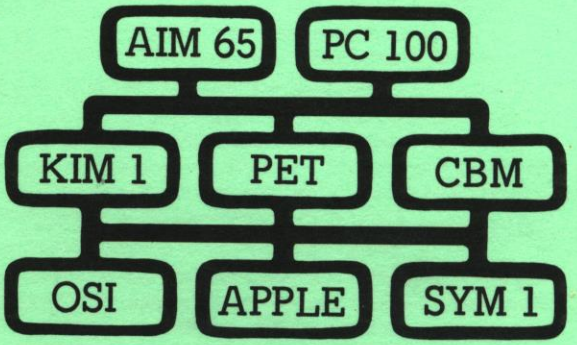
65xx MICRO MAG

COMPUTING · SOFTWARE · HOBBY

DM 9,50 Nr. 33 Oktober 1983



Inhaltsverzeichnis

DUO Plott Interface

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

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

Umrüstsatz MX80 F/T auf MX80 Typ 3

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

Für COMMODORE

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

Speichererweiterung auf 96 K

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

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

Drucker:

RX 80 DM 998,-- + MWSt

ITOH 8510 DM 1495,-- + MWSt

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

Stellberg Computer-Systeme

COMMODORE EPSON C. ITOH SOFTWARE INTERFACE

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

Michael Zimmermann, 6102 Pfungstadt

Mini Floppy Interface für AIM

1. Einleitung

In letzter Zeit hat sich der Preisverfall bei einer Vielzahl nicht nur elektronischer, sondern auch elektromechanischer Komponenten fortgesetzt. Damit sind Bauteile, die noch vor kurzem jenseits der Möglichkeit des Hobbyisten waren, in den Mittelpunkt des Interesses gerückt. Als ein besonders erstrebenswertes Zubehör für jeden Mikrocomputer ist hier eine Floppy Disk anzusehen wegen der allseits bekannten Vorteile für die Speicherung von Daten und Programmen. Bei Preisen von weniger als DM 500,- für ein Mini-Laufwerk sollte es nahezu für jeden Anwender erschwinglich sein.

Ziel dieses Aufsatzes soll es sein, möglichst einfache und kostengünstige Wege aufzuzeichnen, um ein derartiges Laufwerk an den AIM anzuschließen. Auf besonders hohe Speicherdichte oder sonstige Finessen, so z.B. DMA soll dabei im Interesse einer besonders einfachen Konstruktion verzichtet werden. Ebenso wird auf allgemeine Überlegungen zum Aufbau von Laufwerken oder Aufzeichnungsmethoden nicht eingegangen. Auf die entsprechende Literatur von Lesea/Zaks (5), BASF (2) oder Motorola (6) sei verwiesen.

2. Formen der Ansteuerung von Floppy-Disk-Laufwerken

Für die Ansteuerung solcher Laufwerke gibt es, wie für andere Aufgaben auch, eine Reihe von Lösungen, die hier beschrieben und im Sinne der Auswahl einer möglichst einfachen Lösung gegenübergestellt werden sollen.

2.1 Indirekte Ansteuerung

Unter indirekter Ansteuerung soll hier jene Art von Kontrolle verstanden werden, die nicht die steuernden Fähigkeiten des Prozessors selbst ausnutzt, sondern dies indirekt über eine Schnittstelle bewerkstelligt. Diese Form wird von Commodore bevorzugt, entweder über den IEEE-488-Bus oder über eine V-24-Schnittstelle.

Einer Einsparung am AIM würden hier erhöhte Kosten für den eigentlichen Controller gegenüberstehen, dieser müßte über eine eigene Intelligenz verfügen. Diese Lösung kann im Sinne eines erhöhten Durchsatzes erstrebenswert sein, dem Ziel einer kostengünstigen Lösung wird sie aber nicht gerecht.

2.2 Controller mit TTL-Schaltungen

Diese Form der Ansteuerung stellt den Standard in der Mitte der 70-er Jahre dar, eine Beschreibung findet sich bei Motorola (6). Der Aufwand an Bausteinen (mindestens 30) und der Stromverbrauch sind beträchtlich. Aus diesem Grund dürfte ein derartiger Entwurf für einen kostengünstigen Controller nicht geeignet sein.

2.3 Ansteuerung mit unspezifischen Peripheriebausteinen

Eine nicht unwesentliche Einsparung an Bauelementen und im Stromverbrauch ergibt sich bei Verwendung normaler Peripheriebausteine. Hierfür sind dem Verfasser mehrere Entwürfe bekannt geworden, so in allgemeiner Form von Welles (12). Er verwendet nur für die Serialisierung der Daten einen Universal Synchronous Receiver-Transmitter AMI S2350 und sonst TTL-Schaltkreise. Von Allen (11) wird ein RAM als Zwischenspeicher benutzt, in dem die Daten im Bitmuster der Speicherung abgelegt werden. Hierdurch werden Microprocessor und Floppy Disk mit ihren unterschiedlichen Geschwindigkeitsbedürfnissen entkoppelt. Für die daneben auftretenden reinen Steuerungen des Laufwerkes werden auch hier TTL-Schaltkreise benutzt.

Auch Motorola bietet auf dieser Basis eine Lösung an (7), sie beruht auf dem MC6852 für Serialisierung und Deserialisierung der Daten und einem MC6820 für sonstige steuernde Aufgaben. Für den 6500 wurde ein derartiger Controller bereits vor recht langer Zeit von OSI entwickelt, dessen Anpassung an den AIM hat Schnell (11) bereits in dieser Zeitschrift beschrieben.

In neuerer Zeit hat sich Elektor (3) dieses Entwurfes angenommen. Auf die prinzipielle Möglichkeit, diesen Controller im Zusammenhang mit dem AIM zu betreiben, wird in diesem Aufsatz hingewiesen, es fehlen jedoch nähere Angaben. Insgesamt stellt diese Form des Anschlusses hinsichtlich der Hardware eine kostengünstige Alternative dar, erfordert aber bei der ansteuernden Software einen erhöhten Aufwand.

2.4 Ansteuerung mit spezifischen LSI-Bausteinen (Controllern im eigentlichen Sinne)

Für den Zweck der Ansteuerung von Floppy Disk-Laufwerken wurden von den unterschiedlichen Bauelemente-Herstellern eine Reihe von spezifischen Controllern entwickelt, die den Prozessor von einer Vielzahl der hierbei immer wieder vorkommenden Kontrollaufgaben entlasten. Die unterschiedlichen Typen bzw. Familien sollen hier untersucht und in ihrer Verwendbarkeit am AIM einander gegenübergestellt werden.

2.4.1 Western Digital 1771

Der FD1771 ist einer der ersten Bausteine dieser Klasse. Er dürfte deswegen auch die meisten Abkömmlinge hervorgebracht haben. In der Grundkonzeption ist er hardwaremäßig für den Anschluß an einen 8080 vorgesehen. Er hat deswegen einen invertierenden Datenbus, getrenntes Read- und Write-Enable und erfordert 3 Spannungen. Die Anforderungen an den Controller werden durch Einschreiben in direkt adressierbare Register formuliert, ebenso steht das Arbeitsresultat unmittelbar in einem Statusregister zur Verfügung. Insgesamt ist dieser Controller mit seinen Abarten sehr weit verbreitet, so sind u.a. Siemens und National Zweitlieferanten.

2.4.2 Western Digital 179x-01

In dieser Reihe existieren verschiedene Modelle, teilweise mit inverted, teilweise mit true-Data-Bus, neuere Ausführungen mit Side-Select-Output, in jedem Falle aber mit doppelter Spannungsversorgung 5V und 12V. Gegenüber dem FD1717 weisen der 179x und die folgenden Controller dieser Familie eine vereinfachte Read-Clock auf. Für einfache Dichte bei Mini-Disks reicht eine 1 MHz-Clock, die z.B. aus dem Takt Ø2 abgeleitet werden kann. Die Preise für diese Familie bewegen sich bei DM 90,-, ebenso wie bei den folgenden Beschreibungen (Einzelstücke im Plastikgehäuse).

2.4.3 Western Digital 279x-02

Diese Controller entsprechen in den Merkmalen den FD179x-01, erfordern aber nur eine Spannung und senken die Anzahl der Support-Chips durch integrierten PLL. Die Preise hierfür liegen bei DM 160,-.

2.4.4 Synertec 179x-002

Die Controller SYP179x-002 entsprechen in den Merkmalen den FD179x-01, erfordern aber nur eine Spannung. Der Preis liegt bei DM 60,-.

2.4.5 Synertec 6591

Der SY6591 wurde aus dem vorgenannten entwickelt und der Busstruktur des 6500 angepaßt. Da recht wenig verbreitet, liegt der Preis bei DM 90,-.

2.4.6 Nippon Electric upd 765

Dieser recht verbreitete Controller von NEC wird z.B. auch als I8272 von Intel angeboten. Abwandlungen hierzu sind nicht bekannt. Der Baustein erfordert nur eine einzige 5V-Spannung, für einfache Dichte auf Mini Floppies ist eine 4 MHz-Clock erforderlich. Er wurde ebenso wie die FD1771 für die Bus-Struktur des 8080 ausgelegt, ist aber im Gegensatz zu ersterem für den Einsatz unter einem DMA-Controller optimiert.

Die Ansprache erfolgt über nur zwei Register, das Status- und das Datenregister. Hierbei spiegelt der Status nicht die eigentliche Situation der Floppy wider, sondern die Anforderungen des Datenregisters, so z.B. ob dieses eine Read- oder Write-Operation erwartet. Die Befehle an den Controller werden den eigentlichen Daten der Floppy vorangestellt, das Resultat folgt den Daten. Dadurch wird die Programmierung etwas umständlich und ist nicht so gradlinig wie beim FD1771,

der über ein jederzeit abrufbares Register für den eigentlichen Status der Operation verfügt. Die Preise für den upd765 liegen bei DM 80,-.

2.4.7 Motorola 6843

Ein weiterer interessanter Controller dürfte der MC6843 sein, der als 6800-Produkt der 6500-Bus-Struktur angepaßt ist. Er ist jedoch wenig verbreitet, er wurde noch in keinem hobbymäßigen Angebot gesehen. Eine weitere Schwachstelle dieses Typs ist die Formatierung der Diskette. Hier arbeitet der Controller mit doppelter Geschwindigkeit, zu schnell, um ohne DMA nur unter Prozessorkontrolle die Zeitanforderungen erfüllen zu können.

Weitere Controller sollen in diesem Zusammenhang nicht betrachtet werden, da sie wegen sehr starker Bindung an bestimmte Prozessorfamilien im Anschluß Schwierigkeiten machen werden oder nur eine geringe Verbreitung gefunden haben.

2.6 Ergebnis des Vergleiches

Von den dargestellten Controllern dürfte der SYP1793-002 der Baustein der Wahl sein. Er erfordert nur eine einzige Spannung (im Vergleich zum FD1793-01), ist als Mitglied der 1771-Familie verbreitet (im Gegensatz zum 6843), hat einen nichtinvertierenden Datenbus und erfordert keine gesonderte Clock (wie z.B. der upd765), ebenso sind Programmierung und Interfacing recht einfach. Beim Interface bietet auch der SY6591, bezogen auf den höheren Preis, keine wesentlichen Vorteile. - Mit diesen Erkenntnissen soll im Folgenden deswegen eine Schaltung auf der Grundlage des SYP1793-002 beschrieben werden.

Literaturliste

(1) Allen, David: A Minifloppy Interface. In BYTE 2/1978
(2) BASF: BASF 6106 Mini Disk Drive Interface Specification, Rev. 3, 1977
(3) Cuyper, J. de:Floppy Disk Interface für Junior. In Elektor 11-12 1982
(4) Fritzon, Richard: Data on Disk: Implementing File Systems. In Kilobaud 1/1981
(5) Lesea, Austin; Zaks, Rodney: Microprocessor Interface Techniken, Markdorf 1979
(6) Motorola: M6800 Microprocessor Applications Manual, 1975
(7) Motorola: A Floppy Disk Controller using the MC6852 SSDA and other MC6800 Microprocessor Family Parts. Application Note 764 1976
(8) Motorola: Floppy Disk Controller MC6843. Advance Information, 1978
(9) Nicholson, James; Camp, Roger: Build a Super Simple Floppy Disk Interface. BYTE 5-6 1981
(10) Rausch, Albrecht: Mini Floppy Interface. MC 10-11 1982
(11) Schnell, Dr. A.: AIM 65 mit 'fremder' Systemsoftware, 65xx MICRO MAG Nr. 17, 1981
(12) Welles, Dr. Kenneth B.: Build This Economy Floppy Disk Interface. Software for the Economy Floppy Disk. BYTE 2 1977 und BYTE 6 1977
(13) Western Digital Corporation: FD 179X-01 Floppy Disk Formatter/Controller Family. 1979

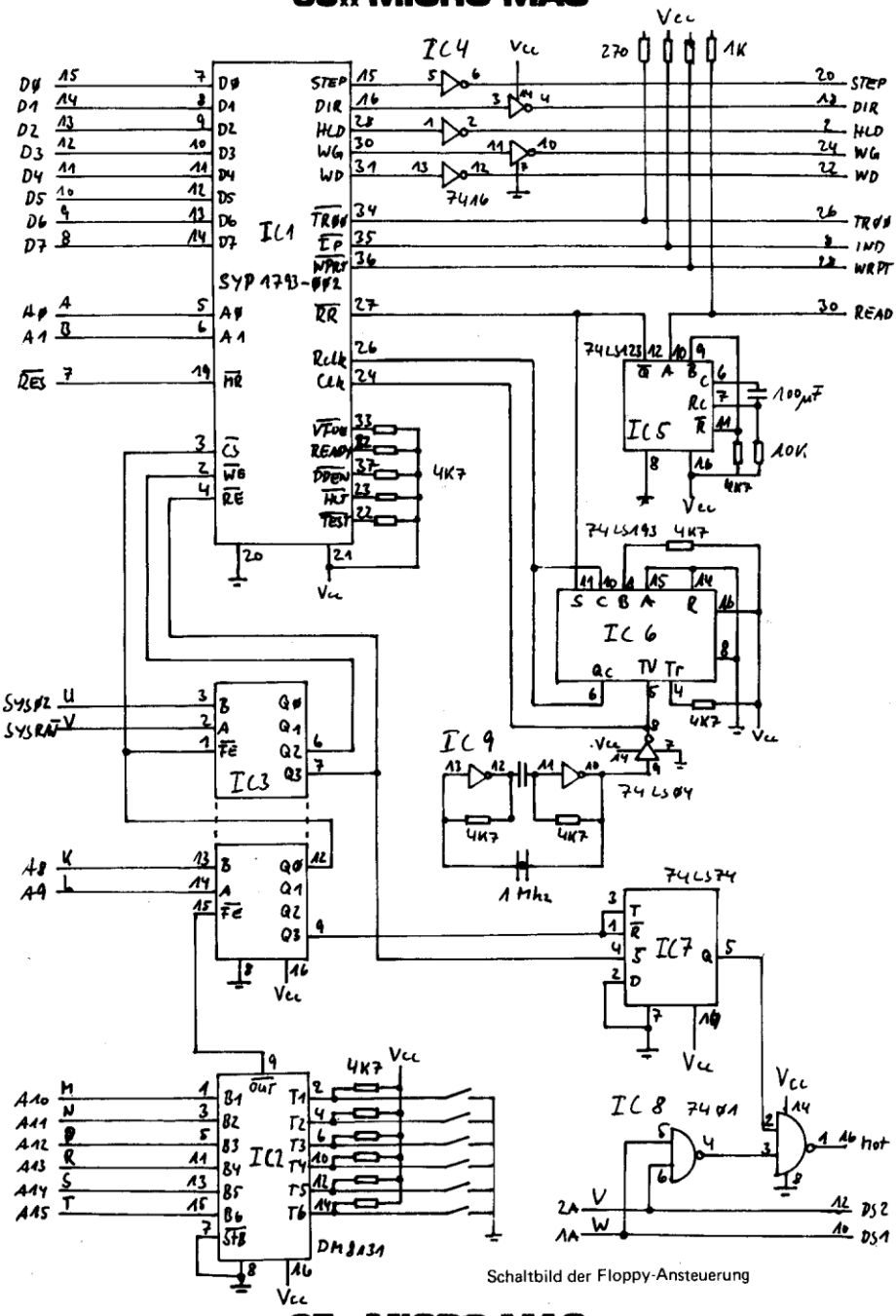
3. Hardware für das AIM-Floppy-Interface

Nach den allgemeinen Betrachtungen über die Auswahl des Controllers soll im Folgenden zuerst die hardwaremäßige Realisierung besprochen werden. Im Mittelpunkt des gesamten Aufbaues steht IC 1, der Controller SYP1793-002. Dieser stellt eine Verbindung zwischem dem Microprozessor und dem Floppy-Drive her.

3.1 Verbindungen zum Prozessor

Prozessorseitig ist der Controller an den Datenbus angeschlossen, auf Treiber kann hierbei verzichtet werden, da der Bus durch den Controller nur mit 1 LS-TTL-Last belegt wird. Die Adressierung erfolgt über die IC's 2 und 3; hierbei erlaubt IC2 (ein DM8131 Unified Bus Controller) eine Auflösung in 1 KB-Schritten, eine weitere Unterteilung nimmt IC 3 vor, ein 74LS139, der in Pages auflöst. Die erste Page im 1K-Adreßraum wird für die Ansprache des Controllers selber und gleichzeitig über IC 7 (74LS74) zum Erzeugen des Motor-On-Signales für das Floppy-Drive benutzt. Mit Adressierung der letzten Page wird dieses Motor-On-Signal wieder zurückgesetzt.

Die Adressierung der internen Register im Controller erfolgt über die Leitungen A0 und A1; ebenso ist das Reset-Signal des Prozessors direkt mit dem Master-Reset des Controllers verbunden. Die hier erforderlichen Read- und Write-Enable-Signale werden über die 2. Seite des IC 3 erzeugt.



Schaltbild der Floppy-Ansteuerung

Hierfür werden Ø2 und R/W benutzt. Alle diese Leitungen bedürfen keiner Treiber, da nicht mehr als LS-TTL-Lasten anliegen. Für die Ansteuerung von bis zu 2 Laufwerken werden die Ausgänge des AIM mitbenutzt, die normalerweise der Tape-Steuerung dienen. Dieser werden zusätzlich derart im IC 5 verknüpft, daß die Motor-On-Leitung nur geschaltet ist, wenn mindestens 1 Laufwerk selektiert wurde. Mit diesen Angaben dürfte der prozessorseitige Anschluß hinreichend beschrieben sein. Wir wollen uns jetzt der Beschaltung in Richtung Floppy-Drive zuwenden.

3.2 Verbindungen des Controllers zum Floppy-Drive

Der Controller ist zur Steuerung einer Reihe gewünschter Funktionen extern zu beschalten. Da hier nur mit einfacher Schreibdichte gearbeitet werden soll, muß der Double Density-Eingang über einen Wiederstand an Vcc liegen. Gleiches gilt für den HALT-Eingang, der nicht mit einem Verzögerungsglied beschaltet zu werden braucht, da die Head Load-Zeit bei den heute angebotenen Laufwerken hinreichend kurz ist. Ebenso ist der READY-Eingang über einen Widerstand an Vcc zu legen, da die Anlaufzeit über das Motor-On-Signal softwaremäßig gesteuert wird.

Die Eingänge TEST und Write-FAULT sind für einen ordnungsmäßigen Betrieb ebenfalls mit 4k7-Widerständen gegen Vcc zu terminieren. Die Ausgangssignale des Controllers, nämlich STEP, DIRECTION, HEAD-LOAD, WRITE-Gate und WRITE DATA werden direkt erzeugt und können ohne weitere Behandlung über IC 4 (Inverter/Driver 7416) dem Laufwerk direkt zugeführt werden. Gleiches gilt für die Eingangssignale TRACK-ZERO, INDEX-PULSE und WRITE-PROTECT. Hier reicht ein Abschluß mit 270 Ohm gegen Vcc aus. Aufwendiger ist der Anschluß der vom Laufwerk zu lesenden Daten. Hierfür ist ein Datenseparator erforderlich, der im folgenden Exkurs behandelt werden soll.

3.3 Data-Separatoren und ihre Eigenschaften

Allgemeine Aufgabe eines Datenseparators ist es, die Takt- und Datenimpulse im unbehandelten Lesesignal zu synchronisieren und zu trennen. Aus diesem Grunde soll auf die Form des Lesesignals kurz eingegangen werden. Für eine detaillierte Darstellung sei insbesondere auf Motorola (6) verwiesen. - Beim Lesesignal, wie es vom 1793 erzeugt und auch wieder vom Floppy-Laufwerk abgegeben wird, handelt es sich um eine frequenzmodulierte Darstellung, die bei einfacher Dichte eine Frequenz von 125 kHz aufweist. Dies bedeutet, daß eine Bitzelle 8 Mikrosekunden lang ist. Eine derartige Zelle wird durch einen Taktpuls von ca. 200 nsec Dauer gestartet. Liegt nach 4 usec ein ebenfalls 200 nsec langerDatenpuls an, so ist der Inhalt der Zelle 1. Fehlt dieser, so ist der Inhalt 0. Da das Beschreiben und Lesen der Daten durch die Drehzahl der Diskette auch mit mechanischen Vorgängen verknüpft ist, können diese Zeitbeziehungen nur bei der Abgabe vom Controller exakt eingehalten werden, nicht aber beim Lesen. Daher ist beim Lesen für jede Bitzelle eine Resynchronisation erforderlich.

Neben der Aufzeichnung der Daten werden bestimmte Steuerzeichen in Anlehnung an das von der Firma IBM entwickelte Verfahren dadurch gekennzeichnet, daß bei ihnen die Taktsignale ausgelassen werden, ein Tatbestand, dem bei Konstruktion des Datenseparators besonders Rechnung zu tragen ist. Die Erkennung von Takt und Daten im Lesestrom erfolgt durch die Read-Clock. Diese muß bei der 1793 jeweils zwischen Anliegen des Takt- und Datensignals ihre Phasenlage wechseln. Die eigentliche Polarität spielt im Gegensatz zu anderen Controllern keine Rolle.

Für den Aufbau eines Datenseparators wurde eine Reihe von Verfahren entwickelt. Während der Benutzer von 8-Zoll-Laufwerken diese Dienstleistung normalerweise bereits in der Laufwerkssteuerung integriert findet, ist dies bei 5 1/4-Zoll-Laufwerken durch eine Benutzerschaltung zu bewerkstelligen. Folgende Verfahren sollen kurz diskutiert und in ihrer Nutzanwendung verglichen werden:

1. Mono-Flop-Techniken
2. PLL-Datenseparatoren
3. Anwendung von Zählern

Die Anwendung von 2 Mono-Flops wird z.B. von Allen (1) beschrieben. Hierbei werden mit einem Takt-Impuls 2 Monoflops gestartet, von denen der erste nach 3 Usec, der zweite nach 5 usec ab-

fällt. In der Zeit nach Ablauf des ersten und vor Ende des zweiten Monoflops wird der Datenimpuls erwartet. Dieses Verfahren setzt also eine einwandfreie Synchronisierung auf die Taktimpulse voraus, ausfallende Takte sind hiermit nicht beherrschbar.

Ein sehr aufwendiges Verfahren ist die Takt-Datentrennung mit einem PLL. Wie von Motorola (6) beschrieben, wird die Spannung an einem VCO den Schwankungen in der Lesegeschwindigkeit nachgeführt. Dieses Verfahren kann auch bei ausfallendem Takt benutzt werden, erfordert aber einen beträchtlichen Aufwand an Bauteilen und ist in der Einstellung nicht ganz problemlos.

Bei einem Datenseparator auf der Grundlage eines Zählers wird dieser durch eine freilaufende Clock angetrieben und zusätzlich durch Takt- und Datenimpulse auf bestimmte Werte voreingestellt. Beschreibungen hierfür finden sich bei Nicholson/Champ (9) und Pausch (10). Eine derartige Schaltung läßt sich mit einer freilaufenden Clock und einem einzigen Zähler aufbauen. Ein zusätzliches Monoflop bringt die Signale auf eine einheitliche Breite. Auch ausfallende Takte werden hierbei bearbeitet, da der Zähler frei läuft. Es findet in diesem Fall nur keine Neusynchronisierung durch Nachladen des Zählers statt. Insgesamt besticht diese Methode durch eine hohe Betriebssicherheit (im Vergleich zu den Monoflops) und einen einfachen Aufbau (im Gegensatz zur PLL-Technik). Deswegen soll dies auch die Takt-Datentrennung unserer Wahl sein und im Folgenden beschrieben werden.

Ausgangspunkt ist eine freilaufende 1 MHz-Clock, die aus einem Quarz und IC 7, einem 74LS04, in bekannter Manier gebildet wird. Diese Clock treibt IC 5, einen Binärzähler 74LS193, der normalerweise frei läuft und dessen Ausgang QC die Read-Clock von 125 kHz bildet. Das unbehandelte Lesesignal vom Laufwerk wird über einen 1k-Widerstand terminiert und mit IC 5, einem Monoflop 74LS123 auf eine einheitliche Breite von 200 nsec gebracht. Dieses Signal wird einerseits in den Raw-Read-Eingang des Controllers geführt, andererseits dient es dem Zähler IC 6 zur Synchronisation. Hierzu wird bei Auftreten eines Signales an der Leseleitung des Laufwerkes auf die Mitte der gerade anliegenden Read-Clock synchronisiert, unabhängig davon, ob es sich um Daten oder Takt handelt. Dies bedeutet, daß entweder der Wert 2, bei Rolk=0, oder 6, bei Rolk=1, in den bis 8 laufenden Binärzähler geladen werden. Bei ausfallenden Taktimpulsen läuft der Zähler einfach weiter, lediglich die Synchronisation fällt aus, wird aber dafür durch den nachfolgenden Datenimpuls vorgenommen. - Trotz des recht einfachen Aufbaues weist der Datenseparator in der realisierten Form eine erstaunliche Betriebssicherheit auf. Es ist besonders zu vermerken, daß ein umständlicher Abgleich nicht erforderlich ist.

3.4 Modifikationen der Schaltung

In der vorliegenden Form befindet sich die hier beschriebene Steuerung der Floppy Disk beim Verfasser mit gutem Erfolg im Einsatz. Trotzden sollen noch einige Gedanken zu Vereinfachungen bzw. Erweiterungen angebracht werden. So ist es z.B. möglich, IC 7 zur Steuerung der Motor-On-leitung wegzulassen. Die Motorsteuerung wird dann alleine durch die Ansprache der Laufwerke bewirkt. Wegen gleicher Schaltkreise schlägt hier aber auch die Steuerung der Bandlaufwerke durch.

Ebenso kann auf eine gesonderte Adreßdekodierung verzichtet werden, wenn man die am Expansion Connector des AIM vorhandenen Leitungen ausnutzt. Werden die hier ebenfalls vorhandenen Signale RAM R/W für WE und invertiertes R/W für RE mitbenutzt, so sind IC2 und 3 überflüssig. Auch an der Clock sind Einsparungen möglich. Auf sie kann vollständig verzichtet werden, wenn Ø2 als 1 MHz-Referenz benutzt wird. Vertraut man weiterhin auf einwandfreie Signalabgabe vom Laufwerk, so ist auch IC 5 entbehrlich. Mit diesem 'Sparprogramm' hat man die Ansteuerung, zumindest beim Betrieb mit nur einem Laufwerk bei dauernd aktiviertem Motor, auf 3 IC's (1, 4, 6) zusammengestrichen, eine Lösung, die Nicholson/Champ (9) andeuten. Es soll aber nicht verschwiegen werden, daß abhängig von Umgebungseinflüssen, die Arbeitssicherheit unter diesen Maßnahmen leiden kann. - Ein genauer Aufschluß, wie weit und an welcher Stelle man einsparen kann, dürfte nur mit einem Oszilloskop möglich sein, das Auskunft darüber geben muß, wie sauber die Signale sind. Der Verfasser sah sich z.B. nicht in der Lage, Ø2 als Clock für den Binärzähler zu benutzen, da dieses Signal hierfür keine ausreichend steile Flanken mehr hatte.

