4A A9 LDA #29       ")"
ØF4C 2Ø JSR ØF57      - AUSGABE -
ØF4F 98 TYA
ØF5Ø ØA ASL .A        NÄCHSTE ÄNWEISUNG
ØF51 3Ø BMI ØF55
ØF53 A9 LDA #FF       FF, UM RETURN ZU VERHINDERN
ØF55 ØA ASL .A        A WIRD GGFS. ØØ, RETURN
ØF56 6Ø RTS           ZURÜCK INS BASIC (ANZEIGEZEIGER IN X)
ØF57 95 STA 16,X      IN BASIC EINGABEPUFFER
ØF59 2Ø JSR FØØØ      DRUCKPUFFER
ØF5C 2Ø JSR EF@5      ANZEIGE
ØF5F E8 INX           X AUF NÄCHSTEN BUCHSTABENPLATZ
ØF6Ø 6Ø RTS

Ø112 4C JMP ØEF6
```

```
<M>=ØF7A 4Ø ØØ 4Ø 1Ø    ARBEITSANWEISUNGEN:
< > ØF7E 43 FØ ØØ 1Ø
< > ØF82 ØØ FF ØØ 1Ø    BIT 7=1 PRINT ("?")
< > ØF86 FF ØØ CØ ØØ        6=1 KLAMMER AUF
< > ØF8A 4Ø 1Ø 4Ø ØØ        5=1 NULL UND KLAMMER ZU
< > ØF8E ØØ 1Ø 4Ø 4Ø        4=1 RETURN
< > ØF92 4Ø
```

```
BESONDERHEITEN: CTRL J WIRD VOM INTERPRETER AUSGEFILTERT
                CTRL M IST RETURN
                CTRL O WIRKT NUR JEDES ZWEITE MAL
                CTRL Z WIRD VOM INTERPRETER AUSGEFILTERT,
                       DAHER NICHT MEHR IN TABELLE
```

```
<M>=ØF61 B8 32 B2 4D      ADRESSEN DER BASIC STATEMENTS IM ROM
< > ØF65 BE AØ 1D 61      BEZUGSADRESSE IST BØ9Ø
< > ØF69 ØE ØØ 61 49
< > ØF6D ØØ Ø6 BE 54
< > ØF71 A6 21 B5 7C
< > ØF75 2D 5D D3 DE
< > ØF79 D8
```

```
90 ENDFORNEXTDATAIN      AUFBAU DER WORTE IM BASIC-ROM
A0 PUTDIMREADLETGOT
B0 ORUNIFRESTOREGOS
C0 UBRETURNREMSTOPO
D0 NNULLWAITLOADSAV
E0 EDEFPOKEPRINTCON
F0 TLISTCLEARGETNEW
00 TAB(TOFNSPC(THEN
10 NOTSTEP+-*/↑ANDO
20 R>=<SGNINTABSUSR
0 FREPOSSQRRNDLOGE
40 XPCOSSINTANATNPE
50 EKLENSTR$VALASCC
60 HR$LEFT$RIGHT$MI
70 D$GONFSNRGODFCO
80 VOMUSBSDD/0IDTML
90 SSTCNUF ERROR I
A0 N
```

Dr. W.-D. Schnell, 5042 Erftstadt

LMR - LOAD, MOVE, RELOCATE

Unter den im MICRO MAG veröffentlichten Programmen fehlte bisher ein Relozier-Programm mit voller Tabellenverarbeitung für den AIM 65. Diese Lücke soll das Dienstprogramm LMR (LOAD, MOVE, RELOCATE) schließen. Angeregt wurde der Verfasser durch das Programm RALOAD von R. Löhr (MICRO MAG 1, Seite 3), das für den AIM umgeschrieben und um die Berechnung virtueller Adressen erweitert wurde. Die Lade-Routine ist eine Abwandlung der Monitor-Routine unter L-Kommando ab E2E6, ähnlich wie es M. Krägeloh (MICRO MAG 12, Seite 43) vorgeschlagen hat. Die MOVE-Routine ist von J. Butterfield (FIRST Book of KIM). Was die Möglichkeiten (z.B. Längenoperatoren, Basistabellenadresse) und Restriktionen betrifft, sei auf die oben aufgeführte Literatur verwiesen. Um eine leichtere Vergleichbarkeit der Programmteile des Programms LMR mit den Original-Programmen zu erzielen, wurden die Original-Labels weitestgehend im LMR-Quelltext beibehalten.

Mit dem Dienstprogramm LMR kann mit dem AIM 65 T-Format einvon einer Cassette kommendes Programm verschoben in den RAM-Bereich geladen werden oder ein im Speicher vorhandenes Programm verschoben werden, mit Relozieren der programminternen Adressen. Außerdem können die programminternen, absoluten Adressen für einen real nicht vorhandenen RAM-Bereich (virtuell) umgerechnet werden. Damit ist es möglich, Programme im realen, zur Verfügung stehenden RAM-Bereich mit denjenigen Adressen zu erstellen, wie sie später für ein nicht im RAM-Bereich adressiertes EPROM vorliegen müssen.

Die gewünschte Dienstleistung wird nach dem Start mit *=200 und G/ SPACE durch die Eingabe von 2 der Zeichen L, M, R, Space (interaktive Abfrage) bewirkt. Nur die in der folgenden Tabelle aufgeführten Zeichenkombinationen sind zulässig und werden vom Programm akzeptiert. Die Tabelle zeigt die sinnvollen Dienstleistungen auf. Im allgemeinen läuft die interaktive Abfrage so, daß sie gut verstanden und richtig beantwortet wird. Ausnahme: Sollen die virtuellen Adressen eines Programmes auf den realen Programmbereich, in dem das Programm steht, zurückgerechnet werden, so ist der Programmteil R zu nehmen und es ist nacheinander einzugeben:

1. nach FROM: Virtuelle Anfangsadresse
2. nach TO: Reale Endadresse
3. nach TO: Reale Anfangsadresse
4. nach V TO: Reale Anfangsadresse

Nach dem Programmlauf sollte darauf geachtet werden, daß das Monitor-Promptzeichen im Display angezeigt wird.

Aus der Beschreibung des LMR-Programmes ist ersichtlich, daß dieses sehr vielfältig eingesetzt werden kann. Insbesondere sei darauf hingewiesen, daß Programmpakete für den direkten Programmlauf oder für die EPROM-Programmierung leicht zusammengestellt werden können. Auch können Monitor-Routinen zur Abwandlung in den RAM-Bereich geladen werden und dort laufen. Mit MOVE können natürlich Tabellen beliebig verschoben werden. Falls die Programme keine Tabellen enthalten und nicht entsprechend RALOAD vorbereitet sind, so muß ein EA (ein NOP) an das Programmende angefügt werden. Falls das Programm Tabellen enthält, so können LOP's (Längenoperatoren) von Hand vorübergehend für interne Tabellen eingefügt und später wieder entfernt werden. Gleiches gilt für die Änderung einer evtl. benötigten Basis-Tabellenadresse.

Besonders hingewiesen sei darauf, daß mit dem Programmteil R keine Einfügungen in Programme oder Befehlslöschungen vorgenommen werden können. Hierzu RELOCATE von J. Butterfield bzw. MOVE AND RELOCATE von I. Dohmann (MICRO MAG 7, Seite 24; Vorsicht bei BRANCH-Befehlen, Tabellen) oder das Programm DIASO von K.-R. Hase einsetzen.

Steht nur ein kleines RAM zur Verfügung, so kann LMR bei der Adresse 039A auch geteilt werden. Es ist dann nur die MSG1 an den ersten Teil anzuhängen und die LOP's sind entsprechend zu setzen.

Tabelle der Programm-Kombinationen

Programm LMR bewirkt die Ladung/Verschiebung/Adreßumrechnung in:

	Realen RAM-Bereich	Realen, verschobenen RAM-Bereich	Virtuelle Adressen im realen RAM-Bereich
L␣	+		
LR	(+)	+	+
M␣		+	
MR	(+)	+	+
R␣	+ ←	→	+

```
0000 BLANK=$E83E
0000 FROM=$E7A3
0000 REDOUT=$E973

0000 OUTPUT=$E97A

0000 OUTDP=$EEFC
0000 CRCK=$EA24

0000 CKERR=$E385
0000 TO=$E7A7
0000 WHEREI=$E848

0000 INALL=$E993
0000 CLRCK=$EB4D
0000 CHEKAR=$E54B
```

```
0000 RBYTE=$E3FD
0000 STBYTE=$E413
0000 CKSUM=$A41E
0000 LOAD4=$E321
0000 MON=$E182
0000 ADDR=$A41C
0000 TABUFF=$0116
0000                *=$00
0000 OAL=*+$0
0000 OAH=*+$1
0000 ANFADL=*+$2
0000 ANFADH=*+$3
0000 ENDADL=*+$4
0000 ENDADH=*+$5
0000 VANFL=*+$6
0000 VANFH=*+$7
```