Neben dieser Economy-Version sind auch durchaus Luxus-Modelle denkbar. So kann z.B. die Ansteuerung der Laufwerke von den Möglichkeiten des AIM entkoppelt werden. Hierfür wäre ein separates Latch erforderlich. Auch höhere Geschwindigkeiten sind erreichbar: Neben einem anderen Quarz in der Clock muß in diesem Falle noch der Data-Request-Ausgang als Bit 7 an einen Eingang gelegt werden, um hier mit einem BIT-Befehl abfragbar zu werden. Darüber hinaus sind nahezu keine Grenzen gesetzt, was Geschwindigkeit und Komfort betrifft, wenn man mit höheren Prozessorgeschwindigkeiten oder DMA arbeitet, beides Techniken, die mit dem AIM Probleme bereiten.

Ziel des Verfassers war es, dem Leser eine ausgeführte Schaltung vorzustellen, die einerseits einfach und kostengünstig ist, die andererseits aber auch hinreichend störsicher und komfortabel arbeitet. Die vorgestellte Schaltung stellt dabei einen guten Kompromiß dar, der durch die später noch zu beschreibende Software unterstützt wird.

(Wird fortgesetzt)

Dipl.-Ing. (FH) Rudolf Kirchgäßner, 6831 Brühl

MCROSSREF

Referenzliste für 65xx-Programme

MCROSSREF listet alle Programmadressen, die auf Felder (Speicheradressen) verweisen. Gleichzeitig wird mit einem *-Symbol vermerkt, ob Speicher-Inhaltsänderungen (Wertzuweisungen) z.B. durch STA oder INC vorgenommen werden. Eine weitere Liste zeigt eine Zusammenstellung der Sprungziele. Das sind Speicheradressen, die im Programmverlauf durch Branch- oder JMP-Befehle angesprungen werden. Aufrufe durch JSR werden ebenfalls mit einem Stern (*) gekennzeichnet. Die indirekte Verwendung von Adressen wird durch Klammern symbolisiert, wogegen ein nachfolgendes Komma indizierte Adressierung bedeutet.

Funktion

Von Jim Butterfield stammt eine Tabelle, in der 6502-Opcodes nach logischen Gesichtspunkten geordnet sind: Alle Befehle werden nach Adressierungsart, Sprungarten usw. dargestellt. Diese Tabelle ist Ausgangspunkt zur Untersuchung der einzelnen Opcodes. Über AND sowie EOR-Befehle wird die Gruppenzugehörigkeit eines Opcodes ermittelt und dadurch Befehlstyp und -länge festgelegt. In eine Verweistabelle werden eingetragen das Sprungziel bzw. die Feldadresse, ein Code für Befehls- und Adressierungsart sowie der Adreßpegel, je Element also 5 Bytes.

Eine eigene Behandlung erfahren die relativen Sprünge: Ihr Sprungziel muß besonders errechnet werden. BIT-Befehle werden - wie auch in diesem Programm - häufig benutzt, um alternative Werte in ein Register zu laden. Damit dies nicht zu verfälschenden Ergebnissen führt, wird abgefragt, ob der auf BIT folgende Befehl ein relativer Sprung ist. Nur dann wird der Operand als Speicheradresse bewertet. Nach der Sortierung werden die Ergebnisse getrennt nach Sprungzielen und Feldern ausgedruckt.

Bedienung

Das vorliegende Programm wurde auf dem CBM 8032 entwickelt, die Systemadressen für BASIC 3 sind angegeben, so daß eine Umstellung ohne Schwierigkeiten möglich ist.

Zunächst sollte das zu untersuchende Maschinenprogramm disassembliert werden, um Tabellen, Texte usw. zu finden. Nur was sinnvollen Mnemonic-Code ergibt, kann auch von MCROSSREF sinnvoll verarbeitet werden. Nach dem Programmstart wird nach freiem Speicherplatz für die Verweistabelle gefragt. Weiter muß die Anzahl der Programmbausteine (Module) angegeben werden. Da möglicherweise das Untersuchungsobjekt im Speicher an eine günstige Stelle verschoben werden mußte, kann die reale von der logischen Startadresse abweichen. MCROSSREF berücksichtigt auch dies. Die Darstellungsform der Modulgrenzen entspricht dem Programm UNASS. Alle Eingaben werden im Hex-Format erwartet, eine Überprüfung auf Gültigkeit erfolgt nicht.

Literatur:
Warnecke,H.: MCC Microtips, Ketsch, Dez. 82
Quint, A.: Sortierprogramm, 65xx MICRO MAG, Ahrensburg, Dez. 80
Kirchgäßner, R.: UNASS, 65xx MICRO MAG, Ahrensburg, Dez. 82

```
                        0010 ;PU "$0:MCROSS.ASM"
                        0020
                        0030 ;          R.KIRCHGAESSNER          (C) 06.07.83
                        0040 ;          NIBELUNGENSTR.4
                        0050 ;          6831 BRUEHL
                        0060
                        0070 ;          TEL:(06202) 74647
                        0080
                        0090                .BA $7B8E            ;SYS 31630
                        0100                .OS
                        0110                .CE
                        0120
                        0130 MCZHMODUL  .DE $00                  ;ZAHL DER MODULE
                        0140 MCZVERWEIS .DE $00                  ;VERWEISE IN 1 ZEILE
                        0150 MCTABPT    .DE $01                  ;POINTER
                        0160 MCZMODUL   .DE $0C                  ;ZAEHLER FUER MODULE
                        0170 MCZPASS    .DE $0F                  ;ZAEHLER FUER PASS
                        0180 MCMEMANF   .DE $18                  ;BEGINN FREIER SPEICHER
                        0190 MCREALPT   .DE $5E                  ;*
                        0200 MCADRKORR  .DE $60                  ;*  6 BYTE
                        0210 MCMODULEND .DE $62                  ;*
                        0220 MCMODPT    .DE $64                  ;POINTER AUF MODULWERTE
                        0230 MCSORTANF  .DE $67                  ;POINTER AUF TABANF
                        0240 MCVERGL    .DE $67                  ;3 BYTE
                        0250 MCOPCODE   .DE $67                  ;OPCODE UND
                        0260 MCBYTE2    .DE $68                  ;MOEGLICHE
                        0270 MCBYTE3    .DE $69                  ;OPERANDEN
                        0280 MCADRART   .DE $B6                  ;ADRESSIERUNGSART
                        0290 MCMZFELD   .DE $B6                  ;MEHRZWECKFELD
                        0300 MCTABANF   .DE $FB                  ;TABELLEN-ANFANG
                        0310 MCLAENGE   .DE $FD                  ;BEFEHL / ELEMENT
                        0320
                        0330 ;    SYSTEM-ADRESSEN
                        0340 ;    ---------------
                        0350
                        0360 DFLTO      .DE $B0                  ;BASIC 3
                        0370 DN         .DE $D4                  ;-------
                        0380 READY      .DE $B3FF                ;$C389
                        0390 SPAC2      .DE $D52E                ;$FDCA
                        0400 SPACE      .DE $D531                ;$FDCD
                        0410 CRLF       .DE $D534                ;$FDD0
                        0420 WROB       .DE $D722                ;$E775
                        0430 RDOA       .DE $D754                ;$E7A7
                        0440 RDOB       .DE $D763                ;$E7B6
                        0450 RDOC       .DE $D798                ;$E7EB
                        0460 LISTN      .DE $FOD5                ;$FOBA
                        0470 SCATN      .DE $F148                ;$F12D
                        0480 STOPEQ     .DE $F335                ;$F301
                        0490 CLRCH      .DE $FFCC
                        0500 BASIN      .DE $FFCF
                        0510 BSOUT      .DE $FFD2
                        0520
7B8E- 20 E6 7B          0530 MCBEGINN   JSR MCPARAM              ;PARAMETER HOLEN
7B91- A9 FF             0540            LDA #$FF
7B93- 85 0F             0550            STA *MCZPASS
7B95- 20 A7 7B          0560 MCWORK     JSR MCPASS               ;2 DURCHLAEUFE
7B98- E6 0F             0570            INC *MCZPASS
7B9A- F0 F9             0580            BEQ MCWORK
7B9C- A2 C6             0590            LDX #MCMSGENDE-MCMSGTAB
7B9E- 20 56 7C          0600            JSR MCPRCRTEXT           ;ENDE-MELDUNG
7BA1- 20 CC FF          0610 MCCLRCH    JSR CLRCH                ;STANDARD I/O SETZEN
7BA4- 4C FF B3          0620            JMP READY                ;ZURUECK ZU BASIC
                        0630
7BA7- A5 0C             0640 MCPASS     LDA *MCZMODUL
7BA9- 85 00             0650            STA *MCZHMODUL
```

```
7BAB- 18        0660            CLC
7BAC- A5 18     0670            LDA *MCMEMANF           ;TABELLENANFANG
7BAE- 85 64     0680            STA *MCMODPT
7BB0- 65 B6     0690            ADC *MCMZFELD           ;UND - POINTER
7BB2- 85 FB     0700            STA *MCTABANF           ;BERECHNEN
7BB4- 85 01     0710            STA *MCTABPT
7BB6- A5 19     0720            LDA *MCMEMANF+1
7BB8- 85 65     0730            STA *MCMODPT+1
7BBA- 69 00     0740            ADC #0
7BBC- 85 FC     0750            STA *MCTABANF+1
7BBE- 85 02     0760            STA *MCTABPT+1
7BC0- 20 68 7C  0770 MCNEXTMOD  JSR MCREADBER           ;MODULWERTE HOLEN
7BC3- 20 8D 7C  0780 MCNEXTBEF  JSR MCDECODE            ;BYTE DISASSEMBLIEREN
7BC6- 20 6E 7D  0790            JSR MCADRPLUS           ;WEITERES BYTE ?
7BC9- B0 F8     0800            BCS MCNEXTBEF           ;JA
7BCB- 18        0810            CLC
7BCC- A5 64     0820            LDA *MCMODPT
7BCE- 69 06     0830            ADC #6
7BD0- 85 64     0840            STA *MCMODPT
7BD2- A5 65     0850            LDA *MCMODPT+1
7BD4- 69 00     0860            ADC #0
7BD6- 85 65     0870            STA *MCMODPT+1
7BD8- C6 00     0880            DEC *MCZHMODUL
7BDA- D0 E4     0890            BNE MCNEXTMOD           ;WEITERES MODUL
7BDC- A9 05     0900            LDA #5                  ;ELEMENTLAENGE
7BDE- 85 FD     0910            STA *MCLAENGE
7BE0- 20 8D 7D  0920            JSR MCSORT              ;VERWEISE SORTIEREN
7BE3- 4C 12 7E  0930            JMP MCAUSGABE           ;FELDER DRUCKEN
                0940
7BE6- A2 00     0950 MCPARAM    LDX #0
7BE8- 20 5C 7C  0960            JSR MCPRTEXT
7BEB- 20 54 D7  0970            JSR RDOA                ;FREIER SPEICHER AB ...
7BEE- A5 FB     0980            LDA *MCTABANF
7BF0- 85 18     0990            STA *MCMEMANF
7BF2- A5 FC     1000            LDA *MCTABANF+1
7BF4- 85 19     1010            STA *MCMEMANF+1
7BF6- A2 30     1020            LDX #MCMSGMODUL-MCMSGTAB
7BF8- 20 56 7C  1030            JSR MCPRCRTEXT          ;ANZAHL MODULE ?
7BFB- 20 63 D7  1040            JSR RDOB                ;LIEST 1 HEX-BYTE
7BFE- 85 0C     1050            STA *MCZMODUL
7C00- 85 00     1060            STA *MCZHMODUL
7C02- A2 45     1070            LDX #MCMSGGRENZ-MCMSGTAB
7C04- 20 56 7C  1080            JSR MCPRCRTEXT          ;MODULGRENZEN
7C07- A0 00     1090            LDY #0                  ;INDEX FUER MCMODPT
7C09- 20 3F 7C  1100 MCPARAM1   JSR MCGET3ADR
7C0C- 20 34 D5  1110            JSR CRLF
7C0F- C6 00     1120            DEC *MCZHMODUL
7C11- D0 F6     1130            BNE MCPARAM1            ;WEITERES MODUL
7C13- 84 B6     1140            STY *MCMZFELD           ;INDEX VON MCMEMANF
7C15- A2 86     1150            LDX #MCMSGPRINT-MCMSGTAB
7C17- 20 56 7C  1160            JSR MCPRCRTEXT          ;AUSDRUCK ?
7C1A- 20 98 D7  1170            JSR RDOC
7C1D- C9 4A     1180            CMP #'J
7C1F- D0 1D     1190            BNE MCPARAM3
7C21- A2 95     1200            LDX #MCMSGNAME-MCMSGTAB         ;UEBERSCHRIFT
7C23- 20 56 7C  1210            JSR MCPRCRTEXT
7C26- A9 04     1220            LDA #4                  ;OPEN FUER DRUCKER
7C28- 85 B0     1230            STA *DFLTO              ;AUSGABE-KANAL
7C2A- 85 D4     1240            STA *DN                 ;DEVICE NUMBER
7C2C- 20 D5 F0  1250            JSR LISTN               ;LISTEN-KOMMANDO
7C2F- 20 48 F1  1260            JSR SCATN               ;BEENDET ATTENTION
7C32- 20 CF FF  1270 MCPARAM2   JSR BASIN               ;PROGRAMM-NAMEN HOLEN
7C35- C9 0D     1280            CMP #$0D
7C37- F0 05     1290            BEQ MCPARAM3
7C39- 20 D2 FF  1300            JSR BSOUT               ;UND DRUCKEN
7C3C- D0 F4     1310            BNE MCPARAM2
7C3E- 60        1320 MCPARAM3   RTS
                1330
7C3F- 20 48 7C  1340 MCGET3ADR  JSR MCGETADR
7C42- 20 45 7C  1350            JSR MCGET1ADR
```

```
7C45- 20 98 D7  1360 MCGET1ADR  JSR RDOC          ;1 ZEICHEN UEBERLESEN
7C48- 20 54 D7  1370 MCGETADR   JSR RDOA          ;1 HEX-ADRESSE HOLEN
7C4B- A5 FB     1380            LDA *MCTABANF
7C4D- 91 18     1390            STA (MCMEMANF),Y
7C4F- C8        1400            INY
7C50- A5 FC     1410            LDA *MCTABANF+1
7C52- 91 18     1420            STA (MCMEMANF),Y
7C54- C8        1430            INY
7C55- 60        1440            RTS
                1450
7C56- 20 34 D5  1460 MCPRCRTEXT JSR CRLF          ;NEUE ZEILEN
7C59- 20 34 D5  1470            JSR CRLF
7C5C- BD 22 7F  1480 MCPRTEXT   LDA MCMSGTAB,X
7C5F- F0 06     1490            BEQ MCPRRTS
7C61- 20 D2 FF  1500            JSR BSOUT         ;MELDUNG AUSGEBEN
7C64- E8        1510            INX
7C65- D0 F5     1520            BNE MCPRTEXT
7C67- 60        1530 MCPRRTS    RTS
                1540
7C68- A0 05     1550 MCREADBER  LDY #5            ;NEUE MODULWERTE HOLEN
7C6A- B1 64     1560 MCREADBER1 LDA (MCMODPT),Y
7C6C- 99 5E 00  1570            STA MCREALPT,Y
7C6F- 88        1580            DEY
7C70- 10 F8     1590            BPL MCREADBER1
7C72- 38        1600            SEC
7C73- A5 60     1610            LDA *MCADRKORR    ;KORREKTURWERT, D.H.
7C75- E5 5E     1620            SBC *MCREALPT     ;VERSCHIEBEWERT
7C77- 85 60     1630            STA *MCADRKORR    ;BERECHNEN
7C79- A5 61     1640            LDA *MCADRKORR+1
7C7B- E5 5F     1650            SBC *MCREALPT+1
7C7D- 85 61     1660            STA *MCADRKORR+1
7C7F- 38        1670            SEC
7C80- A5 62     1680            LDA *MCMODULEND   ;REALE MODULGRENZE
7C82- E5 60     1690            SBC *MCADRKORR    ;BERECHNEN
7C84- 85 62     1700            STA *MCMODULEND
7C86- A5 63     1710            LDA *MCMODULEND+1
7C88- E5 61     1720            SBC *MCADRKORR+1
7C8A- 85 63     1730            STA *MCMODULEND+1
7C8C- 60        1740            RTS
                1750
7C8D- A0 00     1760 MCDECODE   LDY #0
7C8F- B1 5E     1770            LDA (MCREALPT),Y  ;OPCODE
7C91- AA        1780            TAX               ;KOPIEREN
7C92- C8        1790            INY
7C93- B1 5E     1800            LDA (MCREALPT),Y  ;MOEGLICHE OPERANDEN
7C95- 85 68     1810            STA *MCBYTE2
7C97- C8        1820            INY
7C98- B1 5E     1830            LDA (MCREALPT),Y
7C9A- 85 69     1840            STA *MCBYTE3      ;MERKEN
7C9C- E0 2C     1850            CPX #$2C          ;IST OPCODE ..
7C9E- F0 43     1860            BEQ MCDECODE2     ;.. BIT ABSOLUT
7CA0- E0 24     1870            CPX #$24
7CA2- F0 43     1880            BEQ MCDECODE3     ;.. BIT ZERO
7CA4- 8A        1890            TXA
7CA5- 29 1F     1900            AND #$1F          ;RELATIVSPRUNG MASKIEREN
7CA7- C9 10     1910            CMP #$10
7CA9- D0 49     1920            BNE MCDECODE5     ;KEIN RELATIVSPRUNG
7CAB- 18        1930            CLC
7CAC- A5 68     1940            LDA *MCBYTE2      ;SPRUNGZIEL BERECHNEN
7CAE- 48        1950            PHA               ;OPERAND MERKEN
7CAF- 65 5E     1960            ADC *MCREALPT     ;SPRUNGZIEL BERECHNEN
7CB1- 85 68     1970            STA *MCBYTE2
7CB3- A5 5F     1980            LDA *MCREALPT+1
7CB5- 69 00     1990            ADC #0
7CB7- 85 69     2000            STA *MCBYTE3
7CB9- 18        2010            CLC
7CBA- A5 68     2020            LDA *MCBYTE2      ;+ SPRUNGKONSTANTE
7CBC- 69 02     2030            ADC #2
7CBE- 85 68     2040            STA *MCBYTE2
7CC0- A5 69     2050            LDA *MCBYTE3
```

```
7CC2- 69 00      2060              ADC #0
7CC4- 85 69      2070              STA *MCBYTE3
7CC6- 18         2080              CLC
7CC7- A5 68      2090              LDA *MCBYTE2          ;UND KORRIGIEREN
7CC9- 65 60      2100              ADC *MCADRKORR
7CCB- 85 68      2110              STA *MCBYTE2
7CCD- A5 69      2120              LDA *MCBYTE3
7CCF- 65 61      2130              ADC *MCADRKORR+1
7CD1- 85 69      2140              STA *MCBYTE3
7CD3- 68         2150              PLA                   ;URSPRUNGS-OPERAND
7CD4- 10 02      2160              BPL MCDECODE1         ;VORWAERTSSPRUNG
7CD6- C6 69      2170              DEC *MCBYTE3
7CD8- A9 02      2180 MCDECODE1    LDA #2                ;BEFEHLSLAENGE
7CDA- 85 FD      2190              STA *MCLAENGE
7CDC- A9 18      2200              LDA #$18              ;UND -TYP
7CDE- 85 B6      2210              STA *MCADRART
7CE0- 4C 2D 7D   2220              JMP MCSTORE
                 2230
7CE3- C8         2240 MCDECODE2    INY                   ;=3
7CE4- B1 5E      2250              LDA (MCREALPT),Y      ;NAECHSTEN OPCODE ...
7CE6- 2C         2260              .BY $2C
7CE7- A5 69      2270 MCDECODE3    LDA *MCBYTE3          ;... LADEN
7CE9- 29 1F      2280              AND #$1F              ;UND TESTEN, OB
7CEB- C9 10      2290              CMP #$10              ;RELATIVSSPRUNG
7CED- F0 05      2300              BEQ MCDECODE5         ;ALSO "ECHTER" BIT
7CEF- A9 01      2310              LDA #1                ;SONST BIT-BEFEHL ..
7CF1- 85 FD      2320              STA *MCLAENGE         ;..UEBERGEHEN
7CF3- 60         2330 MCDECODE4    RTS
                 2340
7CF4- 8A         2350 MCDECODE5    TXA                   ;MCOPCODE
7CF5- A8         2360              TAY
7CF6- A2 0C      2370              LDX #12
7CF8- 98         2380 MCDECODE6    TYA                   ;BEFEHLSGRUPPE SUCHEN
7CF9- 3D EE 7E   2390              AND MCAND,X
7CFC- 5D FB 7E   2400              EOR MCEOR,X
7CFF- F0 03      2410              BEQ MCDECODE7         ;GEFUNDEN
7D01- CA         2420              DEX
7D02- D0 F4      2430              BNE MCDECODE6
                 2440
7D04- BD 08 7F   2450 MCDECODE7    LDA MCLEN,X           ;BEFEHLSLAENGE
7D07- 85 FD      2460              STA *MCLAENGE
7D09- BD 15 7F   2470              LDA MCTYP,X           ;UND -TYP
7D0C- F0 E5      2480              BEQ MCDECODE4         ;1 BYTE + IMMEDIATE
7D0E- 85 B6      2490              STA *MCADRART
7D10- 10 1B      2500              BPL MCSTORE           ;SPRUNGBEFEHL
7D12- 98         2510              TYA
7D13- 29 E0      2520              AND #$E0              ;TESTEN, OB WERTZUWEISUNG
7D15- C9 A0      2530              CMP #$A0
7D17- F0 0E      2540              BEQ MCDECODE8         ;ALLE A. UND B. SIND LOAD
7D19- 98         2550              TYA
7D1A- 29 E0      2560              AND #$E0
7D1C- C9 80      2570              CMP #$80
7D1E- F0 0D      2580              BEQ MCSTORE           ;ALLE 8. UND 9. SIND STORE
7D20- 98         2590              TYA
7D21- 29 03      2600              AND #$03
7D23- C9 02      2610              CMP #$02
7D25- F0 06      2620              BEQ MCSTORE           ;.2,.6 UND .E AENDERN FELD
7D27- A5 B6      2630 MCDECODE8    LDA *MCADRART
7D29- 09 08      2640              ORA #$08              ;KEINE WERTZUWEISUNG
7D2B- 85 B6      2650              STA *MCADRART
                 2660
7D2D- A0 02      2670 MCSTORE      LDY #2
7D2F- A5 0F      2680              LDA *MCZPASS
7D31- 10 05      2690              BPL MCSTORE1          ;PASS 2
7D33- A5 B6      2700              LDA *MCADRART         ;PASS 1
7D35- 10 05      2710              BPL MCSTORE2          ;NUR SPRUENGE
7D37- 60         2720              RTS
                 2730
7D38- A5 B6      2740 MCSTORE1     LDA *MCADRART
7D3A- 10 31      2750              BPL MCSTORE4          ;KEINE SPRUENGE
```

```
7D3C- 91 01      2760 MCSTORE2   STA (MCTABPT),Y       ;ADRESSART SPEICHERN
7D3E- 29 01      2770            AND #$01              ;UND TESTEN
7D40- F0 03      2780            BEQ MCSTORE3          ;ABSOLUT-ADRESSE
7D42- A9 00      2790            LDA #0                ;ZERO-ADRESSE
7D44- 2C         2800            .BY $2C
7D45- A5 69      2810 MCSTORE3   LDA *MCBYTE3
7D47- A0 00      2820            LDY #0
7D49- 91 01      2830            STA (MCTABPT),Y       ;HIGH-BYTE
7D4B- C8         2840            INY
7D4C- A5 68      2850            LDA *MCBYTE2
7D4E- 91 01      2860            STA (MCTABPT),Y       ;LOW-BYTE
7D50- A0 04      2870            LDY #4
7D52- 18         2880            CLC
7D53- A5 5E      2890            LDA *MCREALPT         ;ADRESSPEGEL
7D55- 65 60      2900            ADC *MCADRKORR
7D57- 91 01      2910            STA (MCTABPT),Y
7D59- 88         2920            DEY                   ;=3
7D5A- A5 5F      2930            LDA *MCREALPT+1
7D5C- 65 61      2940            ADC *MCADRKORR+1
7D5E- 91 01      2950            STA (MCTABPT),Y
7D60- 18         2960            CLC
7D61- A5 01      2970            LDA *MCTABPT          ;NAECHSTES TAB-ELEMENT
7D63- 69 05      2980            ADC #5
7D65- 85 01      2990            STA *MCTABPT
7D67- A5 02      3000            LDA *MCTABPT+1
7D69- 69 00      3010            ADC #0
7D6B- 85 02      3020            STA *MCTABPT+1
7D6D- 60         3030 MCSTORE4   RTS
                 3040
7D6E- 18         3050 MCADRPLUS  CLC                   ;NAECHSTER BEFEHL BZW.
7D6F- A5 5E      3060            LDA *MCREALPT
7D71- 65 FD      3070            ADC *MCLAENGE         ;TABELLEN-ELEMENT
7D73- 85 5E      3080            STA *MCREALPT
7D75- A5 5F      3090            LDA *MCREALPT+1
7D77- 69 00      3100            ADC #0
7D79- 85 5F      3110            STA *MCREALPT+1
7D7B- 20 35 F3   3120 MCIMBER    JSR STOPEQ            ;STOP-TASTE GEDRUECKT
7D7E- D0 03      3130            BNE MCIMBER1
7D80- 4C A1 7B   3140            JMP MCCLRCH           ;ABBRUCH
7D83- 38         3150 MCIMBER1   SEC
7D84- A5 62      3160            LDA *MCMODULEND       ;WENN ENDE ERREICHT,
7D86- E5 5E      3170            SBC *MCREALPT
7D88- A5 63      3180            LDA *MCMODULEND+1
7D8A- E5 5F      3190            SBC *MCREALPT+1       ;.. WIRD CARRY = 0
7D8C- 60         3200            RTS
                 3210
                 3220 ;TABELLENSORT - AEHNLICH MICRO MAG HEFT 16
                 3230 ;AUSWAHLVERFAHREN MIT AUSTAUSCH:
                 3240 ;BEI JEDEM DURCHLAUF WIRD DAS KLEINSTE
                 3250 ;ELEMENT AN DEN TABELLENANFANG GEBRACHT
                 3260
7D8D- 38         3270 MCSORT     SEC
7D8E- A5 01      3280            LDA *MCTABPT          ;ENDADRESSE
7D90- E5 FD      3290            SBC *MCLAENGE
7D92- 85 62      3300            STA *MCMODULEND
7D94- A5 02      3310            LDA *MCTABPT+1
7D96- E9 00      3320            SBC #0
7D98- 85 63      3330            STA *MCMODULEND+1
7D9A- A5 FB      3340            LDA *MCTABANF
7D9C- 85 67      3350            STA *MCSORTANF
7D9E- A5 FC      3360            LDA *MCTABANF+1
7DA0- 85 68      3370            STA *MCSORTANF+1
7DA2- A5 67      3380 MCSORT1    LDA *MCSORTANF        ;NEUER DURCHLAUF
7DA4- 85 01      3390            STA *MCTABPT
7DA6- A5 68      3400            LDA *MCSORTANF+1
7DA8- 85 02      3410            STA *MCTABPT+1
7DAA- 18         3420            CLC
7DAB- A5 01      3430            LDA *MCTABPT
7DAD- 65 FD      3440            ADC *MCLAENGE
7DAF- 85 5E      3450            STA *MCREALPT
```

```
7DB1- A5 02     3460             LDA #MCTABPT+1
7DB3- 69 00     3470             ADC #0
7DB5- 85 5F     3480             STA #MCREALPT+1
7DB7- A0 00     3490 MCSORT2     LDY #0              ;ELEMENTE VERGLEICHEN
7DB9- B1 5E     3500 MCSORT3     LDA (MCREALPT),Y
7DBB- D1 01     3510             CMP (MCTABPT),Y
7DBD- 90 09     3520             BCC MCSORT4         ;KLEINER
7DBF- D0 0F     3530             BNE MCSORT5
                3540
7DC1- C8        3550             INY
7DC2- C4 FD     3560             CPY #MCLAENGE
7DC4- 90 F3     3570             BCC MCSORT3         ;WEITER VERGLEICHEN
7DC6- B0 08     3580             BCS MCSORT5
                3590
7DC8- A5 5E     3600 MCSORT4     LDA #MCREALPT       ;POINTER UEBERNEHMEN
7DCA- 85 01     3610             STA #MCTABPT
7DCC- A5 5F     3620             LDA #MCREALPT+1
7DCE- 85 02     3630             STA #MCTABPT+1
7DD0- 18        3640 MCSORT5     CLC
7DD1- A5 5E     3650             LDA #MCREALPT
7DD3- 65 FD     3660             ADC #MCLAENGE
7DD5- 85 5E     3670             STA #MCREALPT
7DD7- A5 5F     3680             LDA #MCREALPT+1
7DD9- 69 00     3690             ADC #0
7DDB- 85 5F     3700             STA #MCREALPT+1
7DDD- C5 63     3710             CMP #MCMODULEND+1
7DDF- 90 D6     3720             BCC MCSORT2
7DE1- A5 62     3730             LDA #MCMODULEND
7DE3- C5 5E     3740             CMP #MCREALPT
7DE5- B0 D0     3750             BCS MCSORT2
7DE7- 20 02 7E  3760             JSR MCTAUSCH        ;TABENDE ERREICHT
7DEA- 18        3770             CLC
7DEB- A5 67     3780             LDA #MCSORTANF
7DED- 65 FD     3790             ADC #MCLAENGE
7DEF- 85 67     3800             STA #MCSORTANF
7DF1- A5 68     3810             LDA #MCSORTANF+1
7DF3- 69 00     3820             ADC #0
7DF5- 85 68     3830             STA #MCSORTANF+1
7DF7- C5 63     3840             CMP #MCMODULEND+1
7DF9- 90 A7     3850             BCC MCSORT1
7DFB- A5 67     3860             LDA #MCSORTANF
7DFD- C5 62     3870             CMP #MCMODULEND
7DFF- 90 A1     3880             BCC MCSORT1
7E01- 60        3890             RTS
                3900
7E02- A0 04     3910 MCTAUSCH    LDY #4              ;KLEINSTES ELEMENT
7E04- B1 01     3920 MCTAUSCH1   LDA (MCTABPT),Y      ;TAUSCHEN
7E06- 48        3930             PHA                 ;SPEICHERN UEBER STACK
7E07- B1 67     3940             LDA (MCSORTANF),Y
7E09- 91 01     3950             STA (MCTABPT),Y
7E0B- 68        3960             PLA
7E0C- 91 67     3970             STA (MCSORTANF),Y
7E0E- 88        3980             DEY
7E0F- 10 F3     3990             BPL MCTAUSCH1
7E11- 60        4000             RTS
                4010
7E12- A2 AD     4020 MCAUSGABE   LDX #MCMSGZIELE-MCMSGTAB
7E14- A5 0F     4030             LDA #MCZPASS
7E16- D0 02     4040             BNE MCAUSGABE1
7E18- A2 BC     4050             LDX #MCMSGFELD-MCMSGTAB
7E1A- 20 56 7C  4060 MCAUSGABE1  JSR MCPRCRTEXT
7E1D- A5 FB     4070             LDA #MCTABANF       ;ANFANGSWERT SETZEN
7E1F- 85 5E     4080             STA #MCREALPT
7E21- A5 FC     4090             LDA #MCTABANF+1
7E23- 85 5F     4100             STA #MCREALPT+1
7E25- A0 FF     4110             LDY #$FF            ;VERGLEICHSFELD ENTWERTEN
7E27- 84 67     4120             STY #MCVERGL
7E29- 84 68     4130             STY #MCVERGL+1
7E2B- E6 62     4140             INC #MCMODULEND     ;ENDADRESSE + 1
7E2D- D0 02     4150             BNE MCAUSGABE2
7E2F- E6 63     4160             INC #MCMODULEND+1
```

```
7E31- 20 3A 7E  4170 MCAUSGABE2 JSR MCPRELEM        ;ELEMENT AUSGEBEN
7E34- 20 6E 7D  4180            JSR MCADRPLUS       ;TABELLENENDE ?
7E37- B0 F8     4190            BCS MCAUSGABE2      ;NEIN, NAECHSTES ELEMENT
7E39- 60        4200            RTS
                4210
7E3A- A0 00     4220 MCPRELEM   LDY #0              ;GLEICHE ADRESSE ?
7E3C- B1 5E     4230            LDA (MCREALPT),Y
7E3E- C5 67     4240            CMP *MCVERGL
7E40- D0 10     4250            BNE MCPRELEM1
7E42- C8        4260            INY                 ;=1
7E43- B1 5E     4270            LDA (MCREALPT),Y
7E45- C5 68     4280            CMP *MCVERGL+1
7E47- D0 09     4290            BNE MCPRELEM1
7E49- C8        4300            INY                 ;=2
7E4A- B1 5E     4310            LDA (MCREALPT),Y    ;GLEICHE ADRESS-ART ?
7E4C- 29 70     4320            AND #$70
7E4E- C5 69     4330            CMP *MCVERGL+2
7E50- F0 5D     4340            BEQ MCPRELEM7       ;JA
7E52- A0 00     4350 MCPRELEM1  LDY #0              ;NEUE WERTE MERKEN
7E54- B1 5E     4360            LDA (MCREALPT),Y
7E56- 85 67     4370            STA *MCVERGL
7E58- C8        4380            INY                 ;=1
7E59- B1 5E     4390            LDA (MCREALPT),Y
7E5B- 85 68     4400            STA *MCVERGL+1
7E5D- C8        4410            INY
7E5E- B1 5E     4420            LDA (MCREALPT),Y
7E60- 29 70     4430            AND #$70            ;NUR ADRESS-ART MERKEN
7E62- 85 69     4440            STA *MCVERGL+2
7E64- 20 34 D5  4450            JSR CRLF            ;ZEILENWECHSEL
7E67- A0 FF     4460            LDY #$FF
7E69- 84 00     4470            STY *MCZVERWEIS
7E6B- A0 02     4480            LDY #2
7E6D- B1 5E     4490            LDA (MCREALPT),Y
7E6F- 29 40     4500            AND #$40            ;INDIREKTE ADRESSIERUNG
7E71- F0 03     4510            BEQ MCPRELEM2       ;NEIN
7E73- A2 28     4520            LDX #'(
7E75- 2C        4530            .BY $2C
7E76- A2 20     4540 MCPRELEM2  LDX #'
7E78- A0 00     4550            LDY #0
7E7A- B1 5E     4560            LDA (MCREALPT),Y    ;HIGH-BYTE
7E7C- F0 0B     4570            BEQ MCPRELEM3       ;ZERO-ADRESSE
7E7E- A8        4580            TAY
7E7F- 8A        4590            TXA
7E80- 20 D2 FF  4600            JSR BSOUT
7E83- 98        4610            TYA
7E84- 20 22 D7  4620            JSR WROB            ;HIGH-BYTE AUSGEBEN
7E87- D0 07     4630            BNE MCPRELEM4       ;IMMER
                4640
7E89- 20 2E D5  4650 MCPRELEM3  JSR SPAC2           ;ZERO-ADRESSE
7E8C- 8A        4660            TXA
7E8D- 20 D2 FF  4670            JSR BSOUT           ;KLAMMER BZW. SPACE
7E90- A0 01     4680 MCPRELEM4  LDY #1
7E92- B1 5E     4690            LDA (MCREALPT),Y    ;LOW-BYTE
7E94- 20 22 D7  4700            JSR WROB
7E97- C8        4710            INY                 ;=2
7E98- B1 5E     4720            LDA (MCREALPT),Y
7E9A- 29 70     4730            AND #$70            ;ADRESS-ARTEN MASKIEREN
7E9C- C9 10     4740            CMP #$10            ;ABSOLUT ?
7E9E- F0 0A     4750            BEQ MCPRELEM6
7EA0- C9 20     4760            CMP #$20            ;INDEXED ?
7EA2- F0 03     4770            BEQ MCPRELEM5
7EA4- A9 29     4780            LDA #')             ;ALSO INDIREKT
7EA6- 2C        4790            .BY $2C
7EA7- A9 2C     4800 MCPRELEM5  LDA #',
7EA9- 2C        4810            .BY $2C
7EAA- A9 20     4820 MCPRELEM6  LDA #'
7EAC- 20 D2 FF  4830            JSR BSOUT
7EAF- 20 CE 7E  4840 MCPRELEM7  JSR MCPRFORMAT      ;DRUCK-FORMATIERUNG
7EB2- A0 02     4850            LDY #2
```

```
7EB4- B1 5E      4860              LDA (MCREALPT),Y
7EB6- 29 0B      4870              AND #$0B              ;WERTZUWEISUNG BZW. JSR
7EB8- F0 03      4880              BEQ MCPRELEMB
7EBA- A9 20      4890              LDA #'
7EBC- 2C         4900              .BY $2C
7EBD- A9 2A      4910 MCPRELEMB    LDA #'*
7EBF- 20 D2 FF   4920              JSR BSOUT
7EC2- C8         4930              INY                   ;=3
7EC3- B1 5E      4940              LDA (MCREALPT),Y      ;ADRESS-VERWEIS
7EC5- 20 22 D7   4950              JSR WROB              ;AUSGEBEN
7EC8- C8         4960              INY
7EC9- B1 5E      4970              LDA (MCREALPT),Y
7ECB- 4C 22 D7   4980              JMP WROB              ;UND RUECKSPRUNG
                 4990
7ECE- E6 00      5000 MCPRFORMAT   INC *MCZVERWEIS
7ED0- F0 13      5010              BEQ MCPRSPACE         ;NEUE ADRESSE
7ED2- A5 00      5020              LDA *MCZVERWEIS
7ED4- C9 0A      5030              CMP #10               ;BEREITS 10 VERWEISE ?
7ED6- F0 03      5040              BEQ MCNEWLINE
7ED8- 4C 2E D5   5050              JMP SPAC2             ;2 SPACES + RTS
                 5060
7EDB- 20 34 D5   5070 MCNEWLINE    JSR CRLF              ;ZEILENWECHSEL
7EDE- A9 00      5080              LDA #0
7EE0- 85 00      5090              STA *MCZVERWEIS
7EE2- A2 0B      5100              LDX #11
7EE4- 2C         5110              .BY $2C
7EE5- A2 05      5120 MCPRSPACE    LDX #5
7EE7- 20 31 D5   5130 MCPRSPACE1   JSR SPACE             ;TABULATOR
7EEA- CA         5140              DEX
7EEB- D0 FA      5150              BNE MCPRSPACE1
7EED- 60         5160              RTS
                 5170
7EEE- 00         5180 MCAND        .BY $00               ;FALSCHER CODE
7EEF- 1F         5190              .BY $1F               ;(I,X)      01 ... E1
7EF0- 18         5200              .BY $18               ;ABS,X  UNGER  / 8..F
                 5210 ;                                   ABS,Y
7EF1- 1F         5220              .BY $1F               ;(I),Y      11 ... F1
7EF2- 1C         5230              .BY $1C               ;ZPAG,Y UNGER  / 4567
7EF3- 1C         5240              .BY $1C               ;ABS    GERADE / CDEF
7EF4- FF         5250              .BY $FF               ;(JMP)          6C
7EF5- FF         5260              .BY $FF               ;JMP            4C
7EF6- 1F         5270              .BY $1F               ;IMM    GERADE / 9
7EF7- 05         5280              .BY $05               ;1 BYTE ALLE   / 028A
7EF8- FF         5290              .BY $FF               ;JSR            20
7EF9- 9D         5300              .BY $9D               ;IMM     BACE  / 02
7EFA- 1C         5310              .BY $1C               ;ZPAGE  GERADE / 4567
                 5320
7EFB- 00 01 18   5330 MCEOR        .BY $00 $01 $18 $11 $14 $0C $6C
7EFE- 11 14 0C
7F01- 6C
7F02- 4C 09 00   5340              .BY $4C $09 $00 $20 $80 $04
7F05- 20 80 04
7F08- 01 02 03   5350 MCLEN        .BY $01 $02 $03 $02 $02 $03 $03
7F0B- 02 02 03
7F0E- 03
7F0F- 03 02 01   5360              .BY $03 $02 $01 $03 $02 $02
7F12- 03 02 02
7F15- 00 C1 A0   5370 MCTYP        .BY $00 $C1 $A0 $C1 $A1 $90 $48
7F18- C1 A1 90
7F1B- 48
7F1C- 18 00 00   5380              .BY $18 $00 $00 $10 $00 $91
7F1F- 10 00 91
                 5390
                 5400 ;DIE WERTE VON MCTYP SETZEN SICH ZUSAMMEN AUS:
                 5410
                 5420 ;      SPRUNG INDIRECT INDEXED ABS WERT/JSR - - ZERO
                 5430 ;BIT:   7       6       5     4     3     2 1  0
                 5440
                 5450
7F22- 93 12 4D   5460 MCMSGTAB     .BY $93 $12 'MCROSSREF  V1.1' $0D $0D
```

```
7F25- 43 52 4F
7F28- 53 53 52
7F2B- 45 46 20
7F2E- 20 56 31
7F31- 2E 31 0D
7F34- 0D
7F35- 46 52 45  5470              .BY 'FREIER SPEICHER AB: '
7F38- 49 45 52
7F3B- 20 53 50
7F3E- 45 49 43
7F41- 48 45 52
7F44- 20 41 42
7F47- 3A 20
7F49- 30 34 30  5480              .BY '0400' $9D $9D $9D $9D $00
7F4C- 30 9D 9D
7F4F- 9D 9D 00
7F52- 41 4E 5A  5490 MCMSGMODUL .BY 'ANZAHL MODULE ? 01' $9D $9D $00
7F55- 41 48 4C
7F58- 20 4D 4F
7F5B- 44 55 4C
7F5E- 45 20 3F
7F61- 20 30 31
7F64- 9D 9D 00

7F67- 20 4D 4F  5500 MCMSGGRENZ .BY ' MODUL-GRENZEN ' $0D
7F6A- 44 55 4C
7F6D- 2D 47 52
7F70- 45 4E 5A
7F73- 45 4E 20
7F76- 0D
7F77- 20 52 45  5510              .BY ' REAL  LOGISCH' $0D
7F7A- 41 4C 20
7F7D- 20 4C 4F
7F80- 47 49 53
7F83- 43 48 0D
7F86- 53 54 41  5520              .BY 'START START END' $0D $0D
7F89- 52 54 20
7F8C- 53 54 41
7F8F- 52 54 20
7F92- 45 4E 44
7F95- 0D 0D
7F97- 2E 2E 2E  5530              .BY '.... .... ....' $91 $0D $00
7F9A- 2E 20 2E
7F9D- 2E 2E 2E
7FA0- 20 2E 2E
7FA3- 2E 2E 91
7FA6- 0D 00
7FA8- 44 52 55  5540 MCMSGPRINT .BY 'DRUCKEN ? J/N' $9D $00
7FAB- 43 4B 45
7FAE- 4E 20 3F
7FB1- 20 4A 2F
7FB4- 4E 9D 00
7FB7- 50 52 4F  5550 MCMSGNAME  .BY 'PROGRAMM-NAME + DATUM:' $0D $00
7FBA- 47 52 41
7FBD- 4D 4D 2D
7FC0- 4E 41 4D
7FC3- 45 20 2B
7FC6- 20 44 41
7FC9- 54 55 4D
7FCC- 3A 0D 00
7FCF- 0D 53 50  5560 MCMSGZIELE .BY $0D 'SPRUNGZIELE:' $0D $00
7FD2- 52 55 4E
7FD5- 47 5A 49
7FD8- 45 4C 45
7FDB- 3A 0D 00
7FDE- 0D 46 45  5570 MCMSGFELD  .BY $0D 'FELDER:' $0D $00
7FE1- 4C 44 45
7FE4- 52 3A 0D
7FE7- 00
7FE8- 45 4E 44  5580 MCMSGENDE  .BY 'ENDE DER LISTE' $0D $0D $00
7FEB- 45 20 44
```

```
7FEE- 45 52 20
7FF1- 4C 49 53
7FF4- 54 45 0D
7FF7- 0D 00
                5590            .EN
END OF MAE PASS!


 ---  LABEL FILE:  ---

BASIN =FFCF             BSOUT =FFD2             CLRCH =FFCC
CRLF =D534              DFLTO =00B0             DN =00D4
LISTN =FOD5             MCADRART =00B6          MCADRKORR =0060
MCADRPLUS =7D6E         MCAND =7EEE             MCAUSGABE =7E12
MCAUSGABE1 =7E1A        MCAUSGABE2 =7E31        MCBEGINN =7B8E
MCBYTE2 =0068           MCBYTE3 =0069           MCCLRCH =7BA1
MCDECODE =7C8D          MCDECODE1 =7CD8         MCDECODE2 =7CE3
MCDECODE3 =7CE7         MCDECODE4 =7CF3         MCDECODE5 =7CF4
MCDECODE6 =7CF8         MCDECODE7 =7D04         MCDECODE8 =7D27
MCEOR =7EFB             MCGET1ADR =7C45         MCGET3ADR =7C3F
MCGETADR =7C48          MCIMBER =7D7B           MCIMBER1 =7D83
MCLAENGE =00FD          MCLEN =7F08             MCMEMANF =0018
MCMODPT =0064           MCMODULEND =0062        MCMSGENDE =7FE8
MCMSGFELD =7FDE         MCMSGGRENZ =7F67        MCMSGMODUL =7F52
MCMSGNAME =7FB7         MCMSGPRINT =7FA8        MCMSGTAB =7F22
MCMSGZIELE =7FCF        MCMZFELD =00B6          MCNEWLINE =7EDB
MCNEXTBEF =7BC3         MCNEXTMOD =7BC0         MCOPCODE =0067
MCPARAM =7BE6           MCPARAM1 =7C09          MCPARAM2 =7C32
MCPARAM3 =7C3E          MCPASS =7BA7            MCPRORTEXT =7C56
MCPRELEM =7E3A          MCPRELEM1 =7E52         MCPRELEM2 =7E76
MCPRELEM3 =7E89         MCPRELEM4 =7E90         MCPRELEM5 =7EA7
MCPRELEM6 =7EAA         MCPRELEM7 =7EAF         MCPRELEM8 =7EBD
MCPRFORMAT =7ECE        MCPRRTS =7C67           MCPRSPACE =7EE5
MCPRSPACE1 =7EE7        MCPRTEXT =7C5C          MCREADBER =7C68
MCREADBER1 =7C6A        MCREALPT =005E          MCSORT =7D8D
MCSORT1 =7DA2           MCSORT2 =7DB7           MCSORT3 =7DB9
MCSORT4 =7DC8           MCSORT5 =7DD0           MCSORTANF =0067
MCSTORE =7D2D           MCSTORE1 =7D38          MCSTORE2 =7D3C
MCSTORE3 =7D45          MCSTORE4 =7D6D          MCTABANF =00FB
MCTABPT =0001           MCTAUSCH =7E02          MCTAUSCH1 =7E04
MCTYP =7F15             MCVERGL =0067           MCWORK =7B95
MCZHMODUL =0000         MCZMODUL =000C          MCZPASS =000F
MCZVERWEIS =0000        RDOA =D754              RDOB =D763
RDOC =D798              READY =B3FF             SCATN =F148
SPAC2 =D52E             SPACE =D531             STOPEQ =F335
WROB =D722
//0000,7FF9,7FF9
```