```
0000 VENDL=*+$8
0000 VENDH=*+$9
0000 LAUADL=*+$A
0000 LAUADH=*+$B
0000 INCL=*+$C
0000 INCH=*+$D
0000 VINCL=*+$E
0000 VINCH=*+$F
0000 LAENGE=*+$10
0000 FLAG=*+$11
0000 CARRY1=*+$12
0000 CARRY2=*+$13
0000 END=*+$14
0000 OSAL=*+$16
0000 OSAH=*+$17
0000 OEAL=*+$18
0000 OEAH=*+$19
0000 NSAL=*+$1A
0000 NSAH=*+$1B
0000 NEAL=*+$1C
0000 NEAH=*+$1D
0000 BCL=*+$1E
0000 BCH=*+$1F
0000                    *=$200
0200                    ;-------------------
0200                    ;INPUT L/M/R/" " ?
0200
0200 START   2024EA  JSR CRCK
0203         A900    LDA #$00
0205         8511    STA FLAG
0207         A002    LDY #$02
0209         A2FF    LDX #$FF
020B         D8      CLD
020C LMR1    E8      INX
020D         BDE604  LDA MSG1,X
0210         C93B    CMP #';'
0212         F005    BEQ INPUT
0214         207AE9  JSR OUTPUT
0217         D0F3    BNE LMR1
0219 INPUT   2073E9  JSR REDOUT
021C         A280    LDX #$80
021E         C94C    CMP #'L
0220         F012    BEQ LMR2
0222         A240    LDX #$40
0224         C94D    CMP #'M
0226         F00C    BEQ LMR2
0228         A220    LDX #$20
022A         C952    CMP #'R
022C         F006    BEQ LMR2
022E         A200    LDX #$00
0230         C920    CMP #' '
0232         D0CC    BNE START
0234 LMR2    8A      TXA
0235         C511    CMP FLAG
0237         F0C7    BEQ START
0239         18      CLC
023A         6511    ADC FLAG
023C         8511    STA FLAG
023E         88      DEY
023F         D0D8    BNE INPUT
0241         2024EA  JSR CRCK
0244         A511    LDA FLAG
0246         C9C0    CMP #$C0
0248         F0B6    BEQ START
024A         2411    BIT FLAG
024C         303B    BMI LMR4
024E
024E                 ;-------------------
024E                 ;ABFRAGE FUER MOVE
024E
024E MOVE    20A3E7  JSR FROM
0251         B0FB    BCS MOVE
0253         203EE8  JSR BLANK
0256         AD1CA4  LDA ADDR
0259         8500    STA OAL
025B         8516    STA OSAL
025D         AD1DA4  LDA ADDR+1
0260         8501    STA OAH
0262         8517    STA OSAH
0264 LMR3    20A7E7  JSR TO
0267         B0FB    BCS LMR3
0269         2024EA  JSR CRCK
026C         38      SEC
026D         AD1CA4  LDA ADDR
0270         8518    STA OEAL
0272         8504    STA ENDADL
0274         E516    SBC OSAL
0276         851E    STA BCL
0278         AD1DA4  LDA ADDR+1
027B         8519    STA OEAH
027D         8505    STA ENDADH
027F         E517    SBC OSAH
0281         851F    STA BCH
0283         E604    INC ENDADL
0285         D002    BNE LMR4
0287         E605    INC ENDADH
0289
0289                 ;-------------------
0289                 ;ABFRG FUER MOVE+LOAD
0289
0289 LMR4    20A7E7  JSR TO
028C         B0FB    BCS LMR4
028E         2024EA  JSR CRCK
0291         AD1CA4  LDA ADDR
0294         8502    STA ANFADL
0296         851A    STA NSAL
0298         AD1DA4  LDA ADDR+1
029B         8503    STA ANFADH
029D         851B    STA NSAH
029F         A511    LDA FLAG
02A1         C940    CMP #$40
02A3         D003    BNE LMR5
02A5         4C3603  JMP MOVE1
02A8 LMR5    A956    LDA #'V
02AA         207AE9  JSR OUTPUT
```

```
02AD          A920    LDA #' '
02AF          207AE9  JSR OUTPUT
02B2          20A7E7  JSR TO
02B5          B0F1    BCS LMR5
02B7          2024EA  JSR CRCK
02BA          AD1CA4  LDA ADDR
02BD          8506    STA VANFL
02BF          AD1DA4  LDA ADDR+1
02C2          8507    STA VANFH
02C4          A502    LDA ANFADL
02C6          8D1CA4  STA ADDR
02C9          A503    LDA ANFADH
02CB          8D1DA4  STA ADDR+1
02CE          2411    BIT FLAG
02D0          1064    BPL MOVE1
02D2
02D2          ------------------------
02D2          LOAD-PROGRAMM
02D2
02D2 NLOAD    2048E8  JSR WHEREI
02D5 NLOAD1   2093E9  JSR INALL
02D8          C93B    CMP #';
02DA          D0F9    BNE NLOAD1
02DC          204DEB  JSR CLRCK
02DF          204BE5  JSR CHEKAR
02E2          AA      TAX
02E3          204BE5  JSR CHEKAR
02E6          204BE5  JSR CHEKAR
02E9          A900    LDA #0
02EB          CD1601  CMP TABUFF
02EE          D00A    BNE BLKN
02F0          AD1F01  LDA TABUFF+9
02F3          8501    STA OAH
02F5          AD2001  LDA TABUFF+10
02F8          8500    STA OAL
02FA BLKN     8A      TXA
02FB          F01F    BEQ NLOAD4
02FD NLOAD2   20FDE3  JSR RBYTE
0300          2013E4  JSR STBYTE
0303          CA      DEX
0304          D0F7    BNE NLOAD2
0306          20FDE3  JSR RBYTE
0309          CD1EA4  CMP CKSUM+1
030C          D008    BNE NCKERR
030E          20FDE3  JSR RBYTE
0311          CD1EA4  CMP CKSUM
0314          F0BF    BEQ NLOAD1
0316
0316
0316 NCKERR   2085E3  JSR CKERR
0319          4C8902  JMP LMR4
031C NLOAD4   2021E3  JSR LOAD4
031F          AD1CA4  LDA ADDR
0322          8504    STA ENDADL
0324          8514    STA END
0326          AD1DA4  LDA ADDR+1
0329          8505    STA ENDADH
032B          8515    STA END+1
032D          A511    LDA FLAG
032F          C980    CMP #$80
0331          D067    BNE RELOC
0333          4C82E1  JMP MON
0336
0336          ------------------------
0336          MOVE-PROGRAMM
0336
0336 MOVE1    A511    LDA FLAG
0338          C920    CMP #$20
033A          F05E    BEQ RELOC
033C          18      CLC
033D          A0FF    LDY #$FF
033F          A51E    LDA BCL
0341          651A    ADC NSAL
0343          851C    STA NEAL
0345          A51F    LDA BCH
0347          651B    ADC NSAH
0349          851D    STA NEAH
034B          E61E    INC BCL
034D          E61F    INC BCH
034F          38      SEC
0350          A51A    LDA NSAL
0352          E516    SBC OSAL
0354          A51B    LDA NSAH
0356          E517    SBC OSAH
0358 LOOP     A200    LDX #$00
035A          9002    BCC MOV
035C          A202    LDX #$02
035E MOV      A116    LDA (OSAL,X)
0360          811A    STA (NSAL,X)
0362          9014    BCC DOWN
0364          C618    DEC OEAL
0366          98      TYA
0367          4518    EOR OEAL
0369          D002    BNE NOT
036B          C619    DEC OEAH
036D NOT      C61C    DEC NEAL
036F          98      TYA
0370          451C    EOR NEAL
0372          D002    BNE NEIN
0374          C61D    DEC NEAH
0376 NEIN     B00C    BCS COUNT
0378 DOWN     E616    INC OSAL
037A          D002    BNE NYET
037C          E617    INC OSAH
037E NYET     E61A    INC NSAL
0380          D002    BNE COUNT
0382          E61B    INC NSAH
0384 COUNT    C61E    DEC BCL
0386          D002    BNE ONE
0388          C61F    DEC BCH
038A ONE      D0CC    BNE LOOP
038C          A90D    LDA #$0D
038E          20FCEE  JSR OUTDP
0391          A511    LDA FLAG
```