FELDER: Ein Ausschnitt aus der entstehenden Liste für Felder

```
   00       *7BA9   *7BDB   *7C00   *7C0F   *7E69   *7ECE   *7EE0    7ED2
   01       *7BB4   *7D65   *7DA4   *7DCA    7D61    7D8E    7DAB
  (01)      *7D3C   *7D49   *7D4E   *7D57   *7D5E   *7E09    7DBB    7E04
   02       *7B8E   *7D6B   *7DA8   *7DCE    7D67    7D94    7DB1
   0C       *7BFE    7BA7
```

Ein Ausschnitt aus der entstehenden Liste für Sprungziele

```
7C09         7C11
7C32         7C3C
7C3E         7C1F    7C37
7C3F        *7C09
7C45        *7C42
7C48        *7C3F
7C56        *7B9E   *7BF8   *7C04   *7C17   *7C23   *7E1A
7C5C        *7BE8    7C65
7C67         7C5F
7C68        *7BC0
7C6A         7C70
```

□

Gerhard Fischer, 2000 Hamburg 52

Erste Erfahrungen mit LISA

Nach einigen Wochen praktischer Arbeit mit Apple`s neuestem Produkt wird hier ein erster Erfahrungsbericht gegeben.

Mit der Lisa zielt Apple in den Bürocomputer-Markt. Als Hobby- oder Heimcomputer dürfte die Maschine mit ca. 30.000 DM Anschaffungspreis zu teuer sein. Für einen Büromenschen mit gehobenen Ansprüchen gibt es für diesen Preis allerdings eine äußerst attraktive Komplettausstattung, die auf Basis einer modernen Hardware (mit dem leistungsfähigen MC 68000 als CPU) eine revolutionäre Software umfaßt, die in ihrer Geschlossenheit und Benutzerfreundlichkeit ihresgleichen sucht. Mit der Lisa setzt Apple neue Maßstäbe in dem von den mächtigen Mitwettbewerbern IBM, DEC, Wang, usw. bevölkerten Markt.

Die Maus

Das eigentliche Kennzeichen der Lisa ist die "Maus". Dieses kleine Handgerät, durch eine dünne Schnur mit dem Computer verbunden, ist Lisas "Mensch/Maschine-Schnittstelle". Mit der Rollkugel und der Taste ermöglicht die Maus die "intuitive" Bedienungsführung. Durch Rollen auf der Tischplatte wird der Cursor auf dem Bildschirm geführt, ein Befehlswort oder ein Symbol "angeklickt" und schon hat man Lisa einen Befehl erteilt, ohne daß man die Tastatur berührt hätte oder komplizierte Kommandosprachen beherrschen müßte.

Schneller Einstieg

Die eigene Erfahrung bestätigt Apple`s Anspruch, daß man nach einer halben Stunde in der Lage ist, mit der Lisa zu hantieren. Hilfreich dabei ist ein auf einer Diskette mitgeliefertes Lernprogramm "LisaGuide", welches einen in amüsanter Weise mit der Handhabung der Maus und den Grundbegriffen der Lisa-Bedienung vertraut macht. Nach Absolvierung dieses Grundkurses (was ca. eine halbe Stunde dauert) und nur wenigen Stunden weiteren praktischen Umgangs mit der Lisa erfolgt die Mausbedienung tatsächlich intuitiv. Man ist schnell in der Lage, rasch und ohne weiteres Nachdenken mit der Maus über den Bildschirm zu fahren und auch kleinste Boxen zielsicher anzuklicken.

Lisa spricht Deutsch

LisaGuide gibt es bis jetzt (Ende September) nur in Englisch, während ansonsten eine in Deutschland ausgelieferte Maschine "deutsch" ist, d.h. auf dem Bildschirm und in den Menütexten "Deutsch spricht" und eine deutsche Tastatur besitzt. Leider sind Uhrzeit und Kalenderdaten im amerikanischen Format verblieben. Auch die mitgelieferten Handbücher - in der Grundausstattung 7 solide Ringbücher, davon je eins für die 6 sogenannten Bürosysteme in einem separaten Karton (!) plus einem "Owner`s Guide" - sind noch in Englisch, sollen jedoch in wenigen Tagen durch deutsche Versionen ergänzt werden.

Die Hardware

Die Hardware besteht aus der eigentlichen Lisa mit dem MC 68000, 1 MB RAM-Speicher, 2x 860 KB 5¼-Zoll-Diskettenlaufwerken, dem hochauflösenden Grafikbildschirm (weiß auf schwarz, bzw. grau), 2 seriellen und 3 parallelen Schnittstellen in einer baulichen Einheit, einer abgesetzten Tastatur, die mit Spiralkabel und Klinkenstecker mit dem Gerät verbunden ist, dann einer 5MB Festplatte sowie einem hochauflösenden Matrixdrucker, der hard- und softwaremäßig integriert ist (wahlweise ist auch ein teurerer Typenraddrucker erhältlich - das damit erreichbare bessere Schriftbild wird allerdings durch Verzicht auf die mögliche Gestaltungsvielfalt erkauft).

Das Ganze macht einen soliden und durchdachten Eindruck. Die Lisa zeichnet sich durch extreme Servicefreundlichkeit aus. Alle Einzelteile können ohne Werkzeug erreicht und ausgewechselt werden. Die Handbücher beschreiben detailliert jeden dazu erforderlichen Handgriff, ohne allerdings nähere technische Einzelheiten zu geben, die über Angaben der physi-

schen Dimensionen hinausgehen.

Die separate Festplatte, die üblicherweise *auf* der Lisa plaziert wird und dort sowohl optisch als auch akustisch etwas stört (die Lisa selbst ist geräuschlos, die Wärmeabfuhr erfolgt ausschließlich durch Konvektion) dürfte wohl nur als Zwischenlösung anzusehen sein und irgendwann durch eine größere und schnellere Einbauplatte abgelöst werden.

Das Lisa-System

Das eigentlich neue und zukunftweisende an der Lisa ist das Softwarekonzept, das dahinter steht. "Lisa" steht für "Local Integrated Software Architecture" (so Apple - ob auch eine Dame dieses Namens im Spiel war, ist nicht bekannt). Das Lisa-System besteht aus dem Lisa Operating System (Lisa OS), der Lisa Shell und den darin eingebetteten Anwendungspaketen, den sogenannten Bürosystemen.

Die Schreibtisch-Metapher

Das Ganze ist wie ein Schreibtisch organisiert. Der Bildschirm symbolisiert die Schreibtischplatte, auf dem Schreibtisch liegen Schriftstücke oder Zettel, im Lisa-Deutsch "Dokumente" genannt. So, wie man im Büroalltag für eine bestimmte Arbeit ein Blatt Papier nimmt, um darauf irgendetwas zu notieren oder zu rechnen oder zu zeichnen, so holt man sich bei Lisa ein "Formular" für eines der 6 Bürosysteme, legt es auf die "Platte" (Bildschirm) und fängt an zu arbeiten. Dann muß man unterbrechen, weil etwas noch Eiligeres kommt und man beginnt etwas Neues. So wie auf einem echten Schreibtisch können sich auf dem Bildschirm die Unterlagen auftürmen, bis man schließlich die Übersicht verliert. Bis zu 20 verschiedene "Papiere" können auf der Lisa-Platte liegen (d.h., gleichzeitig geöffnet sein). Die hohe Zahl an möglichen Interrupt-Ebenen beim MC 68000 wird bei Lisa ausschließlich für Multitasking benutzt. Auf die auch mögliche Multiuserfähigkeit wurde bewußt verzichtet, um dem Benutzer ein möglichst komfortables System zu bieten, was ja auch durch den mit 1MB reichlich dimensionierten Speicher zum Ausdruck kommt.

Das Lisa-Konzept zielt auf den Laienanwender - dem "Spezialisten" im Büro - der möglicherweise vom Computer keine Ahnung hat. Dementsprechend sind alle Bedienungselemente so angelegt, daß sie der Schreibtisch-Metapher möglichst vollkommen entsprechen. Dazu gehört natürlich, daß man das System durch keine Fehlbedienung zum Absturz bringen kann, so wie es ja auch im wirklichen Leben kaum möglich ist, seinen Schreibtisch zu Fall zu bringen. Dies ist den Apple-Konstrukteuren anscheinend gut gelungen.