```
0393          2920    AND #$20
0395          D003    BNE RELOC
0397          4C82E1  JMP MON
039A
039A          ----------------------
039A          RELOCATE-PROGRAMM
039A
039A RELOC    38      SEC
039B          A502    LDA ANFADL
039D          E500    SBC OAL
039F          850C    STA INCL
03A1          A503    LDA ANFADH
03A3          E501    SBC OAH
03A5          850D    STA INCH
03A7          38      SEC
03A8          A506    LDA VANFL
03AA          E502    SBC ANFADL
03AC          850E    STA VINCL
03AE          A507    LDA VANFH
03B0          E503    SBC ANFADH
03B2          850F    STA VINCH
03B4          18      CLC
03B5          A504    LDA ENDADL
03B7          650E    ADC VINCL
03B9          8508    STA VENDL
03BB          A505    LDA ENDADH
03BD          650F    ADC VINCH
03BF          8509    STA VENDH
03C1          2411    BIT FLAG
03C3          5011    BVC BGN
03C5          18      CLC
03C6          A504    LDA ENDADL
03C8          650C    ADC INCL
03CA          8504    STA ENDADL
03CC          8514    STA END
03CE          A505    LDA ENDADH
03D0          650D    ADC INCH
03D2          8505    STA ENDADH
03D4          8515    STA END+1
03D6
03D6
03D6 BGN      206E04  JSR BEGINN
03D9          A901    LDA #$1
03DB          8510    STA LAENGE
03DD          208D04  JSR RUECK
03E0          A000    LDY #$0
03E2          B104    LDA (ENDADL),Y
03E4          C9EA    CMP #$EA
03E6          F032    BEQ GETOP
03E8
03E8
03E8 TABADJ   48      PHA
03E9          208D04  JSR RUECK
03EC          A900    LDA #$00
03EE          8512    STA CARRY1
03F0          8513    STA CARRY2
03F2          18      CLC
03F3          A50C    LDA INCL
03F5          7104    ADC (ENDADL),Y
03F7          9003    BCC TAB1
03F9          E612    INC CARRY1
03FB          18      CLC
03FC TAB1     AA      TAX
03FD          650E    ADC VINCL
03FF          9104    STA (ENDADL),Y
0401          9003    BCC TAB2
0403          E613    INC CARRY2
0405          18      CLC
0406 TAB2     C8      INY
0407          68      PLA
0408          6512    ADC CARRY1
040A          18      CLC
040B          650D    ADC INCH
040D          8505    STA ENDADH
040F          18      CLC
0410          6513    ADC CARRY2
0412          18      CLC
0413          650F    ADC VINCH
0415          9104    STA (ENDADL),Y
0417          8A      TXA
0418          8504    STA ENDADL
041A
041A
041A GETOP    C610    DEC LAENGE
041C SCHLEI   207704  JSR VORAN
041F          F00D    BEQ OUT1
0421          209904  JSR LBEST
0424          C003    CPY #$03
0426          90F4    BCC SCHLEI
0428          203104  JSR ADJUST
042B          4C1C04  JMP SCHLEI
042E OUT1     4C82E1  JMP MON
0431
0431
0431 ADJUST   A000    LDY #$0
0433          B10A    LDA (LAUADL),Y
0435          2903    AND #$03
0437          4903    EOR #$03
0439          F032    BEQ OUT
043B          C8      INY
043C          18      CLC
043D          B10A    LDA (LAUADL),Y
043F          650C    ADC INCL
0441          AA      TAX
0442          C8      INY
0443          B10A    LDA (LAUADL),Y
0445          650D    ADC INCH
0447          C503    CMP ANFADH
0449          9022    BCC OUT
044B          D004    BNE HIGB
044D          E402    CPX ANFADL
044F          901C    BCC OUT
0451 HIGB     C515    CMP END+1
0453          9006    BCC GOGO
```

```
0455          D016    BNE OUT
0457          E414    CPX END
0459          B012    BCS OUT
045B GOGO     650F    ADC VINCH
045D          910A    STA (LAUADL),Y
045F          88      DEY
0460          8A      TXA
0461          18      CLC
0462          650E    ADC VINCL
0464          910A    STA (LAUADL),Y
0466          C8      INY
0467          B10A    LDA (LAUADL),Y
0469          6900    ADC #$00
046B          910A    STA (LAUADL),Y
046D OUT      60      RTS
046E BEGINN   A502    LDA ANFADL
0470          850A    STA LAUADL
0472          A503    LDA ANFADH
0474          850B    STA LAUADH
0476          60      RTS

0477 VORAN    18      CLC
0478          A50A    LDA LAUADL
047A          6510    ADC LAENGE
047C          850A    STA LAUADL
047E          A50B    LDA LAUADH
0480          6900    ADC #$00
0482          850B    STA LAUADH
0484          C505    CMP ENDADH
0486          3004    BMI OUT4
  A50A    LDA LAUADL
048A          C504    CMP ENDADL
048C OUT4     60      RTS
048D RUECK    38      SEC
048E          A504    LDA ENDADL
0490          E510    SBC LAENGE
0492          8504    STA ENDADL

0494          B002    BCS OUT5
0496          C605    DEC ENDADH
0498 OUT5     60      RTS
0499 LBEST    A000    LDY #$00
049B          B10A    LDA (LAUADL),Y
049D          48      PHA
049E          C8      INY
049F          C900    CMP #$00
04A1          F024    BEQ BEST
04A3          C940    CMP #$40
04A5          F020    BEQ BEST
 C960    CMP #$60
04A9          F01C    BEQ BEST
04AB          A003    LDY #$03
04AD          C9FF    CMP #$FF
04AF          F016    BEQ BEST
04B1          C920    CMP #$20
04B3          F012    BEQ BEST
04B5          291F    AND #$1F
04B7          C919    CMP #$19
04B9          F00C    BEQ BEST
04BB          290F    AND #$0F
04BD          AA      TAX
04BE          BCD604  LDY LAENTB,X
04C1          2903    AND #$03
04C3          4903    EOR #$03
04C5          F004    BEQ LENCOD
04C7 BEST     68      PLA
04C8 OUT2     8410    STY LAENGE
04CA          60      RTS
04CB LENCOD   68      PLA
04CC          C9FF    CMP #$FF
04CE          F0F8    BEQ OUT2
04D0          4A      LSR A
04D1          4A      LSR A
04D2          A8      TAY
04D3          D0F3    BNE OUT2