Der Lisa-Bildschirm - die "Arbeitsplatte"

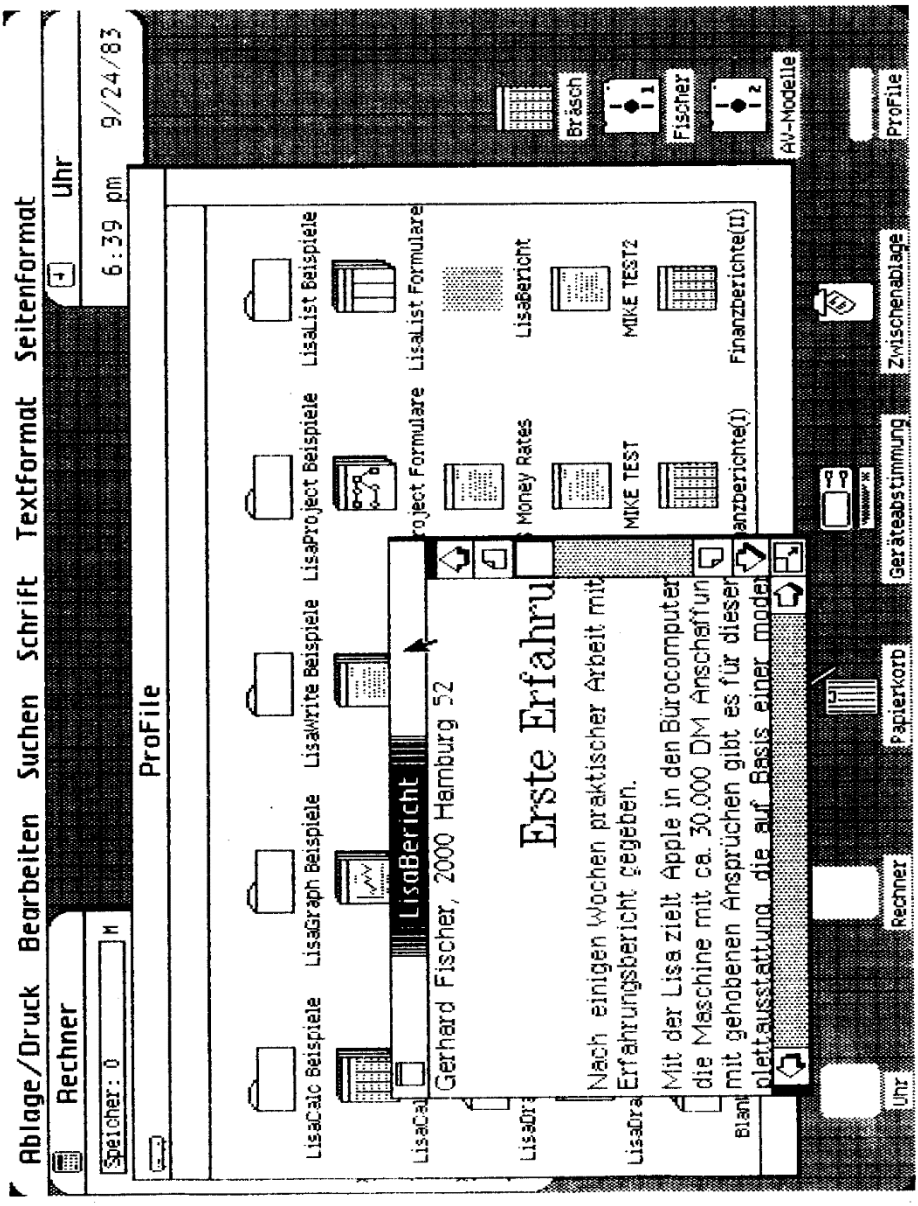
Bei der Lisa fällt einem als erstes die "Bildhaftigkeit" des Bildschirms auf. Alle "Papiere", Symbole und Bedienelemente wirken fast gegenständlich, wie zum anfassen (was ja auch tatsächlich geht mit der Maus).

Die Symbole (in den englischen Handbüchern "Icons" - Ikone - genannt) sind ein wesentliches Bedienungselement der Lisa. Speichern, Sichern, Übertragen auf andere Träger oder Löschen von Unterlagen - das alles geht ganz (vor-) bildlich einfach: Man klickt das entsprechende Symbol an und zieht es mit gedrückter Maustaste über den Bildschirm zur neuen Bestimmung, dem Symbol eines Ordners, einer Diskette oder auch des Papierkorbs - schon erledigt. Keine komplizierten Kommandos, kein Erinnernmüssen des exakten Namens!

Neben den eigentlichen Arbeitsunterlagen können noch andere "Papiere" auf der Tischplatte liegen, z.B. Inhaltsverzeichnisse von der Festplatte oder von Disketten, aber auch von einzelnen "Ordnern" (=Subdirectories), die man auf der Festplatte oder den Disketten anlegen kann und in denen wiederum Arbeitsunterlagen abgelegt sind. Diese Inhaltsverzeichnisse können nach (Maus-) Wahl alphabetisch oder chronologisch geordnet oder in Symbolen dargestellt sein.

Lisa verwaltet den Schreibtisch

Nach dem Einschalten, was übrigens eine längere Prozedur ist, stellt Lisa den Schreibtisch so wieder her, wie er in der vorhergehenden Sitzung verlassen wurde: Hat man vorher alles or-


Ablage/Druck Bearbeiten Suchen Schrift Textformat Seitenformat
Rechner
Speicher: 0
ProFile
Uhr
6:39 pm 9/24/83
LisaCalc Beispiele
LisaGraph Beispiele
LisaWrite Beispiele
LisaProject Beispiele
LisaList Beispiele
LisaBericht
Gerhard Fischer, 2000 Hamburg 52
Erste Erfahru
Nach einigen Wochen praktischer Arbeit mit
Erfahrungsbericht gegeben.
Mit der Lisa zielt Apple in den Bürocomputer
die Maschine mit ca. 30.000 DM Anschaffun
mit gehobenen Ansprüchen gibt es für dieser
LisaList Formulare
Money Rates
LisaBericht
MIKE TEST
MIKE TEST2
Finanzberichte(I)
Finanzberichte(II)
Bräsch
Fischer
AV-Modelle
ProFile
Zwischenablage
Geräteabstimmung
Papierkorb
Rechner
Uhr

dentlich weggeräumt, so findet man eine leere "Platte" vor, wenn nicht, packt Lisa einem den ganzen Wust wieder so hin, wie man ihn verlassen hat. *Leer* ist der Schirm allerdings nicht ganz. Man findet zumindest in der Kopfleiste die Titel der Basismenüs vor, die man bei Bedarf mit der Maus (ähnlich wie Rollos) "herunterziehen" und durch Anklicken eines dort erscheinenden Befehls in Aktion umsetzen kann, sowie am unteren Rand eine Reihe von Symbolen, wie "Profile" (=Festplatte), "Zwischenablage", "Geräteabstimmung", "Papierkorb" und, wenn man will, noch "Uhr" und "Rechner".

Letzterer ist ein kleiner Gag für sich: Es handelt sich um einen im Bildschirm eingeblendeten Taschenrechner, der wahlweise mit der Maus oder über das Zahlenfeld auf der Tastatur bedient werden kann. Zahlreiche Optionen machen den Rechner vielseitig: Eingefleischte Hewlett-Packard-Benutzer brauchen zum Beispiel nicht auf ihre UPN-Logik zu verzichten.

Das Aufräumen des Schreibtisches gestaltet sich einfacher als im praktischen Leben: Nach Anklicken des Befehls "Alles zur Seite legen" verwandelt sich der Papierwust in hübsche kleine Ikone, die dann ordentlich aufgereiht und mit Namen versehen übersichtlich auf der "Platte" liegen.

Wartezeiten störend

Bei allen Transaktionen, wie z.B. Sichern oder Übertragen, erscheint das Symbol einer Sanduhr auf dem Schirm oder gar die Worte "Moment bitte", was in der Regel eine gewisse Wartezeit bedeutet, die zumindest subjektiv als lang empfunden wird. Probe auf`s Exempel: Ein an dieser Stelle gegebener Befehl "Sichern & weiterarbeiten" führt zu einer optisch und akustisch gut wahrnehmbaren "Schaufelarbeit" der Platte, die zu einer Arbeitspause von knapp 30 Sekunden Dauer zwingt. Es besteht der Eindruck, daß die mitgelieferte Festplatte angesichts der Größe der zu "schaufelnden" Systeme sowie der Leistungsfähigkeit der CPU einen Engpaß darstellt. Es wäre wünschenswert, eine schnellere Platte zu haben, die auch ruhig etwas größer sein dürfte, da die jetzige zu ca. zwei Dritteln bereits von den Systemen belegt ist.

Es heißt übrigens, daß bei den ersten Lisen noch ein Aufzeichnungsprogramm mitläuft, welches bei eventuell noch auftretenden Fehlern eine Rekonstruktion des Geschehenen und damit das Erkennen der Ursachen ermöglicht. Sobald dieses Programm in späteren Systemfreigaben (z.Z. Release 1.1) entfernt wird, sollen sich die Wartezeiten erheblich verkürzen.

Die Lisa-Bürosysteme

Zum Lieferumfang der Maschine gehören 6 Systeme:

LisaCalc:	Kalkulations- (Spreadsheet-) Programm
LisaGraph:	Präsentationsgrafik
LisaDraw:	Zeichenprogramm
LisaWrite:	Textsystem
LisaProject:	Projektsystem (Netzplantechnik)
LisaList:	Dateiverwaltungssystem

Dazu kommt noch als 7. System "LisaTerminal", welches in Deutschland nicht zum Lieferumfang gehört und mit 850 DM extra bezahlt werden muß. Es steht noch nicht zur Verfügung, soll aber dem Vernehmen nach hervorragend geeignet sein für die VT100 Emulation.

Die Systeme können hier natürlich nicht im einzelnen besprochen werden. Etwas pauschal läßt sich jedoch folgendes sagen: Bis auf LisaList sind alle Systeme hervorragend und sehr gut durchdacht. Wegen der weitgehend gleichen Bedienungselemente sind sie darüberhinaus außerordentlich leicht zu erlernen und zu handhaben. Hierin liegt zweifellos der ganz große Vorzug der Lisa!

Natürlich braucht man etwas mehr Übung, um sich in jedem der einzelnen Systeme mehr als nur oberflächlich zurechtzufinden. Es gibt dafür zwar keine so schicken Lernprogramme wie LisaGuide, aber man tut gut daran, wenn man zu Beginn das in dem jeweiligen Handbuch befindliche "Tutorial" durchgeht. Das genügt in der Regel, um in dem System gut arbeiten zu

können.

Auf einige Einzelheiten soll hier aber dennoch hingewiesen werden:

Das **LisaCalc** vergleicht sich im wesentlichen mit anderen *guten* Kalkulationsprogrammen. Die Möglichkeit, mit der Maus im Arbeitsbogen herumfahren zu können, ist natürlich eine erhebliche Vereinfachung. Für **LisaWrite** trifft dies ebenfalls zu.

Wer je mit anderen Grafiksystemen gearbeitet hat, wird die Einfachheit und Schnelligkeit begrüßen, mit der man in **LisaGraph** Präsentationsgrafiken erstellen kann. Besonders angenehm ist, daß man das Bild und die Zahlenwerte gleichzeitig auf dem Bildschirm hat. Jede Änderung oder Ergänzung der Zahlen wird sofort im Bild verarbeitet. Auch die Umsetzung von z.B. einer Balkenchart zu einer Linienchart geht im Nu vonstatten. Lästig ist dagegen, daß es manchmal schwierig ist, Raster unterzulegen oder daß es nicht möglich ist, reine Linien zu erzeugen, d.h. Linien ohne aufsitzende Markierungen. Begeisternd einfach ist es, mit LisaCalc Zahlenwerte zu erzeugen, diese in LisaGraph hinüberzuziehen und in ein Bild umzusetzen.

Ganz hervorragend ist **LisaDraw**. Mit diesem System lassen sich Halb- und Ganzgrafiken, Schemazeichnungen usw. bis hin zu richtigen Bildern erzeugen. Der Phantasie sind keine Grenzen gesetzt! Da man mit LisaGraph erzeugte Grafiken in LisaDraw hinüberziehen und dort ergänzen kann, werden einige der Limitierungen von LisaGraph ausgeglichen.

LisaProject ist das System, welches Techniker und Organisatoren am meisten begeistert. Verständlich, wenn man bedenkt, daß die ungeheure Mühe, die es normalerweise kostet, einen größeren Netzplan zu ändern, nun in Minuten von Lisa erledigt wird.

Alle Systeme verfügen über die vielfältigen Schriftmöglichkeiten mit bis zu 11 verschiedenen Schrifttypen sowie Optionen wie Unterstreichen, Fett, Kursiv, usw.. Dieser Artikel wurde übrigens mit LisaWrite erstellt, in "Proportional Modern" mit Blocksatz und auf dem Matrixdrucker im hochauflösenden Modus ausgedruckt. Dabei konnten dank der einblend- und einstellbaren Zeilen- und Seitenlineale die gewünschten Maße für den Satzspiegel millimetergenau eingehalten werden. Der Drucker ist zwar mit dem Apple-Logo versehen, es handelt es sich aber um ein modifiziertes Modell aus der bewährten Itoh-Familie.

Bisher keine Fertigsoftware

Für den Spezialisten im Büro ist Lisa mit ihren vielfältigen Gestaltungsmöglichkeiten die richtige Maschine. Wer dagegen, z.B. als kleiner Gewerbetreibender, anwendungsfertige Software wie Buchhaltungssysteme oder ähnliches sucht, kann mit Lisa zur Zeit noch wenig anfangen. Das scheint beim Absatz der Lisa in den U.S.A. ein Handicap darzustellen, dem Apple dem Vernehmen nach mit einer kräftigen Preissenkung und einer Aufschnürung des Lieferpaketes begegnen will. Offenbar ziehen große Käuferschichten den IBM-PC plus preiswerter Fertigsoftware vor, zumal Lisa gegenüber solchen Systemen optisch teuer wirkt. Dabei wird jedoch verkannt, daß der IBM-PC, durch zusätzliche Investitionen auf vergleichbare Leistungsmerkmale gebracht (1 MB RAM, Festplatte, usw.), erheblich teurer ist als Lisa, soweit vergleichbare Leistungen überhaupt angeboten werden. Die mausbedienten Bürosysteme z.B. sind in dieser Fülle bisher konkurrenzlos. Mit "VisiOn" wird zwar bereits etwas ähnliches angeboten, es umfaßt bisher aber nur ein Kalkulations- und ein Textsystem. Dem Mangel an Fertigsoftware wird dann abgeholfen, wenn, wie angekündigt, Digital Research sein Concurrent CP/M für die Lisa anbietet.

Was wird einem Selbstprogrammierer geboten?

Bis jetzt leider noch nichts - jedoch ist noch für dieses Jahr die Freigabe eines "Workshops" vorgesehen, der innerhalb des Lisa-OS die Programmsprachen PASCAL (Iso), BASIC PLUS (DEC-kompatibel), COBOL sowie Assembler verfügbar macht. Der Workshop soll für Programmeditierung und sonstiges Handling ebenfalls die Mausbedienung nutzen, was den Komfort erheblich erhöhen dürfte. Wie bereits bisher bei Apple üblich, so ist auch für die Lisa Pascal "the language of choice". Der Pascal-Quelltext wird in zwei Schritten zunächst auf P-Code und dann auf MC68000-Maschinencode heruntercompiliert, was *superschnelle* Ausführungszeiten verspricht!

Die Wahl von ISO-PASCAL für die Lisa hat auf der einen Seite den Vorteil, daß nun endlich ein Debugger zur Verfügung stehen wird. Der Nachteil ist leider, daß Apple II & III Programme in UCSD-Pascal nicht ohne weiteres auf Lisa übertragen werden können. Wie groß der Modifizierungsaufwand sein wird, bleibt abzuwarten.

Für später ist noch ein "Toolkit" angekündigt, welches es erlaubt, auch in selbsterstellten Pascal-Programmen die Mausbedienung zu nutzen, was die Erstellung besonders anwenderfreundlicher Programme ermöglicht.

An Betriebsystemen soll außer dem bereits erwähnten Concurrent CP/M später noch XENIX von Microsoft angeboten werden. Ob auch MS-DOS verfügbar gemacht wird, war nicht zu erfahren.

Gesamturteil

Außerordentlich komfortables Einplatzsystem für hohe Leistungsansprüche. Hard- und Software wirken sehr gut durchdacht.

Pluspunkte:

+ Hoher Bedienungskomfort, leicht erlernbar - auch für Computer-Laien.
+ Sehr gute Universal-Softwaresysteme mit hohem Gebrauchsnutzen für Spezialisten.
+ Gute Grafikmöglichkeiten.
+ Große Schriftvielfalt.
+ Komplette Ausstattung.
+ Erfreulich große Diskettenkapazität (860 KB).
+ Gute Druckerqualität.

Minuspunkte:

- Bis jetzt keine Fertigsoftware verfügbar.
- Keine Kompatibilität zu anderen Apple-Produkten.
- Keine Farbgrafik.
- Wartezeiten beim File-Handling - Festplatte könnte schneller sein.
- Nicht transportfreundlich.

St.Dir. Peter Rix, 2350 Neumünster

AIM-Betriebssystem für CBM-Floppy

Vorbemerkungen

Durch grundlegende und anwendungsorientierte Beiträge in den Heften 19, 21 und 27 des 65xx MICRO MAG ist der Betrieb eines Commodore-Floppy für den AIM erschlossen worden. Der vorliegende Beitrag stellt nun ein Betriebssystem vor, das alle Ein- und Ausgabeoptionen des AIM 65 für das Floppy erschließt. Zielsetzung der Arbeit war es, alle Standard-Software des AIM 65, wie Assembler, Editor, BASIC, FORTH zu unterstützen, ohne benutzerspezifische Betriebssystemerweiterungen zu stören.

Variable des Betriebssystems

Das Betriebssystem benötigt keine Zero-Page-Adressen. Belegt werden die Single-Step-Variablen des Monitors. Der Kommandostring wird im Displaybuffer verwaltet. Störungen irgendwelcher Standardsoftware sind damit praktisch ausgeschlossen.

Bedienung des Floppy

Nach Initialisierung des Betriebssystems mit Sprung an seine Anfangsadresse wird die F3-Taste mit einem Menue belegt, das über die Taste 'D' das Directory aufruft, über Taste 'K' den Kommandokanal öffnet und über jede andere Taste zum Monitor zurückkehrt.

Schreib- und Leseoperationen werden von der jeweils benutzten Softwareebene über die User-Optionen IN=U bzw. OUT=U eingeleitet, d.h. von BASIC über SAVE und LOAD, vom Editor über 'R' und 'L', von FORTH über SOURCE usw.. Das Floppy gibt nach Beendigung von Schreib- und Leseoperationen die Kontrolle an die aufrufende Software-Ebene zurück.

Das Floppy wird über Kommandostrings des CBM-BASIC 3 (siehe Floppy-Handbuch) angesprochen. Dabei gelten folgende Besonderheiten:

Memory-Dump und -Load im AIM-Format erfolgt in PRG-Dateien des Floppy, Texte (Editor, BASIC) werden als SEQ-Dateien abgelegt (sequentielle). Wird in einem Kommandostring keine Floppy-Nummer angegeben, so wird auf das zuletzt aktive Laufwerk zugegriffen. Für das Directory wird nur die Laufwerksnummer eingegeben, das $-Zeichen wird automatisch erzeugt. Hinter der Laufwerksnummer können, durch Doppelpunkt abgetrennt, Suchbegriffe mit Jokerzeichen folgen. Das Floppy listet dann nur ein Teil-Directory. Ein laufendes Directory kann mit der Leertaste angehalten und mit der Taste 'H' beendet werden.

Fehlerbehandlung

Durch eine umfangreiche Syntaxprüfung des Kommandostrings werden viele Eingabefehler bereits vor dem Floppy abgefangen. Mit den Meldungen NEW:, READ: oder WRITE: erlaubt das Betriebssystem dann eine Wiederholung der Stringeingabe (z.B. wenn ein Memory-Dump in eine SEQ-Datei erfolgen soll). Werden Fehler erst durch das Floppy erkannt, so erfolgt nach Ausdruck der Floppy-Fehlermeldung Rückkehr zum Monitor.

Besondere Eigenschaften

Mit der etwas lästigen Eigenschaft des AIM 65, für fast jede Betriebssoftware andere Ende-Zeichen der Dateien zu verlangen, wurde beim Floppy-Betriebssystem gründlich aufgeräumt.

Ohne jede Einschränkung und besondere Maßnahmen liest der Editor BASIC Texte, der BASIC-Interpreter Editortexte und der Editor den Memory-Dump im AIM-Format. Auf der Diskette endet jede Datei mit einem einfachen Carriage-Return, die notwendigen Zusatzzeichen, wie weiteres Carriage-Return oder CTRL-Z erzeugt das Betriebssystem als Pseudozeichen selber.

Das Floppy-Betriebssystem kann je einen Schreib- und Lesekanal gleichzeitig offenhalten und bedienen. Damit ist unter anderem das Assemblieren von Floppy zu Floppy selbst bei nur einem Laufwerk möglich. Der Quelltext wird mit IN=U vom Floppy geladen und der Objectcode mit OBJ Y und OUT=U wieder an das Floppy übergeben; der Rechnerspeicher steht voll für die Symboltabelle zur Verfügung. Beim Lesen des Quelltextes über User-Input macht sich ein Fehler des AIM-Assemblers bemerkbar. Der Pass 2 wird nicht gestartet, weil der 1. Sprung über den UIN-Vektor mit gesetztem Carry statt, wie erforderlich, mit gelöschtem Carry erfolgt. Abhilfe ist hier auf zwei Arten möglich:

1. **Ohne Assembler-Änderung**
   Das erste Zeichen im Quelltext darf nicht bedeutsam sein, zweckmäßigerweise läßt man am Anfang der 1. Zeile ein Leerzeichen
   Aus dem auf Diskette aufgezeichneten Quelltext-File erzeugt man ein Arbeits-File, indem man mit 'CONCATE' das Quelltext-File zweimal aneinanderhängt, z.B.

   CO:WORK=0: QUELLE,0: QUELLE

   Der Assembler verarbeitet dann das Arbeitsfile WORK ohne Unterbrechung.
2. **Mit Änderung des Assemblers**
   Die Änderung des Assemblers ist u.a. dann notwendig, wenn die .FILE-Option des Assemblers genutzt werden soll oder wenn die Erzeugung eines WORK-Files vermieden werden soll. Folgender Code im Assembler-ROM bringt die korrekte Funktion:

```
DFCF   C954     CMP #'T'
DFD1   D003     BNE $03
DFD3   4C2FE3   JMP $E32F
DFD6   4C68E8   JMP $E868
DFD9   EA       NOP
  :     :        :
DFDC   EA       NOP
```

Der Assembler stoppt jetzt am Pass-Ende oder nach FILE und fordert mit READ: eine Lesedatei an.

Assembler und FORTH warten beim Einlesen von Text-Dateien nicht auf das Dateiende, sondern brechen nach dem Lesen von Schlüsselworten die Dateibearbeitung ab. Damit das Floppy seine Datei wieder ordnungsgemäß schließen kann, müssen die Schlüsselworte .END bzw. FINIS in der letzten Quelltext-Zeile stehen.

Neben U sind auch alle anderen Ein- und Ausgabeoptionen im Assembler zulässig, so daß ohne weiteres die Assemblierung Memory to Floppy, Floppy to Memory, Tape to Floppy und Floppy to Tape möglich ist.

Anschlußschema

Der Anschlußplan aus Heft 19 des 65xx MICRO MAG wurde übernommen. Da sich Bustreiber als überflüssig erwiesen haben, wurde der CB2-Anschluß des VIA 6522 für ein Interface-Reset benutzt. Will man gleichwohl mit der Bustreiber-Schaltung arbeiten, so läßt man im Programm die IFC-Routine wegfallen und schaltet CB2 immer im Zusammenhang mit DDRA (Datenrichtungsregister) um, und zwar an 3 Stellen im Programm.

```
CBM-8050 "IEEE-FLOPPY 09"
BASIC/ FORTH/ EDITOR/ASSEMBLER/ DUMP/ LOAD
RIX, 18.09.1983
-----------------------------------------

BUSSIGNALE IEEE-PIN  1/AIM-PA0  : DIO1
           IEEE-PIN  2/AIM-PA1  : DIO2
           IEEE-PIN  3/AIM-PA2  : DIO3
           IEEE-PIN  4/AIM-PA3  : DIO4
           IEEE-PIN 13/AIM-PA4  : DIO5
           IEEE-PIN 14/AIM-PA5  : DIO6
           IEEE-PIN 15/AIM-PA6  : DIO7
           IEEE-PIN 16/AIM-PA7  : DIO8
           IEEE-PIN  6/AIM-PB3/7: DAV
           IEEE-PIN  7/AIM-PB2/6: NRFD
           IEEE-PIN  8/AIM-PB1/5: NDAC
           IEEE-PIN  5/AIM-PB0/4: EOI
           IEEE-PIN 11/AIM-CA1/2: ATN
           IEEE-PIN 10/AIM-CB1  : SRQ
           IEEE-PIN  9/AIM-CB2  : IFC
           IEEE-PIN 17/AIM-GND  : REN
           IEEE-PIN 12/18...24  : GND

PRIMAERADRESSE    $2G  : G IST LISTENER
                  $4G  : G IST TALKER
                  $3F  : UNLISTEN ALLE
                  $5F  : UNTALK ALLE
                  G=8  : FLOPPY

SEKUNDAERADRESSE  $6K  : DATENKANAL K
                  $7K  : DATENKANAL K
```

```
                    $EK   : CLOSE KANAL K
                    $FK   : OPEN  KANAL K
                    K=0   : LESEKANAL, PRG
                    K=1   : SCHREIBKANAL, PRG
                    K=2   : LESEKANAL, SEQ
                    K=3   : SCHREIBKANAL, SEQ
                    K=0-E: DATENKANAELE
                    K=F   : KOMMANDOKANAL

0000            ;KONSTANTE
0000 LSN                =$28
0000 TLK                =$48
0000 UNLSN              =$3F
0000 UNTLK              =$5F
0000 OPEN               =$F0
0000 CLOSE              =$E0
0000
0000            ;VARIABLEN-ADRESSEN
0000 WJUMP              =$100           ;SPRUNGVEKTOR, WRITE
0000 RJUMP              =$102           ;SPRUNGVEKTOR, READ
0000 WKANAL             =$104           ;KANAL-#, WRITE
0000 RKANAL             =$105           ;KANAL-#, READ
0000                                    ;KANAL INAKTIV: BIT 7 = 1
0000 UIN                =$108           ;USER-IN-VEKTOR
0000 UOUT               =$10A           ;USER-OUT-VEKTOR
0000 F3                 =$112           ;F3-BEFEHLSTASTE
0000 VIA                =$A000          ;PORT-BASISADRESSE
0000 DRB                =VIA
0000 DRA                =VIA+1
0000 DDRB               =VIA+2
0000 DDRA               =VIA+3
0000 PCR                =VIA+12
0000 WLCHR              =$A40E          ;LETZTES ZEICHEN, WRITE
0000 RLCHR              =$A40F          ;LETZTES ZEICHEN, READ
0000 WFLG               =$A410          ;WRITE-FLAG
0000 RFLG               =$A414          ;READ-FLAG
0000 S1                 =$A41A          ;ADRESS-ZEIGER
0000 STRING             =$A43E          ;KOMMANDO-ZEILE
0000                                    ;VERTRAEGLICH MIT DISPLAY
0000 HEX                =WJUMP
0000 BCD                =RJUMP
0000
0000            ;MONITOR-ADRESSEN
0000 BLANK              =$E83E
0000 RCHEK              =$E907
0000 MONIT              =$E956
0000 RDRUB              =$E95F
0000 OUTPUT             =$E97A
0000 CRLOW              =$EA13
0000 WRAX               =$EA42
0000 NUMA               =$EA46
0000 PHXY               =$EB9E
0000 PLXY               =$EBAC
0000
0000                    * =F3
0112            4C0338 JMP INITI
0115
```

```
0115                    * =$3800
3800                200B3D JSR IFC          ;INTERFACE-RESET
3803 INITI          20743B JSR NEUTRL       ;BUS-GRUNDZUSTAND
3806                8D0401 STA WKANAL       ;KANAL UNBENUTZT
3809                8D0501 STA RKANAL       ;INHALT $FF
380C                A959   LDA #<USOINI
380E                8D0A01 STA UOUT
3811                A938   LDA #>USOINI
3813                8D0B01 STA UOUT+1
3816                A96A   LDA #<USIINI
3818                8D0801 STA UIN
381B                A938   LDA #>USIINI
381D                8D0901 STA UIN+1
3820                A94C   LDA #$4C         ;JMP
3822                8D1201 STA F3           ;F3-TASTE
3825                A903   LDA #<INITI
3827                8D1301 STA F3+1
382A                A938   LDA #>INITI
382C                8D1401 STA F3+2
382F                A034   LDY #MES7-MES1   ;MENUEZEILE
3831                20A93C JSR MESOUT       ;ZUR AUSGABE
3834                A000   LDY #0
3836                205FE9 JSR RDRUB
3839                2013EA JSR CRLOW
383C                C944   CMP #'D'         ;DIRECTORY?
383E                F007   BEQ DIRC
3840                C94B   CMP #'K'         ;KOMMANDO?
3842                F00E   BEQ KOMMDO
3844                4C56E9 JMP MONIT        ;NEIN, FERTIG
3847
3847 DIRC           A015   LDY #MES4-MES1   ;DIRECTORY: $
3849                20A93C JSR MESOUT
384C                A924   LDA #'$'
384E                A000   LDY #0
3850                F02F   BEQ COM
3852
3852 KOMMDO         A00E   LDY #MES3-MES1   ;KDO:
3854                20A93C JSR MESOUT
3857                D023   BNE COMDI        ;STRING HOLEN
3859
3859 USOINI         9003   BCC USO1         ;USER-OUTPUT
385B                6C0001 JMP (WJUMP)      ;FOLGEDURCHLAUF
385E USO1           209EEB JSR PHXY         ;1. DURCHLAUF
3861 USO2           20743B JSR NEUTRL
3864                A201   LDX #1           ;SCHREIB-ZEIGER
3866                A000   LDY #MES1-MES1   ;WRITE:
3868                F00F   BEQ COMDIN       ;STRING HOLEN
386A
386A USIINI         9003   BCC USI1         ;USER-INPUT
386C                6C0201 JMP (RJUMP)      ;FOLGEDURCHLAUF
386F USI1           209EEB JSR PHXY         ;1. DURCHLAUF
3872 USI2           20743B JSR NEUTRL
3875                A280   LDX #$80         ;LESE-ZEIGER
3877                A007   LDY #MES2-MES1   ;READ:
3879
3879 COMDIN         20A93C JSR MESOUT       ;KOMMANDOSTRING HOLEN
387C COMDI          A000   LDY #0           ;ZEICHENZAEHLER
387E COMD           205FE9 JSR RDRUB
```

```
3881 COM      993FA4 STA STRING+1,Y
3884          C8     INY
3885          C90D   CMP #$0D
3887          D0F5   BNE COMD
3889          207AE9 JSR OUTPUT
388C          8C3EA4 STY STRING
388F          AD3FA4 LDA STRING+1     ;1. ZEICHEN
3892          C924   CMP #'$'
3894          F06B   BEQ DIREC        ;ZUM DIRECTORY
3896          C931   CMP #'1'         ;FLOPPY-NUMMER?
3898          F018   BEQ CMD2
389A          C930   CMP #'0'
389C          F014   BEQ CMD2
389E          C93A   CMP #':'         ;ALTE DRIVE-#?
38A0          F010   BEQ CMD2
38A2          A20F   LDX #$F          ;KOMMANDOKANAL
38A4          20D13B JSR HEADER       ;OEFFNEN
38A7          2013EA JSR CRLOW
38AA          A26F   LDX #$60+$F      ;KANAL 15
38AC          20FB3B JSR TALKER       ;ZUM LESEN OEFFNEN
38AF          4CC739 JMP FPMES        ;FLOPPY-MELDUNG LESEN
38B2 CMD2     B93EA4 LDA STRING,Y     ;STRING PRUEFEN
38B5          C92C   CMP #','
38B7          F00D   BEQ AA8
38B9          C93A   CMP #':'
38BB          F020   BEQ AA9
38BD          88     DEY
38BE          D0F2   BNE CMD2
38C0          20A13C JSR SYERR        ;FEHLER
38C3          4C7938 JMP COMDIN       ;STRING HOLEN
38C6
38C6 AA8      C8     INY              ;DATEITYP PRUEFEN
38C7          B93EA4 LDA STRING,Y
38CA          C950   CMP #'P'
38CC          F00F   BEQ AA9          ;PRG-DATEI, X=1/$80 W/R
38CE          C953   CMP #'S'
38D0          F009   BEQ AB1          ;SEQ-DATEI, X=3/$82 W/R
38D2          C957   CMP #'W'
38D4          F005   BEQ AB1
38D6          20A13C JSR SYERR        ;SYNTAXFEHLER
38D9          D09E   BNE COMDIN       ;JMP
38DB AB1      E8     INX              ;X:=X+2
38DC          E8     INX
38DD AA9      8A     TXA
38DE          0A     ASL A            ;2*X UND CC
38DF          AA     TAX
38E0          B00F   BCS AA10
38E2          BD033D LDA TAB,X        ;SPRUNGVEKTOR FUER
38E5          8D0001 STA WJUMP        ;SCHREIBDATEI
38E8          BD043D LDA TAB+1,X      ;LADEN
38EB          8D0101 STA WJUMP+1
38EE          6C0001 JMP (WJUMP)      ;DATEIBEARBEITUNG, WRITE
38F1 AA10     18     CLC
38F2          BD033D LDA TAB,X        ;SPRUNGVEKTOR FUER
38F5          8D0201 STA RJUMP        ;LESEDATEI
38F8          BD043D LDA TAB+1,X      ;LADEN
38FB          8D0301 STA RJUMP+1
38FE          6C0201 JMP (RJUMP)      ;DATEIBEARBEITUNG, READ
3901
```

```
3901 DIREC   A200   LDX #0          ;DIRECTORY
3903         8E0501 STX RKANAL      ;KANAL-#
3906         20D13B JSR HEADER      ;OEFFNEN
3909         20B139 JSR CHECK       ;FEHLER?
390C         A260   LDX #$60+$0     ;OK, KANAL-#
390E         20FB3B JSR TALKER
3911         A005   LDY #5
3913 DIR1    20263C JSR GETCHR
3916         88     DEY
3917         D0FA   BNE DIR1
3919         2013EA JSR CRLOW
391C         2046EA JSR NUMA
391F         20263C JSR GETCHR
3922         20263C JSR GETCHR
3925         203EE8 JSR BLANK
3928 DIR2    20263C JSR GETCHR      ;KOPFZEILE
392B         F005   BEQ DIR3
392D         207AE9 JSR OUTPUT
3930         D0F6   BNE DIR2
3932 DIR3    2013EA JSR CRLOW
3935         A01B   LDY #27
3937         A92D   LDA #'-'
3939 DIR4    207AE9 JSR OUTPUT      ;UNTERSTREICHE
393C         88     DEY
393D         D0FA   BNE DIR4
393F         2013EA JSR CRLOW
3942 DIR5    20263C JSR GETCHR      ;KATALOGZEILE
3945         9064   BCC DIREND
3947         C901   CMP #1          ;STARTBYTE
3949         D0F7   BNE DIR5
394B         20263C JSR GETCHR      ;2. STARTBYTE
394E         20263C JSR GETCHR      ;BLOCK-# LOW
3951         8D0101 STA HEX+1
3954         20263C JSR GETCHR      ;BLOCK-# HIGH
3957         8D0001 STA HEX
395A         F8     SED             ;HEX-DEZ-WANDLUNG
395B         A900   LDA #0
395D         8D0301 STA BCD+1
3960         8D0201 STA BCD
3963         A010   LDY #16
3965         18     CLC
3966 DIR6    AD0301 LDA BCD+1
3969         6D0301 ADC BCD+1
396C         8D0301 STA BCD+1
396F         AD0201 LDA BCD
3972         6D0201 ADC BCD
3975         8D0201 STA BCD
3978         2E0101 ROL HEX+1
397B         2E0001 ROL HEX
397E         88     DEY
397F         10E5   BPL DIR6
3981         AE0301 LDX BCD+1
3984         D8     CLD
3985         2042EA JSR WRAX        ;BLOCKZAHL DEZ. SCHREIBEN
3988         203EE8 JSR BLANK
398B DIR7    20263C JSR GETCHR
398E         F011   BEQ DIR9
3990         C920   CMP #$20
3992         F0F7   BEQ DIR7
```

```
3994           207AE9 JSR OUTPUT
3997 DIR8      20263C JSR GETCHR
399A           F005   BEQ DIR9
399C           207AE9 JSR OUTPUT
399F           D0F6   BNE DIR8
39A1 DIR9      2013EA JSR CRLOW
39A4           2007E9 JSR RCHEK        ;HALT?
39A7           C948   CMP #'H'         ;ENDE?
39A9           D097   BNE DIR5
39AB DIREND    20763A JSR FINRED
39AE           4C56E9 JMP MONIT
39B1
39B1 CHECK     A26F   LDX ##60+$F      ;FEHLERKANAL
39B3           20FB3B JSR TALKER       ;IST TALKER
39B6           20263C JSR GETCHR
39B9           C930   CMP #'0'
39BB           D004   BNE CHE1
39BD           208A3B JSR UNTALK       ;ALLES OK
39C0           60     RTS
39C1 CHE1      2013EA JSR CRLOW
39C4           207AE9 JSR OUTPUT
39C7 FPMES     20263C JSR GETCHR       ;FLOPPY-TEXT
39CA           08     PHP
39CB           207AE9 JSR OUTPUT
39CE           28     PLP
39CF           B0F6   BCS FPMES
39D1           208A3B JSR UNTALK
39D4           A9EF   LDA ##$E0+$F     ;SEK.ADR
39D6           20BA3B JSR LISTEN       ;FEHLERKANAL SCHLIESSEN
39D9           20A33B JSR UNLISN
39DC           4C56E9 JMP MONIT        ;FERTIG
39DF
39DF REDPRG    B048   BCS RDPDAT       ;PRG-DATEI LESEN
39E1           A200   LDX #0           ;KANAL-#
39E3           F01F   BEQ REDPS        ;JMP
39E5
39E5 ERR       20A13C JSR SYERR        ;SYNTAX-FEHLER
39E8           4C7238 JMP USI2         ;NEUER LESEVERSUCH
39EB DSEQ      B054   BCS RDSDAT       ;SEQ-DATEI LESEN
39ED           B93EA4 LDA STRING,Y
39F0           C953   CMP #'S'         ;LESEDATEI?
39F2           D0F1   BNE RDERR
39F4           AD7DA4 LDA $A47D        ;JMP ( ) VOM MONITOR
39F7           C9E6   CMP ##$E6        ;MONITOR <L>-BEFEHL?
39F9           D007   BNE REDS
39FB           AD7EA4 LDA $A47E
39FE           C9E2   CMP ##$E2
3A00           F0E3   BEQ RDERR
3A02 REDS      A202   LDX #2           ;KANAL-#
3A04 REDPS     8E0501 STX RKANAL       ;KANAL-#, READ
3A07           8E1AA4 STX S1           ;ADRESS-ZEIGER VERSTELLEN
3A0A           20D13B JSR HEADER       ;OEFFNE LESEKANAL
3A0D           20B139 JSR CHECK        ;FEHLER?
3A10           AD0501 LDA RKANAL
3A13           297F   AND ##$7F
3A15           0960   ORA ##$60
3A17           AA     TAX              ;SEK.ADR, TALKER
3A18           20FB3B JSR TALKER
```

```
3A1B           A980    LDA #$80
3A1D           8D14A4  STA RFLG
3A20           A90D    LDA #$0D
3A22           8D0FA4  STA RLCHR
3A25           20ACEB  JSR PLXY
3A28 RBACK     60      RTS
3A29
3A29 RDPDAT    20163C  JSR GETTST      ;PRG-DATEN LESEN
3A2C           900D    BCC RDPFIN
3A2E           C90D    CMP #$0D        ;CR?
3A30           D005    BNE RDPCHR
3A32           CD0FA4  CMP RLCHR       ;2. CR?
3A35           F0F2    BEQ RDPDAT      ;IGNORIEREN
3A37 RDPCHR    8D0FA4  STA RLCHR
3A3A           60      RTS
3A3B RDPFIN    20763A  JSR FINRED      ;LETZTES ZEICHEN
3A3E           4C56E9  JMP MONIT       ;FERTIG
3A41
3A41 RDSDAT    2C14A4  BIT RFLG        ;SEQ-DATEN LESEN
3A44           300A    BMI RDS1
3A46           18      CLC             ;DATEI-ENDE
3A47           AD14A4  LDA RFLG        ;ZUERST CR
3A4A           690D    ADC #$0D        ;DANACH CTRL-Z
3A4C           8D14A4  STA RFLG        ;VORTAEUSCHEN
3A4F           60      RTS             ;EDITOR-/BASIC-ENDE
3A50 RDS1      20163C  JSR GETTST      ;ZEICHEN LESEN
3A53           B00C    BCS RDS2
3A55           20763A  JSR FINRED      ;CC, LETZTES ZEICHEN
3A58           A900    LDA #0          ;DATEI IST GESCHLOSSEN
3A5A           8D14A4  STA RFLG        ;PSEUDOZEICHEN ERZEUGEN
3A5D           A90D    LDA #$0D        ;CR
3A5F           D011    BNE RDS3        ;JMP
3A61 RDS2      C90A    CMP #$0A        ;LF?
3A63           F0EB    BEQ RDS1        ;FALLS JA, IGNORIEREN
3A65           C91A    CMP #$1A        ;CTRL-Z?
3A67           F0E7    BEQ RDS1        ;FALLS JA, IGNORIEREN
3A69           C90D    CMP #$0D        ;CR?
3A6B           D005    BNE RDS3
3A6D           CD0FA4  CMP RLCHR       ;FOLGE-CR?
3A70           F0DE    BEQ RDS1        ;FALLS JA, IGNORIEREN
3A72 RDS3      8D0FA4  STA RLCHR       ;ZEICHEN MERKEN
3A75           60      RTS             ;UND UEBERGEBEN
3A76
3A76 FINRED    8E0FA4  STX RLCHR       ;LESEDATEI SCHLIESSEN
3A79           208A3B  JSR UNTALK
3A7C           AD0501  LDA RKANAL
3A7F           09E0    ORA #$E0        ;SEK.ADR, CLOSE
3A81           20BA3B  JSR LISTEN      ;KANAL SCHLIESSEN
3A84           20A33B  JSR UNLISN
3A87           20743B  JSR NEUTRL      ;BUS GRUNDZUSTAND
3A8A           8D0501  STA RKANAL      ;KANAL-# LOESCHEN
3A8D           AE0FA4  LDX RLCHR
3A90           60      RTS
3A91
3A91 WRERR     20A13C  JSR SYERR       ;SYNTAX-FEHLER
3A94           4C6138  JMP USD2        ;NEUER SCHREIBVERSUCH
3A97 WRIPRG    B04C    BCS WRPDAT      ;PRG-DATEI SCHREIBEN
3A99           AD1AA4  LDA S1
```

```
3A9C           C9E1    CMP #$E1        ;EDITOR =<L>-BEFEHL?
3A9E           D007    BNE WRIP
3AA0           AD1BA4  LDA S1+1
3AA3           C9F7    CMP #$F7
3AA5           F0EA    BEQ WRERR
3AA7 WRIP      A201    LDX #1          ;KANAL-#
3AA9           A93B    LDA #';'

3AAB           D01E    BNE WRIPS       ;JMP
3AAD
3AAD WRISEQ    B073    BCS WRSDAT      ;SEQ-DATEI SCHREIBEN
3AAF           B93EA4  LDA STRING,Y
3AB2           C957    CMP #'W'        ;SCHREIBDATEI?
3AB4           D0DB    BNE WRERR
3AB6           AD7DA4  LDA $A47D       ;JMP ( ) VOM MONITOR
3AB9           C93B    CMP #$3B        ;MONITOR <D>-BEFEHL?
3ABB           D007    BNE WRIS
3ABD           AD7EA4  LDA $A47E
3AC0           C9E4    CMP #$E4
3AC2           F0CD    BEQ WRERR
3AC4 WRIS      A203    LDX #3          ;KANAL-#
3AC6           A980    LDA #$80
3AC8           8D10A4  STA WFLG
3ACB WRIPS     8D0EA4  STA WLCHR
3ACE           8E0401  STX WKANAL
3AD1           20D13B  JSR HEADER      ;SCHREIBKANAL OEFFNEN
3AD4           20B139  JSR CHECK       ;FEHLER?
3AD7           AD0401  LDA WKANAL
3ADA           0960    ORA #$60
3ADC           297F    AND #$7F        ;SEK.ADR, WRITE
3ADE           20BA3B  JSR LISTEN
3AE1           20ACEB  JSR PLXY
3AE4           60      RTS
3AE5
3AE5 WRPDAT    68      PLA             ;PRG-DATEN SCHREIBEN
3AE6           C93B    CMP #';'        ;BLOCKANFANG?
3AE8           D00F    BNE WRP2
3AEA           8D0EA4  STA WLCHR
3AED           A97E    LDA #$7E        ;ZAEHLER BIS 2
3AEF           8D10A4  STA WFLG
3AF2           AD0EA4  LDA WLCHR
3AF5 WRP1      20543C  JSR SNDTST      ;ZEICHEN SENDEN
3AF8           60      RTS
3AF9 WRP2      C930    CMP #'0'        ;ASCII-NULL?
3AFB           D010    BNE WRP3
3AFD           2C10A4  BIT WFLG        ;ZEICHENZAEHLER?
3B00           30F3    BMI WRP1
3B02           CD0EA4  CMP WLCHR
3B05           A960    LDA #$60        ;2*'0'
3B07           6A      ROR A
3B08           8D0EA4  STA WLCHR       ;BIT 7 NACH 2*'0' GESETZT
3B0B           297F    AND #$7F
3B0D WRP3      EE10A4  INC WFLG
3B10           C90D    CMP #$0D
3B12           D0E1    BNE WRP1
3B14           2C0EA4  BIT WLCHR       ;LETZTES CR?
3B17           10DC    BPL WRP1
3B19           20543C  JSR SNDTST      ;CR SENDEN
3B1C           204C3B  JSR FINWRT      ;DATEI SCHLIESSEN
```

```
3B1F           4C56E9 JMP MONIT
3B22
3B22 WRSDAT    68     PLA              ;SEQ-DATEN SCHREIBEN
3B23           C90A   CMP #$0A         ;LF?
3B25           F024   BEQ WSBACK       ;IGNORIEREN
3B27           2C10A4 BIT WFLG         ;DATEI GESCHLOSSEN?
3B2A           101F   BPL WSBACK
3B2C           C90D   CMP #$0D         ;CR?
3B2E           D009   BNE WRS1
3B30           CD0EA4 CMP WLCHR        ; 2. CR?
3B33           F017   BEQ FINWRT       ;JA, DATEI SCHLIESSEN
3B35           8D0EA4 STA WLCHR
3B38           60     RTS
3B39 WRS1      48     PHA
3B3A           AD0EA4 LDA WLCHR
3B3D           C90D   CMP #$0D         ;CR?
3B3F           D003   BNE WRS2
3B41           20543C JSR SNDTST       ;CR SENDEN
3B44 WRS2      68     PLA
3B45           8D0EA4 STA WLCHR
3B48           20543C JSR SNDTST       ;ZEICHEN SENDEN
3B4B WSBACK    60     RTS
3B4C
3B4C FINWRT    20F23B JSR EOILOW       ;SCHREIBDATEI SCHLIESSEN
3B4F           A90D   LDA #$0D         ;CR
3B51           20543C JSR SNDTST       ;MIT EOI=LOW SENDEN
3B54           20A33B JSR UNLISN
3B57           8E0EA4 STX WLCHR
3B5A           AD0401 LDA WKANAL
3B5D           09E0   ORA #$E0         ;SEK.ADR, CLOSE
3B5F           20BA3B JSR LISTEN
3B62           20A33B JSR UNLISN
3B65           20743B JSR NEUTRL
3B68           8D0401 STA WKANAL       ;KANAL-# LOESCHEN
3B6B           A900   LDA #0
3B6D           8D10A4 STA WFLG
3B70           AE0EA4 LDX WLCHR
3B73           60     RTS
3B74
3B74 NEUTRL    A9EE   LDA #$EE
3B76           8D0CA0 STA PCR          ;ATN=HIGH
3B79           A90F   LDA #$0F
3B7B           8D02A0 STA DDRB
3B7E           8D00A0 STA DRB
3B81           A9FF   LDA #$FF
3B83           8D03A0 STA DDRA         ;PORT A IST AUSGANG
3B86           8D01A0 STA DRA
3B89           60     RTS
3B8A
3B8A UNTALK    AD0501 LDA RKANAL       ;TALKER ABSTELLEN
3B8D           0980   ORA #$80         ;FLAG SETZEN
3B8F           8D0501 STA RKANAL
3B92           A95F   LDA #UNTLK
3B94 ATLO      20AF3B JSR ATNLO
3B97 LASTKD    20703C JSR SEND
3B9A ATNHI     AD0CA0 LDA PCR
3B9D           0902   ORA #%10
3B9F           8D0CA0 STA PCR          ;ATN=HIGH
3BA2           60     RTS
```

```
3BA3 UNLISN   AD0401  LDA WKANAL       ;LISTENER ABSTELLEN
3BA6          0980    ORA #$80         ;FLAG SETZEN
3BA8          8D0401  STA WKANAL
3BAB          A93F    LDA #UNLSN
3BAD          D0E5    BNE ATLO         ;JMP
3BAF
3BAF ATNLO    48      PHA              ;RETTE AKKU
3BB0          AD0CA0  LDA PCR
3BB3          29FD    AND #%11111101
3BB5          8D0CA0  STA PCR          ;ATN=LOW
3BB8          68      PLA
3BB9          60      RTS
3BBA
3BBA LISTEN   2C0401  BIT WKANAL       ;KANAL OFFEN?
3BBD          7006    BVS LIST
3BBF          0E0401  ASL WKANAL       ;JA
3BC2          4E0401  LSR WKANAL       ;FLAG LOESCHEN
3BC5 LIST     AA      TAX              ;SEK.ADR
3BC6          A928    LDA #LSN         ;PRIM.ADR
3BC8 PSADR    20AF3B  JSR ATNLO        ;KANAL AKTIVIEREN, LISTENER
3BCB          20703C  JSR SEND         ;PRIM.ADR IN A
3BCE          8A      TXA              ;SEK.ADR IN X
3BCF          D0C6    BNE LASTKD
3BD1
3BD1 HEADER   8A      TXA              ;SENDE KDO.STRING
3BD2          09F0    ORA #OPEN        ;KANAL-# IN X
3BD4          20BA3B  JSR LISTEN
3BD7          A001    LDY #1
3BD9 HDR1     B93EA4  LDA STRING,Y
3BDC          20703C  JSR SEND
3BDF          C8      INY
3BE0          CC3EA4  CPY STRING
3BE3          D0F4    BNE HDR1
3BE5          20F23B  JSR EOILOW       ;EOI=LOW
3BE8          B93EA4  LDA STRING,Y
3BEB          20703C  JSR SEND
3BEE          20A33B  JSR UNLISN
3BF1          60      RTS
3BF2
3BF2 EOILOW   AD00A0  LDA DRB          ;EOI=LOW
3BF5          290E    AND #$0E
3BF7          8D00A0  STA DRB
3BFA          60      RTS
3BFB
3BFB TALKER   2C0501  BIT RKANAL       ;KANAL OFFEN?
3BFE          7006    BVS TALK
3C00          0E0501  ASL RKANAL       ;JA
3C03          4E0501  LSR RKANAL       ;FLAG LOESCHEN
3C06 TALK     A948    LDA #TLK         ;KANAL AKTIVIEREN, TALKER
3C08          20C83B  JSR PSADR
3C0B          A900    LDA #0
3C0D          8D03A0  STA DDRA         ;PORT A IST EINGANG
3C10          A90D    LDA #$0D
3C12          8D00A0  STA DRB          ;NDAC=LOW
3C15          60      RTS
3C16
3C16 GETTST   AD0501  LDA RKANAL       ;TALKER AUF BUS?
3C19          100B    BPL GETCHR
```

```
3C1B            297F    AND #$7F        ;FLAG LOESCHEN
3C1D            0960    ORA #$60        ;DATENKANAL-#
3C1F            AA      TAX             ;SEK.ADR
3C20            20A33B  JSR UNLISN      ;LISTENER ABSTELLEN
3C23            20FB3B  JSR TALKER      ;TALKER AKTIVIEREN
3C26 GETCHR     2C00A0  BIT DRB         ;LESE-ROUTINE
3C29            30FB    BMI GETCHR      ;WARTEN AUF DAV=LOW
3C2B            AD00A0  LDA DRB
3C2E            290B    AND #$0B        ;NRFD=LOW
3C30            8D00A0  STA DRB
3C33            AD01A0  LDA DRA         ;GUELTIGE DATEN LESEN
3C36            48      PHA             ;UND RETTEN
3C37            AD00A0  LDA DRB
3C3A            2910    AND #$10        ;CS, FALLS EOI=HIGH
3C3C            C910    CMP #$10        ;CC, FALLS EOI=LOW
3C3E            AD00A0  LDA DRB
3C41            0902    ORA #$02        ;NDAC=HIGH
3C43            8D00A0  STA DRB
3C46 GET1       2C00A0  BIT DRB
3C49            10FB    BPL GET1        ;DAV=HIGH?
3C4B            A90D    LDA #$0D
3C4D            8D00A0  STA DRB
3C50            68      PLA             ;HOLE DATEN
3C51            49FF    EOR #$FF        ;INVERTIERE DATEN
3C53            60      RTS

3C54 SNDTST     2C0501  BIT RKANAL      ;TALKER AKTIV?
3C57            3006    BMI SNDT1
3C59            48      PHA             ;JA, TALKER ABSTELLEN
3C5A            208A3B  JSR UNTALK
3C5D            D006    BNE SNDT2       ;JMP
3C5F SNDT1      2C0401  BIT WKANAL      ;LISTENER AKTIV?
3C62            100C    BPL SEND
3C64            48      PHA             ;NEIN
3C65 SNDT2      AD0401  LDA WKANAL      ;DATENKANAL-#
3C68            297F    AND #$7F        ;FLAG LOESCHEN
3C6A            0960    ORA #$60        ;SEK.ADR
3C6C            20BA3B  JSR LISTEN      ;LISTENER AKTIVIEREN
3C6F            68      PLA
3C70 SEND       49FF    EOR #$FF        ;INVERTIERE DATEN
3C72            8D01A0  STA DRA         ;SENDE DATEN, DAV=HIGH
3C75            A9FF    LDA #$FF
3C77            8D03A0  STA DDRA        ;PORT A IST AUSGANG
3C7A SND1       AD00A0  LDA DRB
3C7D            2960    AND #$60        ;NRFD=HIGH?
3C7F            C940    CMP #$40        ;UND
3C81            D0F7    BNE SND1        ;NDAC=LOW?
3C83            AD00A0  LDA DRB
3C86            2901    AND #$01        ;JA, DAV=LOW
3C88            0906    ORA #$06
3C8A            8D00A0  STA DRB
3C8D SND2       AD00A0  LDA DRB
3C90            2960    AND #$60        ;NDAC=HIGH?
3C92            C920    CMP #$20
3C94            D0F7    BNE SND2
3C96            A90F    LDA #$0F        ;JA, DAV=HIGH
3C98            8D00A0  STA DRB
3C9B            A9FF    LDA #$FF
```

```
3C9D            8D01A0 STA DRA          ;BUS NEUTRAL
3CA0            60     RTS

3CA1 SYERR      A024   LDY #MES6-MES1   ;SYNTAX-ERROR!
3CA3            20A93C JSR MESOUT
3CA6            A01D   LDY #MES5-MES1
3CA8            60     RTS
3CA9
3CA9 MESOUT     2013EA JSR CRLOW
3CAC MESCHR     B9B63C LDA MES1,Y       ;TEXT-AUSGABE
3CAF            C8     INY
3CB0            207AE9 JSR OUTPUT
3CB3            10F7   BPL MESCHR
3CB5            60     RTS
3CB6
3CB6            ;TEXTE: MES1..5 SIND FORMATGEBUNDEN
3CB6 MES1       5752   .BYT 'WRITE:',$A0
3CBC            A0
3CBD MES2       2052   .BYT ' READ:',$A0
3CC3            A0
3CC4 MES3       204B   .BYT ' KMDO:',$A0
3CCA            A0
3CCB MES4       4449   .BYT 'DIREC: ',$A4
3CD2            A4
3CD3 MES5       2020   .BYT '  NEW:',$A0
3CD9            A0
3CDA MES6       0D     .BYT $D,' SYNTAX-ERROR!',$8D
3CDB            2053
3CE9            8D
3CEA MES7       4449   .BYT 'DIREC=D '
3CF2            4B4F   .BYT 'KOMDO=K '
3CFA            4D4F   .BYT 'MONIT=M ',$BF
3D02            BF
3D03
3D03            ;SPRUNGVEKTOREN
3D03 TAB        DF39   .WOR REDPRG,WRIPRG,REDSEQ,WRISEQ
3D05            973A
3D07            EB39
3D09            AD3A
3D0B
3D0B IFC        20743B JSR NEUTRL       ;RESET INTERFACE
3D0E            A9DF   LDA #%11011111   ;IFC=LOW
3D10            2D0CA0 AND PCR
3D13            8D0CA0 STA PCR
3D16            A200   LDX #00
3D18 TIM        E8     INX              ;1,2MS IMPULS
3D19            D0FD   BNE TIM
3D1B            A920   LDA #%00100000   ;IFC=HIGH
3D1D            0D0CA0 ORA PCR
3D20            8D0CA0 STA PCR
3D23 NRES       AD00A0 LDA DRB
3D26            2960   AND #$60
3D28            F0F9   BEQ NRES         ;WARTEN AUF RESET
3D2A            60     RTS
3D2B
3D2B                   .END
3D2B             ERRORS= 0000
```

Dipl.-Ing. Hans W. Werner, 6800 Mannheim 1

Operanden-Mathematik

für den AIM-Assembler

Oft ärgert man sich darüber, daß der Assembler des AIM symbolische Berechnungen nicht ausführen kann, obwohl doch gerade bei Adressierungen in Feldern (z.B. Sprungtabellen) solche Berechnungen sehr wünschenswert sind.