04D5          8F      .BYT $8F
04D6 LAENTB   02      .BYT $02,$02,$02,$01,$02,$02,$02,$01,$01,$02,$01
04D7          02
04D8          02
04D9          01
04DA          02
04DB          02
04DC          02
01 04DE       01
04DF          02
04E0          01
04E1          01      .BYT $01,$03,$03,$03,$03
04E2          03
04E3          03
04E4          03
04E5          03
04E6 MSG1     3243    .BYTE '2CH: L/M/R/" " =;'
04F7          EA      .BYT $EA
04F8                  .END
```

Hinweise des Herausgebers zum Programm LMR: Außer in Heft 1, Seite 3 dieser Zeitschrift wurde das Programm RALOAD auch im 'Buch 1-6 des 65xx MICRO MAG', Seite 133 abgedruckt, und zwar in einer für den KIM-1 formulierten Fassung. Seine grundsätzlichen Dienstleistungen bestehen darin, daß ein auf Cassette abgespeichertes Programm an eine beliebige neue Stelle in den Speicher geladen werden kann und dort an Ort und Stelle in eine lauffähige Version umgerechnet wird. Dabei darf das umzurechnende Programm Tabellen enthalten. Die Umrechnung ist automatisiert, der Programmierer braucht keine Steueranweisungen an das Umrechnungsprogramm zu geben. - Das setzt allerdings voraus, daß das zu ladende Programm selbst einige Informationen für das Umrechnungsprogramm bereitstellt. Die 'Syntax' für diese Parameter ist denkbar einfach: Auf Cassette etc. aufgezeichnete Programme enthalten immer Information über den früheren Startpunkt. Das normale Ladeprogramm der Betriebssysteme enthält ferner nach dem Laden noch immer die Endadresse+1. Der Bediener gibt ferner die neue Startadresse vor. Die Umrechnung findet zwischen neuem Anfang und Ende statt, und zwar unter Berücksichtigung der Herkunftsinformation, um zu erkennen, welches die sich auf das Programm selbst beziehenden absoluten Adressen sind, die es umzurechnen gilt, unter Auslassung also externer absoluter Referenzen.

Nun zur Syntax, zu den Parametern zur Tabellenkennzeichnung: Wenn das letzte Byte des geladenen Programmes ein hex EA, ein NOP-Befehl ist, so hat der Programmierer gekennzeichnet, daß sich am Programmschluß keine Tabelle befindet, für die eine Tabellen-Basisadresse vorgehalten wird. Gleichwohl dürfen sich bei am Schluß mit NOP gekennzeichneten Programmen im Inneren noch Tabellen befinden (s.u.). Ist das letzte Byte ungleich EA, so kennzeichnen die beiden letzten Bytes zusammen eine Adresse, bei der im Ursprungsprogramm die angehängte Tabelle beginnt. Das Umrechnungsprogramm wird seine Programmumrechnung vor dieser Tabellen-Basisadresse abschließen und die beiden letzten Bytes auf den neuen Standort der Tabelle berichtigen. - Zur Kennzeichnung von Tabellen im Inneren eines Programmes: Das Umrechnungsprogramm prüft Programme auf die 'erlaubten' 6502-Opcodes. Wird ein auf hex x3, x7, xB, oder xF endender ('verbotener') Befehlscode angetroffen, so wird damit die Länge einer nachfolgenden Tabelle signalisiert. Dabei bedeuten z.B. LOPs von 03, 07,0B ... FB, daß ein LOP + 1, 2, 3... 61 Tabellenbytes folgen. Ein LOP mit FF wird mit nachfolgenden 2 Tabellenbytes bewertet.
Damit sind folgende Programmstrukturen wahlweise zugelassen:

Instruktionsfolge, LOP, Tabelle, NOP-Befehl,
Instruktionsfolge, LOP, Tabelle 1, LOP, Tabelle 2, ..., NOP
Instruktionsfolge, LOP, Tabelle 1, Instruktionsfolge, LOP, Tabelle 2, ..., NOP
Instruktionsfolge, Tabelle, Tabellenbasisadresse
Instruktionsfolge, LOP, Tabelle 1, ..., Instruktionsfolge, Tabelle X, Basisadresse der Tabelle X usw. ¤

Dipl.-Math. R. Becker-Gottschalk und Dipl.-Phys. K. H. Becker,
FB Biologie der Universität Konstanz

CBM 3032:
Benutzung arithmetischer Interpreterroutinen

Bei der Übertragung eines vorhandenen BASIC-Algorithmus für die 'Fast Fourier Transform' (FFT) in die Maschinensprache wurden die vorhandenen Routinen des Interpreters für die vier Grundrechenarten, sowie für Wurzelbildung, Sinus und Kosinus sowie die Umwandlung 'Real in Integer' und umgekehrt benutzt. Die nachfolgende Tabelle und die Beispiele zeigen in allgemeiner Form, wie und mit welchen Eingangsparametern die diesbezüglichen Dienstleistungen für maschinensprachliche Programmierung anzusprechen sind und wie die Ergebnisse abgeliefert werden.

Definitionen:

```
A:     AKKUMULATOR
X:     X-REGISTER
Y:     Y-REGISTER
GA1:   GLEITKOMMAAKKU 1        ($005E-$0063)
B, C:  GLEITKOMMAVARIABLEN     BHI = HEX. SEITENADRESSE
                               BLO = HEX. EXPONENTADRESSE
BI:    INTEGER VON B           BIHI = HOEHERWERTIGES BYTE
                               BILO = NIEDERWERTIGES BYTE
```

FUNKTION	VERSORGUNG	ERGEBNIS IN	INTERPRETER-ROUTINE 1)
GA1=B	A=BLO, Y=BHI	GA1	$DAAE
	$001F=BLO		
	$0020=BHI	GA1	$DAB2
B=GA1	X=BLO, Y=BHI	B	$DAE0
	$001F=BLO		
	$0020=BHI	B	$DAE7 A)
RUNDUNG(GA1)	GA1	GA1	$DB27
B+C	A=BLO, Y=BHI		
	GA1=C	GA1	$D773
C=-C	GA1=C	GA1	$DEA1
B-C	A=BLO, Y=BHI		
	GA1=-C	GA1	$D773
B*C	A=BLO, Y=BHI		
	GA1=C	GA1	$D934
B/C	A=BLO, Y=BHI		
	GA1=C	GA1	$DA1B
SQR(B)	GA1=B	GA1	$DE5E
COS(B) (RAD)	GA1=B	GA1	$DFD8
SIN(B) (RAD)	GA1=B	GA1	$DFDF
BI=B	GA1=B	$0061=BIHI	
		$0062=BILO	$DBA7 B)
B=BI	A=BIHI, Y=BILO	GA1	$D26D
B.V. ADRESS-SUCHROUTINE	$0042=1.ASCII	$005D=HI C)	
	$0043=2.ASCII	$005C=LO	$CFCD 2)

A) WENN $DAE7 BENUTZT WIRD, VORHER RUNDUNG(GA1) DURCHLAUFEN
B) OHNE VORZEICHEN
C) ADRESSE VOM 1. ASCII

BEISPIELE:

1. LADEN: B NACH GA1

```
LDA BLO
LDY BHI
JSR $DAAE !GA1=B
```

2. SPEICHERN: GA1 NACH C

```
LDX CLO
LDY CHI
JSR $DAE0 !C=GA1
```

ADRESSE VON C STEHT SCHON IN $001F & $0020

```
JSR $DB27 !RUNDUNG GA1
JSR $DAE7 !C=GA1
```

3. SUBTRAKTION B=B-C

```
LDA CLO
LDY CHI
JSR $DAAE !GA1=C
```

ENTFAELLT BEI ADDITION

```
JSR $DEA1 !GA1=-C
LDA BLO
LDY BHI
JSR $D773 !GA1=B-C
LDX BLO
LDY BHI
JSR $DAE0 !B=B-C
```

```
4. B=COS(B)                LDA BLO
                           LDY BHI
                           JSR $DAAE  !GA1=B
                           JSR $DFD8  !GA1=COS(B)
                           LDX BLO
                           LDY BHI
                           JSR $DAE0  !B=COS(B)

5. INTEGER VON B           LDA BLO
                           LDY BHI
                           JSR $DAAE  !GA1=B
                           JSR $DBA7  !GA1=BI
                           LDA $61    !HOEHERWERTIGES BYTE
                           STA BIHI   !NACH BIHI
                           LDA $62    !NIEDERWERTIGES BYTE
                           STA BILO   !NACH BILO
```

1) MIT AUSNAHME VON $D77B ENTNOMMEN AUS:
H.-P. GOERTZ / DUESSELDF. + W. KOELLN / DELMENHORST
65XX MICRO MAG 11: 28 (1980)
2) A. QUINDT / WIESBADEN, 65XX MICRO MAG 14: 35 (1980)

In ihrem Begleitbrief zu obigem Beitrag schreiben die Autoren: Der Algorithmus der 'Fast'-Fourier-Transformation verzichtet auf etliche mehrfach auszuführende (numerische) Summationen der normalen Fourier-Transformation, die sich letztlich im Ergebnis herausheben. Sie ist daher prinzipiell schneller. Er kann bei Programmierung in Maschinensprache mit einer noch akzeptablen Rechenzeit auch auf Kleinrechnern, wie z.B. dem CBM laufen [1]. Unser Programm ist knapp 3 k Bytes lang und benötigt für die Transformation von 4096 Punkten 240 Sekunden, wobei reine Gleitkommaarithmetik verwandt wurde. - Interessierte Leser können sich an die Autoren wenden. - Lit.: 1) Cochran, W.T.; Cooley, J.W.; und andere, "What is the Fast Fourier Transform", Proceedings of the IEEEE (1967) 55, 1664-1674. □

Software-Besprechung

Nachstehend die kurze Besprechung eines Software-Paketes für Adreßverwaltung der Münchener Firma Schiessl & Steiner. Der Verfasser würde es begrüßen, wenn andere Anwender größerer Software-Pakete ebenfalls Besprechungen liefern würden, damit der Markt transparent wird. Auch ein Gedankenaustausch mit anderen Benutzern der gleichen Software wäre angenehm.

Die Software wird auf Diskette geliefert und läuft auf dem CBM 3032. Die vom Verfasser gekaufte Version läuft nicht auf dem 8032. Der Anschaffungspreis betrug um die DM 600,-. Die Diskette ist nicht kopierbar. Eine Reserve-Diskette kann man für DM 50,- dazukaufen. Die Software ist zum Schutz des Urheberrechts bewußt verschlüsselt geschrieben, teilweise nicht einmal auflistbar. Ohne intensive Detailkenntnisse, auch in der Hardware, und etwas Forscherdrang sind Änderungen praktisch nicht möglich. Bei Glaubhaftmachung der Seriosität gibt die Verkäuferfirma wenigstens Hilfestellung für geringfügige Änderungen, die dann auf der Original-Diskette dazukopiert werden.

Es gibt einige wenige Programmfallen, aus denen man nur durch Abschalten oder mit selbst eingebauter Reset-Taste wider herauskommt. Grundsätzlich ist das Paket aber benutzerfreundlich und am Bildschirm selbst erklärend. Einige Formulierungen könnten aber mehr auf den Endbenutzer Rücksicht nehmen. Warum muß eigentlich eine Datentypistin beispielsweise erst durch Fehler lernen, daß 'sektorieren' 'löschen' bedeutet und daß die Wiederholung mehrtätige Arbeit bedeutet. Der Verfasser setzt problemlos ungelernte Hilfskräfte im Dauerbetrieb ein. Verknüpfung mit zusätzlich zu kaufender Textverarbeitung

ist möglich. - Ernsthafte Mängel sind: Keinerlei Sortiermöglichkeit und keine Modifizierbarkeit des Vollausdrucks, d.h. man kann die Verschwendung von programmierten Leerzeilen nicht verhindern, kann keines der Felder im Vollausdruck unterdrücken und kann den Ausdruck nicht formatieren.
Für ein Pilot-Projekt der Verwaltungsrationalisierung benutzt der Verfasser in der KKB Kundenkreditbank seit vier Jahren das IBM-Timesharing-System CALL. Bei dem Versuch, die Datenbestände in einen CBM zu überführen, **wurde das Datenbankprogramm PETAID getestet** (Chip, Oktober 1980, Seite 98). Hier ein kurzer Erfahrungsbericht.

Die Software wird auf Diskette geliefert und ist kopierbar. Sie durfte vor Bezahlung sogar drei Wochen getestet werden. Preis rd. DM 1000,-. Die Dokumentation ist umfangreich, die Bedienerführung ebenfalls, aber nur in englischer Sprache und teilweise nicht auf die Bedürfnisse eines nichtfachmännischen Endbenutzers zugeschnitten. Die Programme sind auflistbar und enthalten sogar einige REM-Dokumentationen. Trotzdem war es dem Verfasser trotz vielstündiger Arbeit nicht möglich, für DFÜ-Verknüpfungen die Input-Routine zu finden, weil das Programm, scherzhaft formuliert, mehr Variablen als Programmzeilen hat. Folgende weitere Mängel haben die Benutzung bisher ebenfalls verhindert:

Keine Möglichkeit, an eine einmal definierte Datei Felder anzuhängen. Keine echte Sortierung, nur das Erzeugen von Index-Dateien, die auf der Programmdiskette automatisch dazugespeichert werden und nach jeder Dateiänderung überholt sind. Sortierung nur nach einem Feld. Ausdruck sortierter Dateien nur mit dem Schlüsselfeld und dem jeweiligen Sortierfeld, d.h. kein Report-Generator, keine Summenbildung. Keine Möglichkeit, Feldinhalte zu errechnen. Die Software stammt angeblich aus Großbritannien, also Schwierigkeit, sich mit dem Urprogrammierer in Verbindung zu setzen.

Diese Besprechung soll kein Verriß sein. Wer mit den angegebenen Einschränkungen leben kann, müßte recht gut bedient sein.

Klaus Schünemann, Köln, Tel.: 0221-23 60 47.

Development Aids der Firma GWK für den AIM 65. Herr Mathias Helm aus Clausthal-Zellerfeld nahm die Produktbesprechung vor: Seit einiger Zeit besitze ich die 'Development Aids' der Fa. GWK-Elektronik, ein 4 KByte-langes Programmpaket für den AIM 65 mit einer eigenen Kommandoebene, von der sich durch Tippen je eines Buchstabens unter anderem folgende Dienstleistungen abrufen lassen:

- DISASSEMBLE TO EDITOR. Dieses Programm ist durch die gewählte 2-Pass-Methode schnell und nutzt den vorhandenen Speicherplatz optimal. Aus 4 K Objektprogramm wird so in ungefähr 2 Minuten eine Quelle von 20K. Recht komfortabel ist der zugehörige Editor für die Spezifikationstabelle zum Unterscheiden von Programm, .WOR, .BYT, Programmlücken und zur Festlegung des Programmendes, der auch für den im Programmpaket befindlichen Relocator benutzt wird. Er enthält auch "Insert-' und 'Kill-'Befehle. Der Output des Disassemblers läßt sich auch auf 'Tape' usw. richten.

- CROSS LIST. Über Objektprogramm werden Listen von Referenzen zu angebbaren Bereichen erstellt, leider nicht geordnet in Form einer Cross-Reference-List.

- Nützlich sind die Features MOVE, FILL, SEARCH BYTE, SEARCH WORD und DISPLAY AND ALTER MEMORY, bei dem pro Zeile 16 Bytes und eine ASCII-Darstellung geboten werden.

- Ferner noch vorhanden: SYN WRITE und SYN READ zum Einstellen des Cassettenrecorder-Interfaces bzw. zum Einstellen des Cassettenrecorders auf die Spurlage einer 'fremden' Cassette, SET DILINK TO hex EF05, DISASSEMBLE, und ein ASSEMBLER-REFORMATTER.

In absehbarer Zeit wird die GWK dieses Programmpaket aufspalten in ein Paket mit zusätzlichen Monitor-Dienstleistungen und ein Paket mit Disassembler in den Editor usw. gleichzeitig werden auch die bis jetzt vorhandenen 'Unverträglichkeiten' mit TV-Interface-Betriebssystemen beseitigt, die den Display-Pointer des AIM 65 (CURPO2) manipulieren. ¤