Wie soll man schon die Anzahl von Elementen in einer Sprungtabelle als Symbol umschreiben, wenn nicht die Division durch 2 zur Verfügung steht? - Oder: Warum muß man immer die Endadresse eines Feldes durch ein zusätzliches Label kennzeichnen, um die Feldlänge zu bestimmen, wenn man dies durch die Multiplikation der Elemente-Anzahl mit der Elemente-Länge viel eleganter und verständlicher ausdrücken kann? - Oder wie leicht ließe sich der Program-Counter mit einem AND auf die nächste Page-Grenze setzen - oder das ASCII-Zeichen durch OR mit einem MSB=1 versehen?

Um diese, die Lesbarkeit und Änderungsfreundlichkeit hemmenden Eigenschaften des AIM-Assemblers zu beseitigen, wurden die nachfolgenden Routinen entwickelt, die eine erweiterte Operandenmathematik zulassen. Es wurde dabei auf eine kurze und den Anforderungen angepaßte Erweiterung Wert gelegt, so daß hier, entgegen der sonst üblichen Beschreibung der Vorzüge, eine Beschreibung der Besonderheiten gegeben werden soll:

- Wegen der wenigen zur Verfügung stehenden Sonderzeichen auf der AIM-Tastatur sind z.T. unübliche Operator-Symbole verwendet worden. Z.B kennzeichnet der '.' die Multiplikation, da das '*' für den Program-Counter reserviert ist.
- Da eine Aneinanderreihung von Operationen zulässig ist, ergibt sich für Multiplikation und Division eine Beschränkung auf positive Zahlen (0 ... 32737); eine nachfolgende Subtraktion oder Addition würde größere Zahlen als negativ ansehen, und außerdem würde eine Division vorzeichenbehafteter Operanden ungleich komplizierter ausfallen. Und: Sind Divisionen mit negativen Zahlen denn wirklich nötig?
- Die Operatoren werden ohne Berücksichtigung sonst üblicher Prioritäten in der Reihenfolge ihres Auftretens bearbeitet.
- Der AIM-Assembler bietet die (sicher ungewollte) Möglichkeit, fehler-ignorierende Operanden-Mathematik in EQU-Anweisungen zu betreiben. Somit ist z.B. eine Division mit Rest ungleich 0 möglich, bekannt als Integer-Funktion in BASIC. Vorsichtshalber sollten die Ergebnisse aber kontrolliert werden.

Es stehen somit also folgende Operationen mit jeweils 15 Datenbits zur Verfügung:

- Addition (+ VZ)
- Subtraktion (+ VZ)
- Multiplikation (nur positiv)
- Division (nur positiv)
- logisch AND (+ VZ)
- logisch OR (+ VZ)
- logisch EXOR (+ VZ)

Literaturhinweis: K.-G. Meyer, Arithmetik für Prozeßsteuerungen, MC 2/83.

```
0000          ;-------------------------------------------------------
0000          ;
0000          ;          Operanden-Mathematik für AIM-Assembler
0000          ;          ======================================
0000          ;
0000          ;  Date :   06.08.1983
0000          ;
0000          ;  File :   ASSMATHE.SRC          Version : 1.1
0000          ;
0000          ;  Copyright :
0000          ;     Hans W. Werner, 6800 Mannheim 1, Heinrich-Lanz-Str. 13
0000          ;
0000          ;-------------------------------------------------------


0000          ; Kurze Funktionsbeschreibung:
0000          ;
0000          ; Die nachfolgenden Routinen erweitern den Assembler des
0000          ; AIM 65  um einige Verknüpfungsmöglichkeiten für Operanden.
0000          ;     So stehen nun außer Addition und Subtraktion auch zur
0000          ; Verfügung:
0000          ;       - Division
0000          ;       - Multiplikation
0000          ;       - logisch UND       ( AND)
0000          ;       - logisch ODER      (  OR)
0000          ;       - log. Antivalenz   (EXOR)
0000          ;
0000          ; Die Operatoren werden grundsätzlich in der Reihenfolge
0000          ; ihres Auftretens bearbeitet, also ohne Prioritäten, so
0000          ; daß eine Klammerung nicht notwendig (und auch nicht
0000          ; zulässig) ist.


0000          ; EQUATE-Vereinbarungen
0000
0000          MINUS  = $2D          ; '-'  Subtraktion
0000          PLUS   = $2B          ; '+'  Addition
0000          MUL    = $2E          ; '.'  Multiplikation
0000          DIV    = $2F          ; '/'  Division
0000          LOGAND = $26          ; '&'  AND
0000          LOGOR  = $3A          ; ':'  OR
0000          LOGXOR = $21          ; '!'  EXOR
0000
0000          ; Page-Zero-Adressen  (v. Assembler festgelegt)
0000
0000          KNVAL  = $13          ; Value des letzten aktuellen Operanden
0000          IEXP   = $27          ; Ergebnis (vorhergehender Operationen)
0000          REST   = $82          ; Hilfsspeicher
0000
0000          ; Adressen im Original-Assembler
0000
0000          ZAPF1  = $D7A8        ; Erkennen eines Operators
0000          ZAPF2  = $D882        ; vom Operator abhängige Verzweigung
0000          ERROR  = $D832        ; unzulässiger Operator, Fehler
0000          OVFLOW = $D86B        ; Ergebnis-Überlauf, Fehler
0000          POSRES = $D872        ; positives Ergebnis
0000          NEGRES = $D879        ; negatives Ergebnis
```

Änderungen im Original-Assembler

```
0000
0000                    * = ZAPF1
D7A8  20007F            JSR XTENS1
D7AB  EA                NOP
D7AC
D7AC                    * = ZAPF2
D882  4C1B7F            JMP XTENS2
D885  EA                NOP
D886            ; Subtraktion
D886            SUBTR
D886
```

Erweiterungen

```
D886
D886                    * = $7F00
7F00            ; gültigen Operator erkennen
7F00  C92D      XTENS1  CMP #MINUS
7F02  F016              BEQ XT1
7F04  C92F              CMP #DIV
7F06  F012              BEQ XT1
7F08  C92E              CMP #MUL
7F0A  F00E              BEQ XT1
7F0C  C926              CMP #LOGAND
7F0E  F00A              BEQ XT1
7F10  C93A              CMP #LOGOR
7F12  F006              BEQ XT1
7F14  C921              CMP #LOGXOR
7F16  F002              BEQ XT1
7F18  98                TYA
7F19  CA                DEX
7F1A  60        XT1     RTS

7F1B
7F1B            ; Operation ausführen
7F1B  C92D      XTENS2  CMP #MINUS
7F1D  F017              BEQ XTS1
7F1F  C92F              CMP #DIV
7F21  F016              BEQ XTS2
7F23  C92E              CMP #MUL
7F25  F015              BEQ XTS3
7F27  C926              CMP #LOGAND
7F29  F014              BEQ XTS4
7F2B  C93A              CMP #LOGOR
7F2D  F013              BEQ XTS5
7F2F  C921              CMP #LOGXOR
7F31  F012              BEQ XTS6
7F33  4C32D8            JMP ERROR       ;kein gültiger Operator dabei
7F36
7F36  4C86D8  XTS1      JMP SUBTR       ;Subtraktion (Original)
7F39  4C487F  XTS2      JMP DVSION      ;Division
7F3C  4C847F  XTS3      JMP MULTIP      ;Multiplikation
7F3F  4CCA7F  XTS4      JMP LGAND       ;log. UND
7F42  4CDB7F  XTS5      JMP LGOR        ;log. ODER
7F45  4CE87F  XTS6      JMP LEXOR       ;log. Antivalenz
```

Division

```
7F48            ; Bedeutung: Lowbyte,Highbyte
7F48            ; Divisor  :   IEXP ,  IEXP+1
7F48            ; Dividend :  KNVAL , KNVAL+1
7F48            ; Quotient :   IEXP ,  IEXP+1
7F48            ; Rest     :   REST ,  REST+1
7F48
7F48  A527      DVSION LDA IEXP
7F4A  0513             ORA KNVAL
7F4C  3079             BMI OVRFLO     ;ein Operand negativ, Fehler!
7F4E  8A               TXA            ;X-Reg. retten
7F4F  48               PHA
7F50  A010             LDY #16        ;16 Bits zu bearbeiten
7F52  A900             LDA #0         ;Divisionsrest initial.
7F54  8582             STA REST
7F56  8583             STA REST+1
7F58  2628      DIV1   ROL IEXP+1     ;Divisor links shiften
7F5A  2627             ROL IEXP       ;.. und MSB ..
7F5C  2683             ROL REST+1     ;.. ins Hilfsregister
7F5E  2682             ROL REST
7F60  38               SEC
7F61  A583             LDA REST+1     ;Subtraktion möglich ?
7F63  E514             SBC KNVAL+1
7F65  AA               TAX
7F66  A582             LDA REST
7F68  E513             SBC KNVAL
7F6A  9004             BCC DIV2       ;nicht möglich !
7F6C  8582             STA REST       ;möglich, Ergebnis ablegen
7F6E  8683             STX REST+1
7F70  88        DIV2   DEY            ;alle Bits bearbeitet ?
7F71  D0E5             BNE DIV1       ;nein, noch einmal
7F73  2628             ROL IEXP+1     ;Quotient zurecht schieben
7F75  2627             ROL IEXP
7F77  68               PLA            ;X-Register wiederherstellen
7F78  AA               TAX
7F79  A527             LDA IEXP
7F7B  2980             AND #$80       ;Vorzeichen kontrollieren
7F7D  0582             ORA REST       ;.. und keinen Divisionsrest
7F7F  0583             ORA REST+1     ;.. zulassen
7F81  4CC27F           JMP WEITER
7F84
```

Multiplikation

```
7F84            ; Multiplikand  :   IEXP ,  IEXP+1
7F84            ; Multiplikator :  KNVAL , KNVAL+1
7F84            ; Ergebnis low  :   REST ,  REST+1
7F84            ; Ergebnis high :  KNVAL , KNVAL+1
7F84
7F84  A527      MULTIP LDA IEXP
7F86  0513             ORA KNVAL
7F88  303D             BMI OVRFLO     ;ein Operand negativ, Fehler!
7F8A  A900             LDA #0         ;Ergebnis-Low-Teil initial.
7F8C  8582             STA REST
7F8E  8583             STA REST+1
7F90  A010             LDY #16        ;16 Bits zu bearbeiten
```

```
7F92  0683    MUL1   ASL REST+1     ;Ergebnis und ..
7F94  2682           ROL REST
7F96  2614           ROL KNVAL+1    ;.. Multiplikator schieben
7F98  2613           ROL KNVAL
7F9A  9015           BCC MUL2       ;MSB=0, Addition unnötig
7F9C  18             CLC
7F9D  A583           LDA REST+1
7F9F  6528           ADC IEXP+1     ;Multiplikand ..
7FA1  8583           STA REST+1
7FA3  A582           LDA REST
7FA5  6527           ADC IEXP       ;.. zum Ergebnis addieren
7FA7  8582           STA REST
7FA9  9006           BCC MUL2
7FAB  E614           INC KNVAL+1    ;Übertrag erfassen
7FAD  D002           BNE MUL2
7FAF  E613           INC KNVAL
7FB1  88      MUL2   DEY            ;alle Bits bearbeitet ?
7FB2  D0DE           BNE MUL1       ;nein
7FB4          ; Ergebnis auswerten
7FB4  A583           LDA REST+1     ;Ergebnis in richtiges
7FB6  8528           STA IEXP+1     ;Register laden (f. Assembler)
7FB8  A582           LDA REST
7FBA  8527           STA IEXP
7FBC  2980           AND #$80       ;Ergebnis-VZ kontrollieren
7FBE  0513           ORA KNVAL      ;.. und nur 16-bit-Ergebnis zulassen
7FC0  0514           ORA KNVAL+1
7FC2  D003    WEITER BNE OVRFLO     ;Ergebnis zu groß !
7FC4  4C72D8  POSTIV JMP POSRES     ;Ergebnis kleiner $8000
7FC7  4C6BD8  OVRFLO JMP OVFLOW     ;Fehler !
7FCA
```

Logische Verknüpfungen AND, OR, XOR

```
7FCA          ; 1. Operand :  IEXP ,  IEXP+1
7FCA          ; 2. Operand : KNVAL , KNVAL+1
7FCA          ; Ergebnis   :  IEXP ,  IEXP+1
7FCA
7FCA  A528    LGAND  LDA IEXP+1     ; AND
7FCC  2514           AND KNVAL+1
7FCE  8528           STA IEXP+1
7FD0  A527           LDA IEXP
7FD2  2513           AND KNVAL
7FD4  8527    LOGAN1 STA IEXP
7FD6  10EC           BPL POSTIV
7FD8  4C79D8  NEGTIV JMP NEGRES     ;negatives Ergebnis (größer $8000)
7FDB
7FDB  A528    LGOR   LDA IEXP+1     ; OR
7FDD  0514           ORA KNVAL+1
7FDF  8528           STA IEXP+1
7FE1  A527           LDA IEXP
7FE3  0513           ORA KNVAL
7FE5  4CD47F         JMP LOGAN1
7FE8  A528    LEXOR  LDA IEXP+1     ; XOR
7FEA  4514           EOR KNVAL+1
7FEC  8528           STA IEXP+1
7FEE  A527           LDA IEXP
7FF0  4513           EOR KNVAL
7FF2  4CD47F         JMP LOGAN1
7FF5                 .END
 ERRORS=0000
```

AIM Spezial (15)

Herr Peter Struß aus 2000 Wedel gibt folgende Hinweise zum Assembler: Die User-Eingabe ist leicht zu erreichen, wenn man den indirekten Sprung bei Adresse DFDA durch einen direkten Sprung zur Teilroutine WHERE! mit JMP E86C ersetzt. Hier wird das Carry 0 und damit die User-Routine angesprungen. Es ist empfehlenswert, im Quelltext ein .END zu verwenden, falls die User-Routine kein 00 als Textende sendet.

Um beim Assembler kleingeschriebene Label zuzulassen (z.B. in der Zusammenarbeit mit dem RE-MON), muß in der Zeichenerkennungsroutine unter Adresse D762 das geändert werden.

Bei der Ausgabe von Steuerzeichen über den DILINK-Vektor mit dem PRINT-Befehl des BASIC wird Linefeed (hex 0A) unterdrückt.

Zum PASCAL des AIM gibt Herr Dipl.-Ing. Hans Werner aus Mannheim folgende Hinweise:

Arrays dürfen nur weniger als 256 Elemente haben.
Konstanten-Definitionen werden, falls als Integer-Größe darstellbar, entsprechend gewandelt, z.B. 65536 zu 0 und 32768 zu –32768.
Ein Kommentar zwischen dem letzten Befehl eines THEN-Blockes und dem ELSE ist unzulässig und führt ggfs. dazu, daß weder THEN noch ELSE ausgeführt werden.

Die 'blödsinnige' Zeilenlängen-Begrenzung auf 60 Zeichen hat viele Nachteile wegen der nur beschränkten Übergabemöglichkeit von Parametern an PROCEDUREs und FUNCTIONs, Aufzählungstyp-Deklarationen sind nur bedingt einsetzbar, Kommentare sind in der Länge begrenzt. Dieses Hemmnis wurde recht einfach wie folgt beseitigt: Die gewünschte Zeilenlänge X (kleiner 256 Zeichen) ist an zwei Stellen im PASCAL verwendet worden, im Byte bei hex 5C71 (Wert X-1) und in B1C6 (Wert X). Bei einer Änderung diesen beiden Bytes kann man die erwähnten Einschränkungen stark herabsetzen. Es ist zu ergänzen: Bei der Umwandlung des Einlesetextes in den PASCAL-Zwischencode steht ein Buffer von 256 Byte zur Verfügung, der bei Überlauf (z.B. zuviele Aufzählungselemente) eine Fehlermeldung ausgibt, denn der Zwischencode kann länger als die Source werden. Vielen Dank!

Herr Andreas Zilker, Schoeplergasse 20 b, 8300 Landshut 1, bittet die Leser um Hinweise, ob sie in der Zusammenarbeit des AIM mit dem NEC-Drucker 8023 ähnliche Beobachtungen gemacht haben und wo die Ursachen liegen: Der Effekt: Bei der Ausgabe einer formatierten Assembler-Liste werden Blanks verschluckt, die auf den Hexcode von Ein-Bytebefehlen folgen, die also an der Schreibposition des 2. und 3. Bytes erscheinen sollen. Dieser Effekt tritt bei direkter Ausgabe auf den Drucker auf, nicht jedoch, wenn die Zeile zuerst in einen Puffer transportiert und von dort aus in 'einem Rutsch' ausgegeben wird.

Im allerersten 'AIM Spezial' wurde bereits daraufaufmerksam gemacht, daß unter I-Kommando des Monitors der Befehl LDX abs,Y falsch ausführt. Herr Wolfgang Masarie aus 8346 Simbach schlägt die nebenstehende Änderung des Disassemblerformates zur Behebung vor. Es bedingt dann allerdings auch eine Änderung anderer Programme, z.B. des symbolischen Disassemblers aus MC 3/1983. Auch hier für vielen Dank!

```
 F4A2 20DBE3  JSR E3DB*
=F4BD
 F4BD F009    BEQ F4C8*
=E3DB
 E3DB 85EA    STA EA
 E3DD 98      TYA
 E3DE 48      PHA
 E3DF A0FF    LDY #FF
 E3E1 C8      INY
 E3E2 84D3    STY D3
 E3E4 2056EB  JSR EB56
 E3E7 2046EA  JSR EA46
 E3EA A4D3    LDY D3
 E3EC C4EA    CPY EA
 E3EE D0F1    BNE E3E1
 E3F0 C003    CPY #03
 E3F2 F006    BEQ E3FA
 E3F4 203BE8  JSR E83B
 E3F7 C8      INY
 E3F8 D0F6    BNE E3F0
 E3FA 68      PLA
 E3FB A8      TAY
 E3FC 60      RTS *
=FCE7
 FCE7 2057EE  JSR EE57*
=FD5B
 FD5B 4C7DFD  JMP FD7D*
=EE57
 EE57 C9BE    CMP #BE
 EE59 D004    BNE EE5F
 EE5B A902    LDA #02
 EE5D D003    BNE EE62
 EE5F BD0FFB  LDA FB0F,X
 EE62 60      RTS
```

Hans-Joachim Langermann, 1000 Berlin 10

AIM 65 DOS (Rockwell)

Das Disketten-Betriebssystem AIM 65 DOS Version 1.0. der Floppy Disk Controller-Karte aus der RM 65-Familie von Rockwell (s. 65xx MICRO MAG Nr. 30) enthält zwei offensichtliche Fehler:

1. Nach der Rückkehr aus der Backup-Routine in den Monitor enthält das Outflag ($A413) ein 'U', wodurch eine falsche Formatierung bei nachfolgenden Ausgaben auf Drucker oder Anzeige entsteht.

2. Die List-File-Routine überträgt bei der Ausgabe auf eine Diskette zwar das Datenfile, aber das Write-Disk-File wird beim Erkennen der End-of-File-Marke nicht abgeschlossen, so daß das Directory nicht auf den neuen Stand gebracht wird. So ist das kopierte File vom DOS aus nicht zugänglich.

Die Fehlerbehebung ist relativ einfach, da einige ROM-Bereiche vom DOS nicht genutzt werden ($8333 bis 8339 und 84BE bis 84CD enthalten 00). Eine zusätzliche Routine findet damit leicht Platz. Das Original-ROM muß allerdings mit den Korrekturen in ein EPROM übertragen werden. Folgende Änderungen sind vorzunehmen:

```
                *=$8337

8337  4CBE84  LISEND JMP LISEN2

                *=$8343

8343  F0F2          BEQ LISEND  ;EOF FOUND

                *=$84BE

84BE  AD13A4  LISEN2 LDA OUTFLG
84C1  C955           CMP #'U'    ;OUT=U=DISK ?
84C3  D003           BNE LISRET  ;NO, RETURN
84C5  4CAD82         JMP CLSOUT  ;YES, CLOSE WRITE DISK FILE
84C8  60      LISRET RTS
```

Die folgenden Befehle setzen das Outflag in der Backup-Routine nach der Übertragung eines Files zurück. An dieser Stelle befand sich vorher ein Aufruf von 'CLSOUT', der jetzt überflüssig ist, da die Backup-Routine das Ausgabeprogramm des List-Befehls mitbenutzt.

```
                *=$8433

8433  A90D          LDA #$0D    ;SET OUTFLAG TO D/P
8435  8D13A4        STA OUTFLG
8438  EA            NOP
```

□

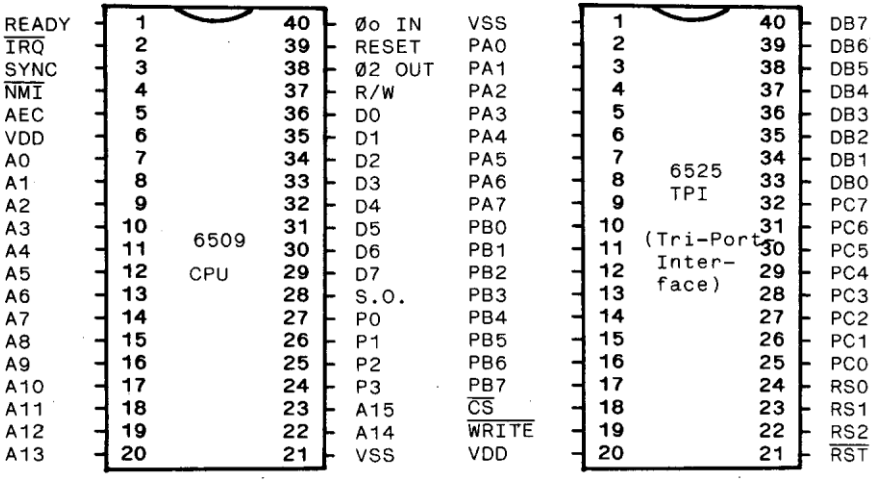
Commodore Chips

Seit der Einführung der Volks-Computer vor etwa zwei Jahren sind auch neue Zentraleinheiten und Interface-Chips in den Computern enthalten, über die z.T. wenig bekannt war. Wie es scheint, ist nicht einmal die deutsche Commodore im Besitz von Original-Datenblättern. Wenn man einen guten Draht dahin hat, erhält man Fotokopien von Fotokopien von (..) Datenblättern. Sie sind manchmal nur noch undeutlich lesbar. Für ein breiteres Publikum daher die folgende Übersicht über die Hauptmerkmale wichtiger Chips (ohne Gewähr):

CPU 6509

Hier liegen nur die ersten beiden Seiten eines Datenblattes vor. Ihnen ist das abgebildete Pinning zu entnehmen. Es gibt sie offensichtlich für 1, 2 und 3 MHz. Im Commodore 710 ist die 2 MHz-Version in Gebrauch. Weiterhin heißt es: Takterzeugung auf dem Chip, 5V Versorgungsspannung, Memory Management bis 1 MB und Direct Memory Access (DMA).

Die 56 Befehle und 13 Adressierungsarten entsprechen denen der CPU 6502. Für und wegen des Memory Management treten folgende Besonderheiten hinzu: Die sonst in der Zero Page liegenden Adressen 0 und 1 werden als zusätzliche Register der CPU dekodiert, und zwar die Adresse 0 als 'Execution Register' und die Zelle 1 als 'Indirection Register'. Was heißt das nun? Wenn die CPU eine Instruktion aus dem Speicher holt, dann sendet sie den Inhalt des Execution Registers auf die vier Pins P0 bis P3. Sie ist damit eine Maschine mit 20 Adreßbus-Pins und kann so 16 Segmente oder Banks von jeweils 64 KB ansprechen. In jedem dieser Speichersegmente findet sie natürlich wieder die Zellen 0 und 1 (on Chip) sowie eine Zeropage, eine Stack-Page und den übrigen Speicher. Beim power up und beim Reset sendet sie auf den Pins P0 bis P3 'High', so daß dann die Bank hex 'F' bzw. dez. 15 adressiert wird.



Das Umschalten der Programmbank geschieht durch die CPU selber, und zwar mit einem der Befehle STA 00, STX 00 oder STY 00, Abspeicherung also in das Register 0. Und wenige Zyklen später ist die CPU aus der Programmbank verschwunden, aus der sie bisher ihre Befehle entnahm. Sie holt den nächsten Befehl aus der jetzt eingestellten Bank, und zwar an der Folgeadresse, die dem STA-Befehl folgt. Soweit ist die Angelegenheit klar und ungefährlich, außer wenn im Interruptbetrieb gearbeitet wird, wie es bei Commodore für die Abfrage der Tastatur üblich ist. Dann

nämlich muß der Interrupt in der Datenbank abgefertigt werden, in der die Festwertspeicher und Interfacebausteine liegen (Bank 15). In der neuen Bank muß der Interrupt also einen Vektor finden, der ihn mit einer Routine zur Bank 15 zurückführt. Nun tritt aber folgende Schwierigkeit auf: Auf dem Stack der neuen Bank liegen wegen des Interruptes die Return-Adresse und der letzte Status. Der Befehl RTI (Return from Interrupt) liegt ohne geeignete Stack-Werte in Bank 15. Und dort sollte auch bekannt sein, aus welcher Bank der Aussprung zur Interruptbehandlung erfolgte. Die Lösung liegt im Parametertransport von Bank zu Bank, und hier spielt das Indirection Register (Adresse 1) die entscheidende Rolle (s.u.).