Expansion für CBM und AIM 65 Die Neudecker-Elektronik in Berlin liefert jetzt den Proportionalschrift-Nadeldrucker Centronics 737 mit einem speziell für CBM entwickelten IEEE 488-Interface mit Codewandler PET/ASCII. Damit ist die volle Benutzung anspruchsvoller Textverarbeitungsprogramme wie WORDPRO 3 und 4 möglich. Schreibmaschinen, wie die ES100 können nach Umrüsten mit dem IEC-Interface am CBM betrieben werden. An den Application Connector des AIM 65 kann der Centronix 737 ohne Interface angeschlossen werden Software wird mitgeliefert. Neu im Vertriebsprogramm von Neudecker sind auch hochauflösende Farbmonitore und ein grüner 12"-Monitor mit 20 MHz von NEC. ¤

Buchbesprechung

Taschenrechner + Mikrocomputer Jahrbuch 1981 Anwendungsbereiche, Produktübersichten, Programmierung, Entwicklungstendenzen. Mit 139 Abb., 36 Tabellen, 59 Programmen und Adressen, Vieweg-Verlag, Braunschweig, 1980, 296 Seiten, kart. DM 24,80, ISBN 3-528-04179-X.

Das wiederum von Harald Schumny herausgegebene Vieweg-Jahrbuch enthält die Abschnitte Taschenrechner, Mikrocomputer, Programmsammlung, Praxisprobleme, Daten, Sachwortverzeichnis. Aus der Fülle der sorgfältig bearbeiteten Information sind für den Leser dieser Zeitschrift besonders erwähnenswert Artikelzur Sprachauswahl, zu PASCAL und zu den neuen 16 Bit-Prozessoren im zweiten Abschnitt. Im Programmteil ist für den AIM die nichtlineare Regression beschrieben, für den CBM die Umwandlung von 16 Bit Binärworten in BCD. Auf vier Seiten wird ferner eine Übersicht über in Fachzeitschriften veröffentlichte Mikrocomputer- und Taschenrechnerprogramme, geordnet nach Sachgebieten ggegen. - Zusammen mit den Übersichten und Anschriften ein wiederum empfehlenswertes Handbuch.

Programmierung des 6502, Rodnay Zaks Deutsche Erstausgabe der dritten amerikanischen Auflage, MSB-Verlag, Markdorf 1981, 350 Seiten, ISBN 2-902414-25-0, DM 44,--.

Das Buch traf bei Redaktionsschluß hier ein, so daß noch keine umfassende Besprechung stattfinden kann. Die gut lesbare Übersetzung besorgte Bernd Pohl. Der Inhalt und der Umfang der Abschnitte gliedert sich wie folgt: Grundlagen (28), Hardware-Organisation des 6502 (12 Seiten), Grundlegende Programmiertechniken (40), Der 6502 Befehlssatz (82), Adressierungsarten (20), Ein- und Ausgabetechniken (40), Ein- und Ausgabebausteine (12), Anwendungsbeispiele (12), Datenstrukturen (64), Programmentwicklung (22) und Anhang (15 Seiten). - Gegenüber der seinerzeit besprochenen und kritisierten ersten amerikanischen Auflage fällt die vermehrte Präsentation von Programmierbeispielen auf. Es ist allerdings zu bemerken, daß der Druck der Assemblerlisten in mehreren Fällen zugelaufen ist, so daß eine Entzifferung erschwert ist. - Auf den Inhalt werden wir noch zurückkommen.

Editorial

Die Serie 'Ein- und Ausgabe am AIM 65' wird im kommenden Heft fortgesetzt. Eine ähnliche Serie wird für den CBM vorbereitet. Heft 18 wird ferner ein Programmpaket für eine virtuelle 16-Bit-CPU enthalten, die mit der 6502 simuliert wird. Eine weitere Reihe wird dem 6809 gewidmet sein, Architektur, Signale, Befehlssatz, Programmierbeispiele.

Sprachen: Das Rockwell-FORTH für den AIM 65 soll jetzt ab März ausgeliefert werden. Wie so oft verspätet sich die Lieferfähigkeit gegenüber der frühzeitigen Ankündigung. PL 65: Obwohl diese Sprache für den AIM 65 bereits erhältlich ist, wurde sie hier noch nicht implementiert. Eine Besprechung aus der Leserschaft ist daher willkommen, vor allem in Hinsicht auf die Hantierbarkeit und auf die Erleichterungen, die sie gegenüber der Programmierung unter Assembler bietet. PL 65 ist eine strukturierte Sprache, deren Statements assemblernah sind und deren Ausgabefile ein Quelltext ist, der mit dem Assembler zu Maschinencode weiterverarbeitet wird.

An der Sprachfront des CBM darf man einige Bewegung erwarten. Am 5. und 6. Februar 81 fand hier in Ahrensburg ein Arbeitstreffen statt, in dessen Verlauf ein BASIC-Precompiler vorgestellt wurde, der offensichtlich vor der Markteinführung steht. Er erlaubt strukturierte Programmierung in Verbindung auch mit der üblichen Freistil- oder 'Spagetti-'Programmierung unter BASIC. Er erlaubt also statements wie DO, WHILE, CASE, END und liefert als Ergebnis BASIC-Statements. Bemerkenswert ist die Modularisierung: Unterprogramme und erklärte Funktionen arbeiten mit lokalen Parametern, die ihnen, wie in anderen Sprachen, vom Aufrufer übergeben werden. Erzeugte und getestete Module können für die dauerhafte und künftige Verwendung unter Namen auf Diskette abgespeichert und mit ihrem Aufruf eingebunden werden. In der Testphase eines größeren Programmes kann auch mit noch nicht ausprogrammierten Programmstummeln operiert werden.

Strukturiertes BASIC mag für den Gelegenheitsprogrammierer vielleicht zu gewöhnungsbedürftig sein, der Profi darf aber auf Sicht echte Erleichterungen und Zeitersparnisse erwarten. Es mag sein, daß man einem solchen Precompiler wegen der beliebten Grundsprache BASIC mit ihren bequemen Stringbearbeitungsmöglichkeiten den Vorzug gegenüber PASCAL geben wird.

Auch ein echter BASIC-Compiler, der eine schon beachtliche Teilmenge des Befehlssatzes in ausführende Maschinensprache übersetzt, wurde im Lauf gezeigt.

Der CBM PASCAL-Compiler (in diesem Heft besprochen) arbeitet noch mit dem intermediären P-Code, der zur runtime kurz interpretiert und ausgeführt wird. Die Entwicklung geht aber auch hier weiter, und zwar hin zur Übersetzung des Quellenprogrammes in den schnell ausführenden Maschinencode. Ein entsprechender Integer Subset Compiler aus unabhängiger Quelle wurde während des Treffens weiter bearbeitet.

Den Veröffentlichungen in dieser und auch in anderen Zeitschriften ist zu entnehmen, daß allenthalben am Ausbau der Sprachmodule (Assembler, BASIC) gearbeitet wird. Soweit von hier aus zu übersehen, gibt es aber noch keine Bemühungen ein dezimal rechnendes BASIC zu kreieren oder einen dezimalen Datentyp unter BASIC ggfs. mit definierbarer Feldlänge, wie er für kaufmännische Anwendungen benötigt würde. Es kann der 6502-Prozessor dezimal rechnen. Das verbreitete Microsoft-BASIC macht davon keinen Gebrauch. Die binären Datentypen Integer und Gleitkomma erlauben zwar eine kompakte Abspeicherung von Variablen und ein schnelles Multiplizieren und Dividieren, sie bedingen aber auch Ungenauigkeiten des Ergebnisses. Man sollte das bei künftigen Arbeiten an den Sprachen bedenken. Mit den größeren zur Verfügung stehenden Speichern der nächsten Generation (128 KB, möglicherweise auch 4 MB Adressierfähigkeit) ist der vermehrte Platzbedarf dezimaler Variabler kein gewichtiger Grund mehr gegen ihre Implementierung. - Die Werkzeuge für solche Arbeiten sind weitgehend veröffentlicht, u.a. auch in den ersten 6 Heften dieser Zeitschrift, dort neben vielen anderen wichtigen Grundlösungen, für die wir häufig nutzbringende Weiterbearbeitungen finden.

Hardware: Wie es scheint, kommen in diesen Tagen die ersten APPLE III zur Auslieferung. - Bei Rockwell bleibt es bei der Ankündigung des AIM 65/40 zum Frühsommer (?). - Nach den Andeutungen darf man im Jahresverlauf auch von anderer Seite mit leistungsfähigeren 6502-Maschinen rechnen. Der Herausgeber vermutet aber, daß wir hierzulande wegen des rapiden Kursanstieges für den US-Dollar eher mit relativen Preiszuwächsen rechnen müssen, denn mit den schon gewohnten Verhilligungen. Für amerikanische Fachliteratur wurden aus diesem Grunde die Preise schon drastisch erhöht.

Das Maß der in unserem Land nicht nur außenwirtschaftlich bedingten Preissteigerungen veranlaßte andere Fachverlage bereits zur Jahreswende zur Anhebung ihrer Abonnementspreise. Auch das 65xx MICRO MAG wird zum 1. April 1981 etwas teurer (Abo im Inland DM 49,--, im Ausland DM 54,--). Der Leser darf vergewissert sein, daß die Bemühungen weiterhin auf Bewahrung der Leistungsspitze in der Darbietung ernsthaft verwertbarer Softwareinformation gerichtet sind. □

English Summary

PASCAL für CBM describes the new Commodore release of disk based TCI. PASCAL and its special command and handling features on the CBM. - Kaufmännisches Rechnen auf dem CBM offers a calculating package for business applications which does not use floating point numbers but ASCII and yields accuracy to the least digit. - Grafik-Zugriff gives acces to graphics for the CBM with business keyboard. - 1/4 Grafik allows to plot curves on the CBM screen. - AIM mit fremder Systensoftware contains an interface for the OSI Floppy Disk and runs the AIM under OSI DOS and BASIC. - The articles 'Datenaustausch' and 'Rechnerkopplung' cover features of multiprocessor systems. - Following are articles on the quality and aging of tapes for high speed recording and complete BASIC statements by single stroke together with CTRL on the AIM. - LMR is a big utility for AIM to load to new location, to move or to move and relocate, even to non existent RAM. - Benutzung arithmetischer Interpreterroutinen gives instructions of how to use and enter arithmetical routines in machine language on the CBM. □

Abo-Aktion

Bei dieser bis zum 31. März 1981 (Poststempel) befristeten Abo-Aktion erhalten Sie als Dank für jedes neu angeworbene Abonnement der Zeitschrift 65xx MICRO MAG einen 24-poligen ROM-Sockel (DIL) mit 'Null Ziehkraft', der Ihnen das schnelle und verletzungsfreie Einsetzten und Wechseln von ROMs und EPROMs auf der Computerplatine ermöglicht.

Außerdem nehmen Sie für jedes neue Abonnement an einer am 4. April 1981 stattfindenden Verlosung (unter Ausschluß des Rechtsweges) teil, bei der folgende Preise verlost werden:

1. Preis: 1 SHARP BASIC-Taschencomputer PC 1210,
2.-5. Preis: je 1 Buch '6502 Software Design' von L.J. Scanlon,
6.-15. Preis: je 1 'Das Buch 1-6 des 65xx MICRO MAG'.

Die Namen der Hauptgewinner werden in dieser Zeitschrift veröffentlicht. — Benutzen Sie bitte in diser Abo-Aktion zur eindeutigen Zurodnung die beiliegende Bestellkarte!

Für AIM 65

Thermodrucker, 20-stellig, komplett	DM 129,--
Thermopapierrollen, 10 Rollen à 25 m	19,80
Alpha-Tastatur	79,--
16 K RAM-Erweiterung	500,--
12 K RAM-Erweiterung	420,--
8 K RAM-Erweiterung	340,--

Zum direkten Anschluß an alle 6502-Personal Computer, inkl. Steuerungssoftware:

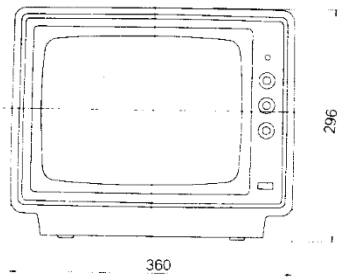
ECMA 34/41 kompakt Version - Kassettenlaufwerk
Read after write, Controler, Schreib/Leseverstärker
Netzteil, Kabel, Gehäuse

DM 2.495,--

EDS SIEBERT GMBH
ELEKTRONISCHE DATENERFASSUNGS-SYSTEME

2804 LILIENTHAL
KÖNIGSBERGER STR. 11, TELEFON 04298/4615

NEC 12" MONOCHROME CHARACTER DISPLAY



The JB-1201M Monochrome Character Display is designed for both character and graphic displays for use in combination with a micro computer.

Video Input Signal

The input is a composite video and sync. signal, featuring simpler connection.

Built-in Sound System

Built-in sound system greatly expands the range of uses possible.

Flat Picture Screen

The picture screen features a flat face for clearer, more easily read images.

658.- DM

SPECIFICATIONS

Picture Tube	C1270P31
	12" diagonal & 90° deflection
Phosphor	Green (P31)
Video Input Signal	Composite video signal
Polarity	Negative sync.
Level	1.0Vp-p
Impedance	75 Ω (switchable to higher impedance)
Input Terminal	Phono pin jacks
Active Display Area	231(W) x 175(H) mm
Scanning Frequency	
Horizontal	15.75kHz (63.5μS)
Vertical	60Hz (16.67mS) 50 Hz (20.0mS)
Active Video Period	
Horizontal	48.0μS max.
Vertical	14.61mS max. 17.78mS max.

Video Bandwidth	30Hz - 20MHz (±3dB)
Display Characters	80 characters with 25 lines
	5 x 7 dot matrix (8 x 8 dot/cell)
Sound Output Power	1.0W (10% distortion)
Sound Input Power	−36dBm (input impedance 5 kΩ)
	(0.1Vp-p)
Controls	
Inside	H. width, V. height, V. line, focus, sub brightness
Outside	Volume, brightness, contrast, H. hold, V. hold
Operating Ambient Temperature	0°C - +40°C
Power Supply	A type: 120V AC (±10%) 60Hz
	E type: 110V/120V/220V/240V 50/60Hz
Power Consumption	30W
Dimensions	360(W) x 296(H) x 330(D) mm
Weight	6.0kg

NOTE: The above specifications are subject to change without notice for further improvement.

DIPL.-ING. HORST NEUDECKER

Ingenieurbüro für Elektronik, Zweigstelle Elektronikladen, Mehringplatz 13, 1000 Berlin 61

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

Systemerweiterung für AIM 65 / PC 100 auf Europakarten

Einheitliches Format 100x160 mm, einheitlich nur +5 Volt. AIM-Expansion-Bus kompatibel, "S44-Bus", AIM-Software kompatibel, alle Karten auch mit 64-poligen a/c DIN-Stecker (MCS-Bus) lieferbar. Adreßdekodierung auf jeder Karte, anschlußfertig und ausführlich getestet, 1 Jahr Garantie. - Für den PC 100 gibt es Sonderausführungen, die mit der im Gerät vorhandenen Stromversorgung auskommen.

32 KB RAM-Modul - die meistgekaufte AIM-Speichererweiterung!
792,- + MWSt = 894,96 DM

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

VIDEO-Interface 615,- + MWSt = 694,95 DM
mit erweitertem Betriebssystem für Editor, Assembler, Monitor.

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

EPROM-Karte 32 KB für 8 Stück 2716 und 2532 oder 4+4 St. gemischt
ohne EPROMs 180,- + MWSt = 203,40 DM

BUS EXPANSION zusätzlich gepuffert, 6 Steckplätze, erweiterbar, lieferbar für 44- und 64-polige Karten
240,- + MWSt = 271,20 DM

ICs (Preise incl. MWSt), jeweils das Spitzenfabrikat (F, NEC, Motorola, TI etc., Mengenrabatt 10% ab 8 Stück pro Typ):

2114L/450 = DM 10,-	2114 CMOS = DM 18,-	4116/200 = DM 12,50
2708L/450 = DM 17,-	2716 = DM 27,-	2532 = DM 52,-
6502 = DM 28,-	6522 = DM 28,-	6532 = DM 32,-