Die 16 Adreßbuspins werden als Three-State- und DMA-fähig bezeichnet. Zu welchen Zeiten der Zustand Three-State eintritt, ist nicht bekannt. Es wird aber vermutet, daß der Pin AEC so etwas bedeutet wie 'Adress Enable Control'. Man müßte gelegentlich im 710 die Leiterbahnen verfolgen und nachschauen, ob er ggfs. im Zusammenhang mit dem Eingangssignal READY den Prozessor anhält und den hochohmigen Zustand für DMA erzeugt.

Nun zur Paramterübergabe von Bank zu Bank mit dem Indirection Register (1): Alle Adressierungsarten, bis auf eine, holen und speichern Daten mit LDA, STA usw. in der Bank, die in Zelle 0 eingestellt ist, in der aktuellen Programmbank also. Nur die indirekte Zero Page-Adressierung mit Y, also LDA (Pointer),Y, wirkt auf die Datenbank, die im Indirection Register eingestellt ist. Und es kann in der Zero Page jeder beliebige Pointer benutzt werden (außer in 0/1). Diese Adressierungsart hat also den totalen Daten-Durchgriff auf maximal 16 Bänke. Wie nun der Parametertransport im Zusammenhang mit dem Interrupt und zwecks Ausführung von Unterprogrammen in anderen Bänken benutzt wird, dazu weitere Ausführungen in diesem Heft beim CBM 710.

6525 Tri-Port Interface

Der TPI hat 24 individuell Programmierbare I/O Pins Ports A, B und C) oder 16 I/O Pin und an Port C 2 Handshake-Leitungen und 5 Interrupteingänge mit Prioritätssteuerung. Nach dem Datenblatt gibt es den TPI für 1, 2 und 3 MHz-Betrieb. Über RS0 bis RS3 werden 8 Register auf dem Chip angesprochen, es sind dies (Rx wobei x relativ zur Grundadresse des TPI):

- R0 PRA - Port Register A
- R1 PRB - Port Register B
- R2 PRC - Port Register C
- R3 DDRA - Datenrichtungsregister A
- R4 DDBR - Datenrichtungsregister B
- R5 DDRC - Datenrichtunsregister C/ Interrupt Maskenregister
- R6 CR - Control Register
- R7 AIR - Active Interrupt Register

Das Setzen der Datenrichtungsregister und Ports geschieht in der üblichen Weise: 1 im Bit der Datenrichtung = Ausgang, 1 im Port = High Level et vice versa. Interessant sind danach vor allem die Register CR und AIR.

7	6	5	4	3	2	1	0	Bit
CB1	CB0	CA1	CA0	IE4	IE3	IP	MC	CR= Kontrollregister
			A4	A3	A2	A1	A0	AIR= Active Interrupt Register
			M4	M3	M2	M1	M0	DDRC=Maskenregister, wenn MC=1
CB	CA	$\overline{IRQ}$	I4	I3	I2	I1	I0	Port C= Sonderfunktion, wenn MC=1

Abhängig vom Bit MC im Kontrollregister wird die Rolle des Datenrichtungsregisters und auch des Portes C gesteuert. Mit MC=0 spielen sie die übliche Port-Rolle. Besonderheiten treten also mit MC=1 ein. Dann können die Pins PC7 und PC6 als Handshake-Leitungen benutzt werden, die mit CB bzw. mit CA bezeichnet werden. Ihre Funktion im einzelnen wird iebenfalls im Kontrollregis-

ter gesteuert, und zwar in den korrespondierenden Bits 7 und 6 für den Pin CB und in den Bits 5 und 4 für Pin CA. Hierfür gelten folgende Steuerungsmöglichkeiten:

Bit CA1	Bit CA2	Modus	Beschreibung
0	0	Handshake beim Lesen	CA wird HI bei aktivem Übergang des I3 Interrupteingangs (s.u.) und wird LO bei einer Leseoperation des Prozessors 'Read A Data'. Das ermöglicht Kontrolle von Datentransfers von der Peripherie zum Prozessor hin.
0	1	Pulsausgabe	CA geht für 1 ms auf LO beim Lesen von Port A.
1	0	Manuell	CA wird LO gesteuert.
1	1	Manuell	CA wird HI gesteuert.
CB1	**CB2**		
0	0	Handshake beim Schreiben	CB wird LO mit dem Schreiben nach Port B und geht auf HI zurück, wenn an I4 eine aktive Flanke auftritt: Verständigung der daß Daten bereitstehen.
0	1	Pulsausgabe	CB geht für 1 ms nach Beschreiben des Port B auf LO.
1	0	Manuell	CB wird LO gesetzt.
1	1	Manuell	CB wird HI gesetzt.

Wenn im Kontrollregister-Bit MC=1, dann ist nicht nur eine Handshake-Kontrolle für CB und CA möglich, sondern auch eine Interruptsteuerung. Dann können die Pins PC0 bis PC4 als Eingänge für Interruptsignale benutzt werden. Ein aktiver zugelassener Interrupt zieht dann Pin PC5 prozessorseitig (entsprechend zu beschalten) auf LOW. Das Datenrichtungsregister dient jetzt als Maske für die Zulassung zum Interrupt. Jede 1 in DDRC läßt einen Interrupt vom korrespondierenden PC-Pin zu. Für die Interrupteingänge I4 und I3 gibt es im Kontrollregister CR eine Steuerungsmöglichkeit in IE4 und IE3 für die Impulsflanke, die zum Interrupt führt: Bit ist 0 für negative Flanke und =1 für positive Flanke. I2 bis I0 können nur auf abfallende Flanken reagieren.

Die Batterie der I4 bis I0 wird als Interrupt Latch Register bezeichnet. Aktive (zugelassene) Interrupte propagieren sofort zum AIR (Active Interrupt Register). Wenn IP=0 (keine Prioritäten) und wenn AIR zwecks Feststellung der Quelle gelesen wird, dann wird auf jeden Fall das Bit Ix dort gelöscht. Wenn nun vor dem Lesen des AIR ein anderer Interrupt hinterherschoß, so geht dessen Bit im AIR beim Lesen desselben verloren.

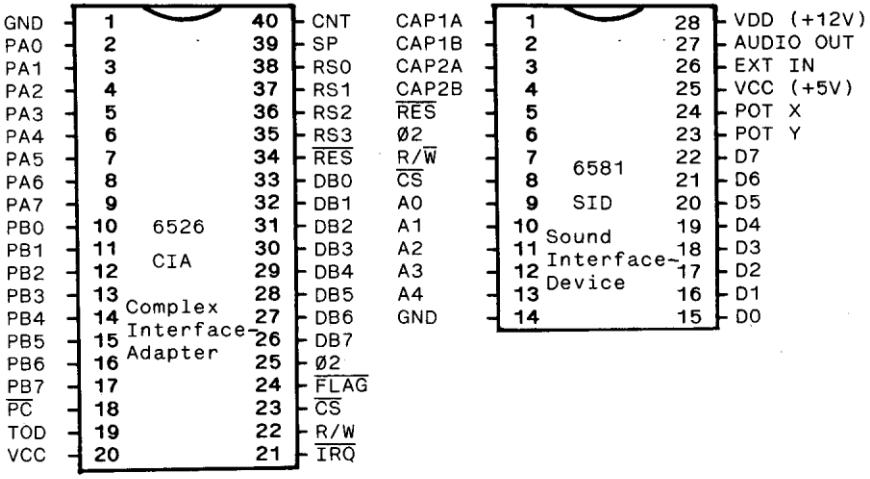
Interrupte können prioritisiert werden, wenn IP=1 im Kontrollregister. Dann rangiert I4 vor I3 usw.. Prioritäten sind also durch äußere Beschaltung herzustellen. Im Zusammenhang mit Interrupten hat das Register AIR im Zusammenhang mit einem hinter ihm stehenden Stapel besondere Bedeutung. Zunächst ist zu sagen, daß beim Lesen des AIR der IRQ-Ausgang zu HIGH zurückgenommen wird. Auch ist klar, daß Interrupte mit geringerer Priorität eine Interruptroutine mit einer höheren nicht unterbrechen können. Solange ein höherer Interrupt in Bearbeitung ist, wird kein niederes Ix-Bit in das AIR propagiert.

Wichtig ist nun: Beim Lesen des AIR wird sein Inhalt auf einen internen Stack geworfen. Wenn der Prozessor nach Feststellung der Interruptquelle und ihrer Abarbeitung fertig ist, muß er einen Schreibvorgang in das AIR vornehmen. Dadurch geht nichts kaputt, vielmehr das letzte Item vom Stapel geholt, wobei das Bit des zuletzt abgearbeiteten Interrupts automatisch gelöscht wird. Wenn der AIR-Status jetzt noch ein aktives Bit hat, so geht die IRQ-Leitung wieder solange auf LOW, bis AIR gelesen wird. Natürlich kann nach dem Entstacken wieder ein gelatchtes Bit von Ix nach AIR gelangen. Die klare Beschreibung dieser sinnvollen Automatik fiel nicht ganz leicht. Der Autor hofft, daß sie gleichwohl durchsichtig geworden ist.

Ein unnötiges Lesen und Beschreiben des AIR ist daher zu unterlassen.

6526 Complex Interface Adapter (CIA)

Dieser Baustein kann in mancher Weise mit der VIA 6522 verglichen werden. Er hat die Ports A und B mit den davorliegenden Datenrichtungsregistern, 2 unabhängige Timer mit jeweils 16 Bit und mit gleich breitem Vorspeicher (Latch) und ein Register zur seriellen Datenübermittlung (SDR), das wieder mit der Übermittlung des höchstwertigen Datenbits beginnt. Den Port-Bits PB7 und PB6 kann im Zusammenhang mit Timer B bzw. A wiederum eine Sonderfunktion zur Impulsausgabe zugewiesen werden. Als Besonderheit treten Register für eine Echtzeituhr hinzu, die im BCD-Format geführt wird (TOD=Time of Day). Und: Einige Register haben beim Lesen eine andere Funktion als beim Schreiben. Wir haben folgendes Pinout und folgende Register:



R0 Port A
R1 Port B
R2 DDRA, Datenrichtungsregister
R3 DDRB, Datenrichtungsregister
R4 TAL, Timer A LOW, Counter bei Read und Latch bei Write
R5 TAH, Timer A HIGH, dito
R6 TBL, Timer B LOW, dito
R7 TBH, Timer B HIGH, dito
R8 TOD, Zehntelsekunden in den Bits 3-0, alle TOD-Register im BCD-Format
R9 TOD mit Einersekunden in Bits 3-0, Zehnersekunden in Bits 7-4
RA TOD, Einerminuten in Bits 3-0, Zehnerminuten in Bits 7-4
RB TOD, Einerstunden in Bits 3-0, Zehnerstunden in Bits 7-4
RC SDR, Serial Data Register, arbeitet mit Pin SP als E/A zusammen
RD ICR, Interrupt Control Register. Read: Bestimmung der Quelle, Write. Setze Maske
RE CRA, Control Register für Timer A, SP-Funktion und TOD-Trigger 50/60 Hz
RF CBR, Control Register für Timer B, Alarm, Uhrzeit setzen, Taktquelle

Datenrichtungsregister und Portpins werden in der üblichen Art (wie z.B. bei der VIA) bedient. Zum seriellen Port SDR: Das Kontrollbit SP im Register CRA bestimmt den Modus. Wenn SP=0, dann ist der Pin SP Eingabe, es wird an Pin CNT ein externer Takt benötigt, um das Shiften im SDR zu ermöglichen (steigende Flanke an CNT, die offensichtlich synchron zu Ø2 sein muß). Wenn SP=1, dann ist der Pin SP Ausgang, und CNT gibt das Clocksignal ab, vorausgesetzt, daß Timer A nonstop läuft und damit auf CNT arbeitet. Die Übertragung beginnt mit dem Beschrei-

7	6	5	4	3	2	1	0	Bit
IR	0	0	Flag	SP	Alarm	TB	TA	RD = ICR beim Lesen
S/C	x	x	Flag	SP	Alarm	TB	TA	RD = ICR beim Schreiben

TODIN	SPM	INM	Load	RUNM	OUTM	PBON	Start	RE = CRA, Control Register A
TOD/A	INM6	INM5	Load	RUNM	OUTM	PBON	START	RF = CRB, Control Register B

ben des SDR, wobei ein Byte gebuffert werden kann. Weil die Übertragung unüblich mit dem höchstwertigen Bit beginnt und weil es keine Start- und Stopbits gibt, ist die serielle Übertragung dieser Art nur zwischen gleichartigen Bausteinen in Betracht zu ziehen.

Die Timer werden durch die gleichartigen Kontrollregister CRA und CRB gesteuert. Mit Start=1 startet der Timer, mit =0 kann er jederzeit angehalten werden. Nach einem One-Shot (s.u.) ist dieses Bit automatisch 0. PBON=0: Normale Timerfunktion ohne Wirkung auf PB7 bzw. PB6. PBON=1: Der Timerdurchgang durch 0 wird an den PB-Pin geleitet. OUTM=0: Pulsausgabe, OUTM=1: Signalumkehr am PB-Pin. RUNM=0: Nonstop-Betrieb, RUNM=1: One-Shot-Betrieb (Monoflop). Load=1: Forciertes Laden des Timers. Beim Lesen wird man das Load-Bit zu 0 finden. Das Bescheiben mit 0 hat keine weitere Wirkung. INM=0: Der Timer liest Ø2-Pulse, INM=1: Er liest aufsteigende Flanken am Pin CNT (Ereigniszähler, serielle Datenübertragung). SPM wurde bereits abgehandelt. TODIN=0: 60 Hz-Takt an CNT, TODIN=1: 50 Hz (Takt für die Echtzeituhr, extern vom Netz gesteuert).

Beim Timer B gibt es folgende Besonderheiten: Mit TOD/A=0: Setzen der Echtzeituhr, mit =1 Setzen/Voreinstellen einer Alarmzeit. Die Bits INM6 und INM5 haben in der Kombination folgende Bedeutung: 00= Timer B zählt Ø2-Pulse. 01= Timer B zählt positive CNT-Durchgänge. 10= Timer B ist Untersetzungszähler für Timer A (lange Zeiträume sind vorprogrammierbar). 11= wie 10, nur daß die Nulldurchgänge von Timer A nur registriert werden, wenn Pin CNT=1. Die Register TAL, TAH, TBL und TBH der Timer sind beim Schreiben Zwischenspeicher (Latches), aus denen beim Start mit CRA/CRB-Bit in die Zähler durchgeladen wird oder aus denen im Nonstop-Betrieb nachgeladen wird (wie bei der VIA). Beim Lesen erhält man aus diesen Registern dagegen den aktuellen Zählerstand.

Die Echtzeituhr TOD nebst Alarmvoreinstellung: Sie wird im BCD-Format geführt und kann wahlweise vom Systemtakt oder von einer externen 50/60 Hz-Quelle gesteuert werden. Auch hier gibt es einen Zwischenspeicher, um die Zeitangabe konstant zu halten. Alle 4 TOD-Register werden beim Lesen der Stunde gelatcht, und zwar solange, bis die 1/10 Sekunden gelesen sind. Die Uhr läuft dabei weiter. Ohne dieses Latchen können andere Register 'on the fly' gelesen werden. Das Setzen der Zeit läuft analog: Das Beschreiben des Stundenregisters hält die Uhr an. Sie startet exakt und zur gewünschten Zeit mit dem Beschreiben der 1/10 Sekunden. Das Alarmregister ist Interruptfähig.

Interrupt-Kontrolle mit dem ICR: Es gibt 5 zulässige Interruptquellen, Timer A und B-Durchgänge, Alarmzeit ist eingetreten, das serielle Register muß nach 8 Shifts bedient werden, am Pin FLAG ist eine negative Flanke aufgetreten. Beim Lesen spiegelt das ICR den Interruptstatus wieder. Sobald auch nur 1 Bit besetzt ist, ist auch IR gesetzt (für den BIT-Befehl). Mit dem Lesen werden allerdings auch alle Bits zurückgesetzt, so daß es in der Verantwortung des Programmierers steht, Information ggfs. zwischenzuspeichern. Mit dem Beschreiben wird wie bei der VIA eine Maske gesetzt. Jede logische 1 führt an entsprechender Stelle zum Interrupt Enable. Dabei hat es folgende Besonderheit: Wenn S/C=0, dann wird das entsprechende Maskenbit zurückgesetzt (weg geANDed). Wenn S/C=1, dann wird ein Bit an entsprechende Stelle hinzu geODERt.

Zu den Pins und Funktionen des 6526 CIA sind einige Nachträge angebracht, für die unglücklicherweise Seiten in den hier vorliegenden Fotokopien fehlten (daher ohne Gewähr): IRQ ist ein Ausgang mit aktiv Low. SP ist Ein- oder Ausgang, je nach Richtung der eingestellten seriellen Datenübermittlung. PC ist nur Ausgang für Handshake. Er geht für eine Taktzeit auf Low, wenn an Port B ein Schreib- oder Lesevorgang erfolgte. Sofern auch Port A in eine dann 16 Bit breite Übermittlung einbezogen werden soll, muß erst Port A bedient werden, dann Port B. Das Gegenstück dazu ist der Eingang FLAG, der auf abfallende Flanke getriggert ist. An diesem Pin gibt die Peripherie ihren Handshake ab. - Der PIN 19 (TOD) dient zur Entgegennahme einer externen stabilen Frequenz zur Steuerung der Echtzeituhr.

Für die Sound Interface Device SID 6581 wird in diesem Heft nur das Pinout abgedruckt. Der Chip hat 29 Register, die eine breitere Darstellung erfordern.

R.L. □

CBM 710 Spezial (1)

Das kleine Handbuch zum Rechner enthält keine **Zuordnung der Anschlußleisten zu den Bezeichnungen auf der Platine.** Daher diese Aufstellung:

CN6, CN7	System Bus Connectors
CN3	Cassette (von der Firmware nicht unterstützt)
CN2	IEEE 488bus
CN1	RS 232-Schnittstelle
CN11	User Port
CN3	Keyboard
CN12	Co-Processor Connector
CN4	Cartridge Connector

Der Userport des hiesigen Rechners hatte zur großen Überraschung keine Verbindung mit Ground! Das ist ein Fehler im Platinen-Layout. Beim Betrieb des Ports wird zu beachten sein, daß ein Teil seiner Signalleitungen auch vom IEEE-Bus benutzt wird. Es sind die Datenleitungen des Busses an den Pins15-22 am Userport. Einerseits mag das den Vorteil haben, daß die Daten auch hier abgenommen werden könnten. Eine Steuerung sollte man hier aber nicht anschließen. Beschaltet sind sie mit den Pins des Portes A der bei Adresse DC00 decodierten CIA. Ungeteilt hat man deren Port B, sowie -PC, CNT und SP der CIA sowie -FLAG, wenn (wie wohl meist) keine Datasette betrieben wird. Hinzu treten von dem bei DE00 decodierten Tri-Port-Interface 6525 die mit PRB2, PRB3 bezeichneten Portpins. Bei Pin 25 des CN11 , der auf den Belegungsplänen mit -IRQ bezeichnet wird, dürfte es sich in Wirklichkeit um den Pin PC5 desselben Tri-Ports handeln, der wahlweise als Interrupt-Ausgang betrieben werden kann (siehe in diesem Heft).

Seit einigen Wochen liegt hier nun **Das ROM-RAM-I/O Assemblerlisting für die Commodore-Serie 610 und 710** vor. Man muß zunächst einmal die Autoren R. Schineis und O. M. Braun dafür loben, das sie dieses nützliche Buch so schnell herausgebracht haben. Es hat fast 380 Seiten und enthält nach Zeilennummern geordnet einen (maschinell) kommentierten Quelltext, die Speicheradressen, und den Opcode. Weiterhin werden die Belegungen im RAM dokumentiert. Die beiden Teile für BASIC und Betriebssystem sind jeweils mit einer Cross-Referenz-Liste für die Symbole und Label versehen. Weiterhin wird dokumentiert, welche Interface-Pins zu welchen Anschlüssen führen. Ein nachkommentierter Quelltext kann natürlich nie so gut sein, wie ein Originaltext. Gleichwohl ist dieses Buch jedem zu empfehlen, der sich mit den maschinensprachlichen Möglichkeiten der Maschine beschäftigen will.

Zunächst einmal ist festzustellen, daß es am Ende des F-ROMs wieder eine (Standard?)-Sprungleiste gibt. Für alle wichtigen E/A-Routinen zeigt sie mit indirekten Sprüngen auf Adressen ab hex 300 im RAM. Der Benutzer ist damit in der Lage, hier die Vektoren für seine Zwecke um-

zuhängen.Die Routine Restore ab FBA2 transportiert 26 Standard-Vektoren sowie einen einen indirekten Sprungbefehl dorthin. Ruft man diese Routine bei FBA9 bei 'Vector' auf, und hat zuvor im Akku die Segmentnummer, in X die Adresse Low und in Y die Adresse High übergeben, die auf eine eigene Vektortafel zeigen, so besorgt dieses Unterprogramm den Transport der eigenen Vektortafel auf eine bequeme Weise.

Diese Maschine hat auch keine starre Kettung mehr an BASIC. Ab Hex 280 im RAM werden Vektoren indirekt angesprungen für Fehlerroutine, Token-Umwandlungsroutine, Tokenausgabe-Expanderroutine, Dispatcher, Bewertung, CHRGET und CHRGOT usw.. Damit ist es im Verlaufe leicht möglich, jede andere Sprache auf diesen Geräten zu interpretieren oder zu compilieren. In den ROMs sind sogar schon die Routinen für die Korrespondenz mit einem Co-Prozessor enthalten (Z80 oder 8088).Sie erfolgt über einen CIA-Baustein 6526, der als Zusatz bei Adresse DB00 decodiert wird. - Es ist auch möglich, das Gerät ohne BASIC-ROMs zu betreiben, wenn man auf die Adresse hex E000 springt (nach Reset). Man landet dann im Monitor.

In der Summe ist festzustellen, daß der CBM 710 mit ASCII-Tastatur benutzerfreundlich und weitsichtig konzipiert worden ist. Man muß nur hoffen, daß angesichts der bei Commodore üblichen schnellen Modellwechsel, eine Kontinuität für dieses Gerät besteht. Nur so werden sich Softwarehäuser und Betreiber seiner hervorragenden Möglichkeiten bedienen und dafür Software schreiben.

Für Bezieher des vorerwähnten Buches sei mitgeteilt, daß dort auf den Anfangsseiten erklärt wird, wie man eigene Assemblerprogramme zur Ausführung in einer anderen Bank bringt. Leider ist dort auf Seite 4, Zeile 0110 ein Druckfehler enthalten, der die Test-Demo behindert. Es muß dort heißen PLP mit Opcode hex 28.

Wir kommen damit auf die Programmausführung in den verschiedenen Speicher-BANKs. Der Leser möge zur Sicherheit im Artikel 'Commodore Chips' in diesem Heft blättern und sich die Besonderheiten der CPU 6509 vor Augen führen. Es bleibt kurz zu resümieren: Beim Einschalten ist der Computer in Bank 15. Das Execution Register für die Programmbank liegt in der Adresse 00 und für Befehle vom Typ LDA (POINTER),Y liegt die Indirection Bank in Speicherzelle 01 on chip.

Das Betriebssystem arbeitet grundsätzlich in dieser Bank 15. Hier befinden sich die Festwertspeicher und etwas RAM (Adressen 0-700 und D000-D7FF - Video) für die Bedürfnisse des Systems. Zu Tokens umgewandelter BASIC-Text wird bei der ASCII-Version dieses Rechners in der Bank 1 abgelegt und bei der Programmausführung von dort her mit der indirekten Adressierungsart gelesen, interpretiert und zur Ausführung gebracht. Variable werden in der Bank 2 angelegt, wobei deren oberster Teil den Text enthält, mit denen die Funktionstasten F1-F20 belegt sind. Das Betriebssystem und der Interpreter müssen also nie die Bank 15 verlassen, weil in den anderen Bänken nur Daten liegen, an denen gearbeitet wird. In diesem Zusammenhag wird auch klar, daß die deutsche Version dieses Rechners mit zusätzlichen Bänken für BASIC-Text und Variable ein anderes Betriebssystem für die Bankverwaltung benötigt.

Der Befehl BANK n dient dementsprechend nur zum Voreinstellen einer Speicherstelle, die bei der Ausführung der Befehle PEEK, POKE und SYS in das Indirection Register bzw. in das Execution Register übertragen wird. Man kann also tatsächlich mit BANK n und SYS xyz in ein maschinensprachliches Anwenderprogramm springen. Wenn man dabei durch Setzen des Status die Möglichkeit des IRQ-Interrupts ausgeschlossen hat, dann kann man schließlich aus der Bank n unbeschadet nach Bank 15 zurückkehren. Man hat den Nachteil, daß während der Durchführung des eigenen Programmes das E/A-System dann lahmgelegt ist. Diese Art der Abarbeitung kann daher eigentlich nur dann empfohlen werden, wenn E/A nicht benötigt wird und wenn es sich um sehr aufwendige Abarbeitungen handelt.

Das Dilemma des Interrupts, der ja in der Speicherbank ankommt, in der gerade ein Anwenderprogramm ausgeführt wird, kann auf verschiedene Weise angegangen werden. Im vorerwähnten Buch ist ein Weg aufgezeichnet, der dem Anwender auch aus einer von 15 verschiedenen Speicherbank her das E/A-System zur Verfügung stellt und die Verwendung jeden Unterprogrammes in der Systembank erlaubt.

Das Protokoll der Überleitung in eine andere Bank, vorzugsweise die Systembank 15, und für die Zurückführung in die aufrufende Bank ist etwas kompliziert. Zunächst ist zu bemerken, daß die Programmbank des Anwenders im obersten Speicherbereich mit Vektoren für IRQ, NMI und RESET zu versehen ist, ferner mit einer Sprungleiste, die derjenigen im F-Bereich des ROMs ähnelt. Diese Sprünge sind jedoch auf eine Routine des Anwenders gerichtet, die dort EXSUBF genannt wird und die mit indirekt indizierten Befehlen (mit Y indiziert) einen Parametertransport in die Zielbank vornimmt. Das Protokoll sieht weiterhin vor, daß der Stackpointer der jeweiligen Bank in der Zelle 01FF konserviert wird. So kann man aus jeder Bank heraus den Stack einer anderen Bank besichtigen und manipulieren. Dieser Stackpointer der anderen Bank wird auf weite Strecken in Y als ein künstlicher Stackpointer mitgeführt, um den fremden Stack zu beschicken. Voraussetzung für diese Stackmanipulationen ist nun, daß in der Zielbank ebenfalls ein Abbild der Routine für den Parametertransport vorhanden ist. Und da hat Commodore schon vorgesorgt. Wir finden dort ab Adresse FE9D die Ausführung des SYS-Befehles und ab FEB2 den Parametertransport.

Zunächst werden der Status und A, X, und Y auf den eigenen Stack gerettet. Dann wird der letzte Stackpointer des anderen Stacks nach Y übertragen. Dann wird die Nummer der aufrufenden Bank auf den anderen Stack gelegt. Dorthin wird auch die Rückkehradresse-1 einer Routine gepumpt, die EXCRTS genannt wurde und die später in die aufrufende Bank zurückleitet.

Nun wird auf dem eigenen Stack nachgeschaut, welche Routine denn eigentlich zur Ausführung gelangen sollte, als die Sprungleiste benutzt wurde. Die entsprechend manipulierte Adresse wird auf den anderen Stack gebracht. Es folgen dann die Registerinhalte aus der eigenen Bank. Es folgt die Rückkehradresse-1 der Routine EXPULL, die die 4 Register aus dem anderen Stack wiederherstellen soll. Dann wird der eigene Stack hergerichtet und der Stackpointer nach 01FF gebracht. Schließlich wird die Datenbank umgeschaltet. Und ganz zum Schluß dann die Execution bank. Die Programmfortsetzung mit RTS erfolgt dann bereits in der anderen Bank.

In der Zielbank werden zunächst die 4 Register vom Stack geholt, es wird die eigentlich gemeinte Routine ausgeführt (jeweils durch das RTS angesprungen), es folgt danach die Routine zur Rückleitung EXRCTS.