6821 = DM 15,-	6845 = DM 74,-	2114L/200 = DM 11,-

Centronix-Matrixdrucker 737 7x9 Matrix, 3 Schrifttypen, auch Proportionalschrift, 3 Papierzuführungen, Mikroprozessorelektronik, Randausgleich und mehr Eigenschaften (Prospekt anfordern!).
Mit Centronix-Interface und AIM-Interfacesoftware DM 2.350,-

DIPL.-ING. HORST NEUDECKER

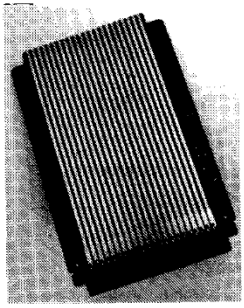
INGENIEURBÜRO FÜR MESSTECHNIK

MEHRINGPLATZ 13
1000 BERLIN 61

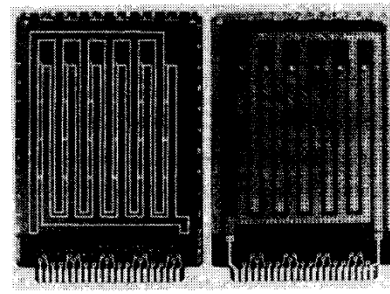
TEL.: 030- 614 89 00
030- 251 20 00

Verkauf von Leiterplatten, Steckadapter, Relaisplatinen

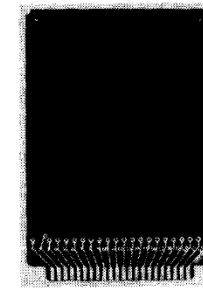
1



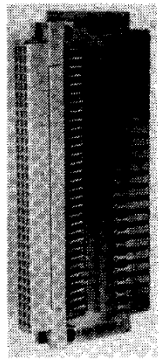
2



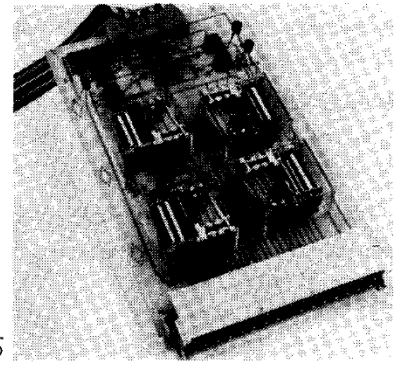
3

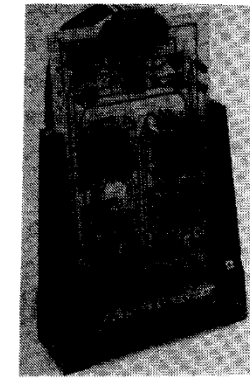


4



5





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

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

STECKER
INGENIEURBURO

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

SUPER — CBM — SOFTWARE
IM EPROM:

****

EXBAS'81 (C) P.ENGELS

==========

14 NEUE BASIC-BEFEHLE

#DEZ	:HEX-DEZ WANDLUNG
#HEX$	:DEZ-HEX WANDLUNG
#RAM	:SPEICHERBEGRENZUNG
#ALFA$	:SORTIERT STRINGS
#PLOT	:SETZT CURSOR AUF SCHIRM-KOORDINATEN
#RESTLINE	:RESTORE AUF PROGR. ZEILE
#STRING$	:INITIALISIERT STRING
#EXMO	:SPUNG IN MONITOR
#INSTR	:INSTRING - BEFEHL
#LINE	:INPUT MIT ALLEN SONDERZEICHEN
#CLINE	:#LINE VON CASSETTE
#SUB$	:AENDERT TEIL-STRING EINES STRINGS
#PRUS	:PRINT - USING (FORMATIERTER AUSDRUCK)
#OFF	:SCHALTET EXBAS'81 AUS

EXMO'81 (C) P.ENGELS

==========

11 NEUE MONITOR-BEFEHLE

B= BRANCHBERECHNUNG	C= CLEAR MEMORY
D= DISASSEMBLER	E= EXOR-BERECHNUNG
F= FLAG-ANZEIGE	Q= SRUNG INS BASIC
N= ADRESSUMRECHNUNG	T= MEMORY-TRANSFER
$= HEX-DEZ WANDLUNG	%= HEX-BINAER WANDL.
"= ASCII-BERECHNUNG	

NEW-ROM

==========

NEUES CBM-OP.-SYSTEM
INITIALISIERT EXMO & EXBAS BEIM EINSCHALTEN
(KEIN 'SYS' ERFORDERLICH)
ERWEITERT CBM UM TASTEN-REPEAT, BEI DEM GLEICH-
ZEITIG MIT 'SAVE' & 'LOAD' GEARBEITET WERDEN KANN

PREIS FUER 3 EPROMS (6 K-BYTE) + BESCHREIBUNG
+ CASS. MIT BEISPIELEN: 333,—DM

==========

PETER ENGELS
SCHWEIZERSTR. 17
5350 EUSKIRCHEN Tel.: 022 26 - 53 58

****

Neu im Fachliteratur-Angebot von MSB...

	Fuer Microprocessor 6 5 0 2 :			
MSB360C	6502 Software Gourmet Guide + Cookbook		39,--	Scelbi
MSB352A	6502 Assembly Language Programming	Leventhal	54,30	Osborne
MSB372A	Programming the 6502	R.Zaks	44,--	SYBEX
	Unsere *** N E U E *** Uebersetzung			
MSB372D	PROGRAMMIERUNG fuer 6502 (in Deutsch)	R.Zaks	44,--	M S B
MSB373E	6502 Applications - Book	R.Zaks	36,--	SYBEX
MSB374E	Programming A Microcomputer 6502	Forster	32,--	Addison
MSB379E	6502 Games - N E U -	R.Zaks	36,--	SYBEX
	Die Zeitschrift MICRO 6502 (engl.) + Zusammenfassungen:			
	MICRO 6502 The Manazine of the APPLE,KIM,SYM			
	AIM, PET/CBM and other 6502 SYSTEMS	M.Tripp	8,--	MICRO-INK
MSB229E	Abonnement fuer 12 Hefte		79,--	
MSB271E	Best of MICRO Vol. 1 (Heft 1-6)		32,--	MICRO-INK
MSB222C	Best of MICRO Vol. 2 (Heft 7-12)		32,--	MICRO-INK
MSB225E	Best of MICRO Vol. 3 (Heft 13-24)		39,--	MICRO-INK
MSB412C	6502 Software Design	Scanlon	38,--	TAB
MSB223C	PET and the IEEE 488 Bus (GPIB)	Fisher/J.	51,10	Osborne
MSB224E	PET Personal Computer Guide (2.Edition)	Donahue	47,10	Osborne
	Buecher von Rockwell, MOS Technology			
MSB371E	R 6500 Hardware Manual	Rockwell	12,--	
MSB371D	6502 Hardware Handbuch (deutsch)	MOS	24,--	
MSB376E	R 6500 Programming Manual	Rockwell	12,--	
MSB376D	6502 Programmier Handbuch (deutsch)	MOS	28,--	
MSB378C	AIM-65 Monitor Listing komplett	Rockwell	18,--	
MSB377E	AIM 65 Users Guide	Rockwell	22,--	
MSB377D	AIM-65 Anwender Handbuch (deutsch)	Rockwell	28,--	
	Buecher fuer BASIC-Spiele + LERNUNTERLAGEN:			
MSB120E	The BASIC-Handbook	Lien	44,--	Compusoft
MSB402C	Learning LEVEL II (fuer TRS -80 User)	Lien	45,--	Compusoft
MSB121E	The BEST of Interface Age Vol.1			
	Software in BASIC (vier Interpreter)	Warren	39,--	
MSB411E	Best of INTERFACE AGE Vol. II		32,--	
MSB122E	BASIC Computer Games I (101)	Ahl	30,-	C.C.P.
MSB123E	More BASIC Computer Games II (84)	Ahl	30,--	C.C.P.
MSB040E	40 Computer Games from XB MICROCOMPUTING		24,--	W. Green
MSB124E	BASIC and the Personal Computer	Dwyer	38,--	Addison
MSB129E	8K Microsoft BASIC Reverence Manual		19,80	Hofacker
MSB130E	Calculating with BASIC (Math)		30,--	Scelbi
MSB504E	Problems for Computer Solution Lehrer Ausgabe		39,--	C.C.P.
MSB505E	Problems for Computer Solution Schueler Ausgabe		24,--	C.C.P.
MSB510E	Computers in Math. Sourcebook of Ideas		65,--	Scelbi
MSB231E	Your FIRST COMPUTER	R.Zaks	22,--	SYBEX
MSB232E	MICROPROCESSORS from CHIPS to SYSTEMS	R.Zaks	44,--	SYBEX
MSB233E	Microprocessor INTERFACE Techniques	R.Zaks	44,--	SYBEX
	Unser BESTSELLER **********			
MSB280D	Microprocessor INTERFACE Techniken 2.deutsche Auflage, jetzt 440 Seiten.	R.Zaks	44,--	M S B
MSB254E	Inside BASIC Games	R.Zaks	42,--	SYBEX
MSB250E	58 BASIC Exercises	R.Zaks	38,--	SYBEX
MSB318E	Introduction to PASCAL (U.S.C.D.)	R.Zaks	39,--	SYBEX
MSB320E	PASCAL Handbook	R.Zaks	42,--	SYBEX
MSB380E	CP/M Handbook ***** N E U *****	R.Zaks	39,--	SYBEX
MSB711E	Computershop's 5-1QB u. CP/M Sammlung		9,--	M S B

Preise incl. 6,5 % Mwst. plus Porto
Wir versenden Porto-Frei, wenn der Bestellung ein Scheck beiliegt.

Fachliteratur MSB VERLAG

MSB Verlag
M Nedela
Postfach 1420

D-7778 Markdorf
Tel 07544 3575
Telex 734628 msb d

GWK

FÜR TECHNISCHE ELEKTRONIK mbH.

NEU! NEU! NEU! NEU!

12 KByte EXTENDED BASIC

für

AIM 65 / PC 100

Sehr umfangreiches und komfortables BASIC,

u.a. mit folgenden zusätzlichen Befehlen:

INDEVICE	OUTDEVICE	RENUMBER
EDIT	QUIT	FORMSTR$
HEXSTR$	BINSTR$	INSTRING
INPUT LINE	AUTO	HELP
ON ERROR GO TO	XOR	TRACE
DPEEK (16 Bit)	DPOKE (16 Bit)	

Preis: DM 445,-- + MWSt + Datenträger

Erhältlich auf Cassette, Diskette, EPROM.

GWK Gesellschaft für Technische Elektronik mbH.

D-5120 Herzogenrath

Asternstraße 2

Tel.: 024 06 - 623 94

65xx MICRO MAG

COMPUTING · SOFTWARE · HOBBY

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

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

Bezugsbedingungen: Abonnement ab laufender Ausgabe für 6 Hefte DM 45,-- (Inlandsendpreis). Ausland/Foreign via surface mail DM 50,--, USA air $ 27. Abonnements laufen bis auf Widerruf mit Kündigungsmöglichkeit bis zu zwei Wochen vor deren Ablauf.
Neue Abonnementspreise ab 1.4.1981: DM 49,-- /Inland, DM 54,-- /Ausland, $ 29,-- USA air.

Nachliefermöglichkeiten: Die Hefte 1-6 sind nur noch als Buch lieferbar, siehe Anzeige unten. Die Hefte 7-16 können einzeln oder komplett nachgeliefert werden, und zwar zum Endpreis von DM 7,80/Stück.

Private Besteller werden um Überweisung oder Scheck zusammen mit der Bestellung gebeten. Richten Sie bitte Ihre Überweisungen auf das Konto Roland Löhr, Nr. 20/01121 bei der Vereins- und Westbank in Ahrensburg, BLZ 200 300 00 oder auf das Postscheckkonto Roland Löhr PSchA Hamburg, Nr. 654 70-202.