Der Vorgang ist ganz schön kompliziert, um ein Unterprogramm einer anderen Bank zu benutzen. Der Stack ist sicher eine neutrale Möglichkeit für die Parameterübergabe. Nach Ansicht des Autors sollte es etwas einfacher auch mit Briefkästen gehen und anderen Schnittstellen für die Program Execution von Bank zu Bank. Nachdem die Dinge aber so von Commodore vorgesehen sind, kann man sie halt so benutzen. Mit freundlicher Genehmigung durch Herrn Schneis wird aus vorerwähntem Buch nachstehend das mögliche Protokoll abgedruckt. Wer die Abläufe besser verstehen will, als es aus einer solchen verbalen und programm-mäßigen Schilderung möglich ist, dem sei empfohlen, das Protokoll an anderer Stelle im RAM im Einzelschrittbetrieb ablaufen zu lassen und jeweils die Stacks und Register zu beobachten. Unabhängig vom CBM 710 sind solche Protokolle in allen Fällen interessant, wo Speicherbänke samt Zero Page und Stack Page umgeschaltet werden.

```
-------------------------------------------------------------
CBM 710 BANKING-ROUTINEN NACH HARD + SOFT GMBH
0000            0001 ;R. LÖHR 3.9.83
0000            0002 ;FILE CBM710.03
-------------------------------------------------------------
SYMBOLE
0000            0004 E6509  =$0000          ;EXECUTION REGISTER
0000            0005 I6509  =$0001          ;INDIRECTION REG.
0000            0006 IPOINT =$AC            ;RAM INDIRECT POINTER
0000            0007 STACKP =$1FF           ;GESCHÜTZTE LAGE
0000            0008 EXSUB  =$FEB2          ;EXECUTE EXTERNAL SUBROUTINE
-------------------------------------------------------------
ANSPRUNG IRQ/BRK IM RAM SEGMENT
0000            0010        *=$FE64
FE64 85AC       0011 TXIRQ  STA IPOINT
FE66 68         0012        PLA             ;STACKPOINTER
FE67 48         0013        PHA
FE68 85AD       0014        STA IPOINT+1
FE6A A501       0015        LDA I6509
```

```
FE6C 48       0016          PHA               ;ALTES DATENSEGMENT
FE6D A5AD     0017          LDA IPOINT+1      ;STACKPOINTER IN AC
FE6F 2099FE   0018          JSR TXIRQ2        ;DIESE RTS-ADRESSE SPEICHERN
FE72 85AC     0019          STA IPOINT
FE74 68       0020          PLA
FE75 8501     0021          STA I6509         ;ALTES DATENSEGMENT
FE77 A5AC     0022          LDA IPOINT
FE79 40       0023          RTI
FE7A          0024
-------------------------------------------------------------------
ANSPRUNG NMI IM RAM-SEGMENT
FE7A 48       0026 TXNMI    PHA
FE7B A5AC     0027          LDA IPOINT
FE7D 48       0028          PHA
FE7E A5AD     0029          LDA IPOINT+1
FE80 48       0030          PHA
FE81 ADFF01   0031          LDA STACKP
FE84 48       0032          PHA
FE85 A501     0033          LDA I6509
FE87 48       0034          PHA               ;ENDE RETTEN
FE88 2035FF   0035          JSR EXNMI         ;AUFRUF NMI ALS SUBROUTINE IN BANK 15
FE8B 68       0036          PLA               ;RESTORE REGISTERS
FE8C 8501     0037          STA I6509
FE8E 68       0038          PLA
FE8F 8DFF01   0039          STA STACKP
FE92 68       0040          PLA
FE93 85AD     0041          STA IPOINT+1
FE95 68       0042          PLA
FE96 85AC     0043          STA IPOINT
FE98 40       0044          RTI
FE99          0045
-------------------------------------------------------------------
"RTS"-ADRESSE IRQ, BRK
FE99 2910     0047 TXIRQ2   AND #$10          ;TESTE AUF BREAK
FE9B D005     0048          BNE EXBRKX        ;ZU BREAK
-------------------------------------------------------------------
AUSFUEHRUNG IRQ IM SEGMENT $0F
FE9D A5AC     0050          LDA IPOINT
FE9F 4C3CFF   0051          JMP EXIRQ
-------------------------------------------------------------------
AUSFÜHRUNG BRK IM SEGMENT 15
FEA2 A5AC     0053 EXBRKX   LDA IPOINT
FEA4 4C39FF   0054          JMP EXBRK         ;AUSFÜHRUNG BRK
A7          0055
-------------------------------------------------------------------
AUSFÜHRUNG EXTERNE SUBROUTINE
FEA7 08       0057 EXSUBF   PHP               ;STATUS
FEA8 48       0058          PHA
FEA9 A90F     0059          LDA #$0F
FEAB 8501     0060          STA I6509

FEAD 68       0061          PLA
FEAE 28       0062          PLP               ;STATUS ZURÜCK
FEAF 4CB2FE   0063          JMP EXSUB         ;EXTERNE SUBROUTINE
FEB2          0064          *=EXSUB
FEB2 08       0065          PHP               ;STATUS
FEB3 78       0066          SEI
FEB4 48       0067          PHA
FEB5 8A       0068          TXA
FEB6 48       0069          PHA
FEB7 98       0070          TYA
FEB8 48       0071          PHA
FEB9          0072
FEB9 2019FF   0073          JSR IPINIT
FEBC A8       0074          TAY               ;STACKPOINTER OF OTHER SEGMENT
FEBD A500     0075          LDA E6509         ;PROGRAMMSEGMENT
FEBF 202AFF   0076          JSR PUTAS         ;AUF ANDEREN STACK
FEC2 A904     0077          LDA #<EXCRT2      ;XFER-SEG RTS
FEC4 A2FF     0078          LDX #>EXCRT2
FEC6 2024FF   0079          JSR PUTAXS        ;TO OTHER STACK
FEC9 BA       0080          TSX               ;CURRENT SP
```

```
FECA BD0501   0081          LDA $105,X      ;SP+5 - AKTUELLE ROUTINE ADDR. LO
FECD 38       0082          SEC
FECE E903     0083          SBC #3          ;-3 WEGEN JSR DIESER ROUTINE
FED0 48       0084          PHA
FED1 BD0601   0085          LDA $106,X
FED4 E900     0086          SBC #0
FED6 AA       0087          TAX             ;ADDR HI
FED7 68       0088          PLA             ;ADDR. LO
FED8 2024FF   0089          JSR PUTAXS      ;AUF DEN ANDEREN STACK
FEDB 98       0090          TYA             ;STACKPOINTER
FEDC 38       0091 EXCOMM   SEC
FEDD E904     0092          SBC #4          ;WEGEN 4 BYTES
FEDF BDFF01   0093          STA STACKP
FEE2 A8       0094          TAY
FEE3 A204     0095          LDX #4          ;4 REGISTER
FEE5 68       0096 EXSU10   PLA
FEE6 C8       0097          INY
FEE7 91AC     0098          STA (IPOINT),Y
FEE9 CA       0099          DEX
FEEA D0F9     0100          BNE EXSU10
FEEC ACFF01   0101          LDY STACKP
FEEF A92D     0102          LDA #<EXPUL2
FEF1 A2FF     0103          LDX #>EXPUL2
FEF3 2024FF   0106          JSR PUTAXS      ;AUF ANDEREN STACK
FEF6 68       0105          PLA
FEF7 68       0106          PLA
FEF8 BA       0107 EXGBYE   TSX             ;SP DER LFS. BANK
FEF9 8EFF01   0108          STX STACKP      ;SICHERN
EFC 98       0109          TYA
FEFD AA       0110          TAX             ;SP DES NEUEN SEGMENTES
FEFE 9A       0111          TXS
FEFF A501     0112          LDA I6509       ;DATENSEGMENTREG
FF01 4CF6FF   0113          JMP GBYE
-------------------------------------------------------------------
RUCKKEHRADR. WENN RTI
FF04 EA       0115          NOP
-------------------------------------------------------------------
WENN RTS
FF05          0117 EXCRTS
FF05 08       0118          PHP             ;STATUS
FF06 08       0119          PHP
FF07 78       0120          SEI
FF08 48       0121          PHA
FF09 8A       0122          TXA
FF0A 48       0123          PHA
FF0B 98       0124          TYA
FF0C 48       0125          PHA
FF0D BA       0126          TSX             ;SP
FF0E BD0601   0127          LDA $106,X
FF11 8501     0128          STA I6509
FF13 2019FF   0129          JSR IPINIT
FF16 4CDCFE   0130          JMP EXCOMM
-------------------------------------------------------------------
IPOINT INITIALISIEREN
FF19          0132 IPINIT
FF19 A001     0133          LDY #1          ;ADRESSE $100 TO POINTER
FF1B 84AD     0134          STY IPOINT+1
FF1D 88       0135          DEY
FF1E 84AC     0136          STY IPOINT
FF20 88       0137          DEY
FF21 B1AC     0138          LDA (IPOINT),Y  ;ALTEN STACKPOINTER DES SEGMENTS
F23 60       0139          RTS
-------------------------------------------------------------------
AC UND X INS ANDERE SEGMENT AUF DEN STACK
FF24 48       0141 PUTAXS PHA
FF25 8A       0142          TXA
FF26 91AC     0143          STA (IPOINT),Y
FF28 88       0144          DEY
FF29 68       0145          PLA             ;AC ZURUCK
```

```
AC AUF DEN ANDEREN STACK
FF2A 91AC     0147 PUTAS  STA (IPOINT),Y
FF2C 88       0148        DEY               ;AKTUALISIERE MITLAUFENDEN SP
FF2D 60       0149        RTS

REGISTER ZURUCK
FF2E 68       0151 EXPULL PLA
FF2F A8       0152        TAY
FF30 68       0153        PLA
FF31 AA       0154        TAX
FF32 68       0155        PLA
FF33 28       0156        PLP
FF34 60       0157        RTS

EXTERNE INTERRUPT NMI, BRK, IRQ, BANK 15
FF35 20A7FE   0159 EXNMI  JSR EXSUBF
FF38 EA       0160        NOP
FF39 20A7FE   0161 EXBRK  JSR EXSUBF
FF3C          0162 EXIRQ
FF3C 20A7FE   0163        JSR EXSUBF
FF3F          0164
FF3F          0165 EXCRT2 =EXCRTB-1
FF3F          0166 EXPUL2 =EXPULL-1
FF3F          0167

VEKTOR-TABELLE
FF3F          0169        *=$FF6F
FF6F 20A7FE   0170 KVRESE JSR EXSUBF      ;NETZ EIN/AUS RESET
FF72 20A7FE   0171 KIPCGO JSR EXSUBF      ;Z-BEF. MONITOR UMSCH. COPROZESSOR
FF75 20A7FE   0172 KFUNKY JSR EXSUBF      ;FUNKTIONSTASTEN
FF78 20A7FE   0173 KIPCRQ JSR EXSUBF      ;IPC ANFORDERUNG
FF7B 20A7FE   0174 KIOINI JSR EXSUBF      ;INITIALISIERUNG DER E/A
FF7E 20A7FE   0175 KCINT  JSR EXSUBF      ;INIT BILDSCHIRM
FF81 20A7FE   0176 KALLOC JSR EXSUBF      ;ALLOCATION ROUTINE
FF84 20A7FE   0177 KVECTO JSR EXSUBF      ;E/A VEKTOREN LESEN/SCHREIBEN
FF87 20A7FE   0178 KRESTO JSR EXSUBF      ;STANDARD E/A VEKTOREN SETZEN
FF8A 20A7FE   0179 KLKUPS JSR EXSUBF      ;GERÄTE U. SEK. ADRESSE SUCHEN
FF8D 20A7FE   0180 KLKUPL JSR EXSUBF      ;GERÄTE-/SEK. ADR. U. LOG. FILE-# SU.
FF90 20A7FE   0181 KSTMSG JSR EXSUBF      ;MELDUNG DES OS AUSGEBEN
FF93 20A7FE   0182 KSECND JSR EXSUBF      ;IEC: SEK-ADR. NACH LISTEN AUSGEBEN
FF96 20A7FE   0183 KTKSA  JSR EXSUBF      ;IEC: SEK-ADR. NACH TALK UAGEBEN
FF99 20A7FE   0184 KMEMTP JSR EXSUBF      ;MEMTOP LESEN/SCHREIBEN
FF9C 20A7FE   0185 KMEMBT JSR EXSUBF      ;MEMBOTTOM LESEN/SCHR.
F9F 20A7FE    0186 KSCNKY JSR EXSUBF      ;TASTATUR ABFRAGEN
FFA2 20A7FE   0187 KSETMO JSR EXSUBF      ;IEC# ZEITÜBERWACHUNG EIN
FFA5 20A7FE   0188 KACPTR JSR EXSUBF      ;1 BYTE VIN IEC IN AC
FFA8 20A7FE   0189 KCIOUT JSR EXSUBF      ;1 BYTE VON AC NACH IEC
FFAB 20A7FE   0190 KUNTLK JSR EXSUBF      ;UNTALK AUF IEC AUSG.
FFAE 20A7FE   0191 KUNLSN JSR EXSUBF      ;UNLISTEN AUF IEC AUSG.
FFB1 20A7FE   0192 KLISRN JSR EXSUBF      ;LISTEN AUF IEC
FFB4 20A7FE   0193 KTALK  JSR EXSUBF      ;TALK AUF IEC
FFB7 20A7FE   0194 KREAST JSR EXSUBF      ;E/A STATUS R/W
FFBA 20A7FE   0195 KSTLFS JSR EXSUBF      ;FILE PRI- SEK-ADR. EINTR.
FFBD 20A7FE   0196 KSTNAM JSR EXSUBF      ;FILENAME: LÄNGE UND ADR. EINTR.
FFC0 20A7FE   0197 KOPEN  JSR EXSUBF      ;LOG. DATEI ÖFFNEN/BEF. AUSG.
FFC3 20A7FE   0198 KCLOSE JSR EXSUBF      ;LOG. FILE SCHLIESSEN
FFC6 20A7FE   0199 KCHKIN JSR EXSUBF      ;EINGABEKANAL ÖFFNEN
FFC9 20A7FE   0200 KCKOUT JSR EXSUBF      ;AUSGABEKANAL ÖFFNEN
FFCC 20A7FE   0201 KCLRCH JSR EXSUBF      ;E/A-KANAL SCHLIESSEN
FFCF 20A7FE   0202 KBASIN JSR EXSUBF      ;EINGABE 1 CHAR. V. AKTIVEN KANAL IN A
FD2 20A7FE    0203 KBSOUT JSR EXSUBF      ;AUSGABE 1 CHAR AUF AKT. KAN.
FFD5 20A7FE   0204 KLOAD  JSR EXSUBF      ;EINLESEN V. LOG. FILE
FFD8 20A7FE   0205 KSAVE  JSR EXSUBF      ;ABSPEICHERN A. LOG. FILE
FFDB 20A7FE   0206 KSTTIM JSR EXSUBF      ;INTERNE UHR STELLEN
FFDE 20A7FE   0207 KRDTIM JSR EXSUBF      ;UHR LESEN
FFE1 20A7FE   0208 KSTOP  JSR EXSUBF      ;STOP-TASTE LESEN
FFE4 20A7FE   0209 KGETIN JSR EXSUBF      ;1 ZCH V. AKT KAN. NACH AC
FFE7 20A7FE   0210 KCLALL JSR EXSUBF      ; ALLE LOG. FILES CLOSE
FFEA 20A7FE   0211 KUDTIM JSR EXSUBF      ;UHR BEDIENEN
FFED 20A7FE   0212 KSCROR JSR EXSUBF      ;MAX. ANZ. ZEILEN=X, SP.=Y
FFF0 20A7FE   0213 KPLOT  JSR EXSUBF      ;CURSOR KOORD. R/W
```

```
F3 20A7FE  0218 IOBASE JSR EXSUBF +BASISADR. IO-GER. NACH X,Y
-------------------------------------------------------------
GOOD BYE
FFF6 8500     0220 GBYE   STA E6509       ;EXECUTION REG
FFF8 60       0221        RTS
-------------------------------------------------------------
HARDWARE-VEKTOREN
FFF9 7AFE     0223 HANMI  .WOR TXNMI       ;INDIR JMP NMI VON $FFFA
FFFB 0000     0224 HARES  .WOR 00          ;SYSTEM RESET IMMER NACH BANK 15
FFFD 64FE     0225 HAIRQ  .WOR TXIRQ       ;IRQ-ROUTINE
FFFF          0226        .END                                  R.L.□
```

Dipl.-Ing. Rüdiger Wollenberg, 4600 Dortmund 50

ROFF

Textformatierung für 6502-Mikroprozessoren

1.0 Einleitung

Nach neueren Studien werden über 70% der Computersysteme überwiegend für Textver - und bearbeitung eingesetzt. Verglichen mit den alten Rechnern, die vor ca. vier Jahren aktuell gewesen sind, verfügen die meisten neueren Systeme über die entsprechenden internen Möglichkeiten wie Groß - und Kleinschreibung, Deutscher Zeichensatz mit Umlauten und "ß", sowie Tastenrepeat. Auch besitzen heute schon viele Systeme das beste Editiersystem, einen Bildschirmeditor /4/. Ältere sind von Seiten der Hersteller oder durch eigene Erweiterungen /2,3,4/ auf ein passables Niveau gebracht worden, um den Anforderungen der Textverarbeitung genüge zu leisten.

Gerade wenn längere Berichte zu schreiben sind, wie sie in Forschung, Entwicklung, aber auch bei der Erstellung von studentischen Diplom - und Seminararbeiten anfallen, wünscht man sich eine Hilfe, die über einen reinen Editor hinausgeht und einem die Arbeit des Textformatierens abnimmt.

1.1 Eigenschaften eines Formatierers

Die Hauptaufgabe eines Formatierers ist der Ausgleich des rechten Randes. Dazu wird eine Zeile Wort für Wort aufgefüllt, bis das nächste Wort nicht mehr hinein paßt. Anschließend erfolgt die Erstellung des sogenannten Blocksatzes über das Auffüllen von Blanks. Grundsätzlich ist es egal, wie viele Worte die Orginaltextzeile enthält und durch wie viele Blanks zwei Worte getrennt sind. Dies erlaubt ein relativ zwangloses Erstellen des Schriftsatzes.

Das Ausrichten des rechten Randes ist nicht einfach, weil man einen Lochfluß auf einer Textseite vermeiden muß. Hier ist eine Lösung gewählt worden, die ein wechselseitiges Auffüllen vom rechten und linken Rand aus gestattet.

Zu den weiteren Eigenschaften gehört eine Beinflussung des rechten und linken Randes, um z.B. Paragrapheneinrückungen oder variable Textbreiten zu ermöglichen. Die Anzahl der Zeilen pro Seite ist per Steuerkommando den individuellen Gegebenheiten anpaßbar. Eine automatische Seitennummerierung, sowie eine beliebig gestaltbare Kopf - und Fußzeile geben dem Text ein Aussehen, wie man es in Büchern findet.

Der Komfort kann durch weitere Kommandos erhöht werden. Dazu gehören Befehle zur Zentrierung von Zeilen, zur Peripherieansteuerung bei Fettgeschriebenen und unterstrichenen Zeilen. Man kann für den Ausdruck von Tabellen eine Mindestzeilenzahl anfordern. Ist sie nicht mehr gegeben, wird die Textseite verlassen. Der Ausdruck erfolgt dann auf der nächsten Seite.

1.2 Entstehung von ROFF

Das hier vorgestellte Programm ROFF ist eine weitere von vielen bereits existierenden Versionen,

die RUNOFF, ROFF, NROFF oder auch TROFF heißen. Die meisten laufen auf Mini - und Großrechnern, z.B. Hewlett-Packard und Prime - Prozeßrechnern und sind in FORTRAN oder PASCAL geschrieben. Einige andere Versionen sind auf UNIX - und CTSS -Systemen implementiert.

Wegen der höheren Rechenleistung und besseren Programmiermöglichkeiten sind dort zumeist noch erheblich mehr Kommandos definiert, wie z.B. automatische Silbentrennung, Proportionalschriftverarbeitung, Zeichenersetzung.

Die hier vorgestellte Version lehnt sich stark an Kernighan/Plaugers Programmierwerkzeuge /1/ an, die einen Formatierer in FORTRAN vorstellen. Das Buch ist für alle eine wichtige Lektüre, die sich mit der Programmierung komplexer Programmsysteme beschäftigen. Es bietet auch auf anderen Softwaregebieten viele Hilfen und Anregungen. Für eine vertiefende Auseinandersetzung mit der Problematik von Textformatierern ist es dringend empfohlen.

1.3 Implementierungshinweise

Die Substanz dieses Artikel liegt hauptsächlich in der Umsetzung der 16-Bit-Zweierkomplement-Arithmetik in ein 8-Bit-Zahlenformat ohne negative Zahlen. Das hat zur Folge, daß die automatische Seitennummerierung lediglich bis zur Seite 255 reicht. Ebenso ist es mit der Ausrichtung des rechten und linken Randes und der Zeilenzahl pro Seite. In der Regel bedeutet der Übergang auf die 8-Bitzahlen keine Einschränkung. Lediglich die Plausibilitätsprüfung der eingegebenen Kommandos leidet etwas, weil negative Parameter nicht mehr von sehr großen positiven unterschieden werden können.

Die zweite fast gleichgewichige Arbeit steckt in der Parameterbehandlung der Routinen. Im FORTRAN-Compiler werden die Parameter in Form einer Adresse an Unterprogramme übergeben. Dabei kann der Variablenname im Unterprogramm von dem des Hauptprogrammes abweichen. In der Assemblersprache ist nur eine Gleichnamigkeit möglich. Der Befehlssatz des 6502 unterstützt leider eine ParameterÜbergabe per Adresse von den Maschinenbefehlen her nur indirekt. Es wäre also eine eigene Routine zu entwickeln, die das Parameterhandling durchführt. Hier wurde ein anderer Weg bestritten, der übersichtlicher ist. Einzelne Unterprogramme sind doppelt ausgelegt und unterscheiden sich nur durch die verwendeten Variablen. Der Bytebedarf ist nicht sehr viel größer als bei dem vorgenannten Verfahren.

Weil viele druckende Peripheriegeräte heutzutage über eigene Intelligenz verfügen, wurde hier exemplarisch die Druckeransteuerung über Escapesequenzen durchgeführt. Eine Escapesequenz beginnt mit dem Escapezeichen hexadezimal 1B und einem weiteren ASCII-Code, der die Art der Sequenz repräsentiert. Auf diese Weise konnten die Fettschrift- und Unterstreichmöglichkeiten der Typenradschreibmaschine des Autors genutzt werden. /1/ schildert die Implementierung der Unterstreichung auf jedem Druckgerät über eine Auflösung der Zeichen in "Zeichen", "RÜckschritt", "Unterstrich".

Das Programm wurde für 6502-Mikroprozessoren entwickelt und auf dem AIM 65 mit geändertem Monitorprogramm REMON /3/ implementiert. Weiterhin existiert ein Screeneditor /4/, der Voraussetzung für eine komfortable Textbearbeitung ist. Die Adaption auf anderen Mikrocomputern mit 6502-CPU ist kein großes Problem, weil lediglich zwei kleine Unterprogramme auf den jeweiligen Rechner angepaßt werden müssen. Dies ist zum einen die Routine GETLIN, die eine Textzeile in den Eingangsvektor INBUF überträgt, und das Programm PUTC, das ein ASCII - Zeichen auf das Peripheriegerät ausgibt.

Die Belegung der Zeropage für die Variablen ist nicht zwingend. Es kann auch auf andere Speicherbereiche ausgewichen werden.

Vom Autor kann ein 2532-EPROM bezogen werden, das in den $Cxxx - Sockel der AIM 65 - Hauptplatine gesteckt wird und das sowohl unter dem normalen AIM - Betriebssystem als auch unter dem verbesserten Monitorprogramm REMON /3/ läuft. Die hier beschriebene Software ist darin vollständig enthalten. Sie wird durch Ansprung der Speicherstelle $C800 aktiviert. Auf die

Ausgabe OUT= kann man einen der üblichen Ausgabekanäle anwählen. Der zu bearbeitende Text muß zuvor im Speicherbereich des Editors abgelegt sein. Er wird durch die Formatierung nicht verändert. Zu beachten ist, daß neben dem Zeropage - Bereich auch die Speicherstellen von $200 bis $400 als interne Bufferspeicher genutzt werden.

2.0 Kommandos

Der Eingabetext wird von ROFF in zwei Kategorien unterteilt. Ist das erste Zeichen einer neuen Zeile ein Punkt, wird dies als Kommandozeile interpretiert und der Befehl ausgeführt. Alle anderen Zeilen sind reine Textzeilen, deren Inhalt je nach Wahl formatiert oder unformatiert gedruckt wird.

Ein Kommando besteht außer dem Punkt noch aus mindestens zwei Buchstaben und eventuellen nachfolgenden Parametern. Die Kommandos können dabei klein oder auch groß geschrieben sein. So bedeutet beispielsweise die Zeile .sp 2, daß während des Ausdrucks zwei Leerzeilen gelassen werden.

Weil absolute Werte für Parameter oft unhandlich sind, läßt ROFF auch relative Zahlen zu. Durch ein vorgestelltes '+' bzw. '-' wird der aktuelle Wert nicht durch einen festen ersetzt, sondern um die Größe verändert.

Selbstverständlich werden auch Texte ohne jedes Formatierkommando bearbeitet, weil eine Grundeinstellung der verschiedenen Parameter während der Initialisierung vorgenommen wird.

Im folgenden wird der Kommandosatz von ROFF beschrieben:

Der oben geschilderte Blocksatz wird mit der Anweisung

.nf (no fill)

ausgeschaltet. Damit werden die Textzeilen ohne weitere Modifikation auf die Ausgabe gegeben.

.fi (fill)

schaltet den Blocksatzbetrieb wieder ein. Wird der Blocksatz ausgeschaltet, kann es vorkommen, daß eine Zeile noch nicht aufgefüllt gewesen ist und damit noch nicht ausgegeben war. Es muß daher zuerst diese Zeile ausgedruckt werden. Solch ein Abbruch

.br (break)

wird von einigen Kommandos implizit vorgenommen.

Ein Abbruch wird auch von

.sp n (space)

bewirkt, das n Leerzeilen ausführt. Ist der untere Rand einer Seite erreicht, werden die dann überflüssigen Leerzeilen ignoriert, damit alle Seiten üblicherweise mit Text beginnen. Eine Leerzeile hat die gleiche Auswirkung wie .sp 1.

.ls n (line space)

bedeutet, daß immer n Leerzeilen zwischen zwei ausdruckbare Zeilen generiert werden. Das kann während der Entwurfsphase benutzt werden, um in den Text Einfügungen einzugeben.

Soll eine neue Seite angefangen werden, wie dies beispielsweise bei dem Beginn eines neuen Kapitels nötig sein könnte, wird der Befehl

.bp n (begin page)

benutzt. Dabei bezeichnet n die Zahl mit der die Seite nummeriert wird. Läßt man den Parameter n weg, wird die neue Seite automatisch mit der folgenden Seitenzahl beziffert. Leere Seiten, z. B.

für Bilder oder Diagramme, werden über .bp +2 erzeugt.

Die Zeilenzahl pro Seite ist auf 66 eingestellt. Sie kann geändert werden mit

.pl n (page length).

Um Absätze und Tabellen nicht zu zerschneiden, kann mit

.ne n (need)

geprüft werden, ob eine Seite noch mindestens n Zeilen bis zum Ende enthält. Ist das nicht mehr gewährleistet, wird die aktuelle Seite verlassen und der Text auf der frischen Seite geschrieben. Die Sequenz für Freiräume zum späteren Einkleben von Bildmaterialien lautet:

```
.ne n (Prüfung, ob Freiraum vorhanden ist)
.sp n (Schafft Freiraum).
```

Um bei neuen Absätzen nicht einzelne Zeilen auf der unteren Blattseite stehen zu haben, gibt es den Befehl

.pp (paragraph).

Er führt den Befehl .sp 1 aus, wenn noch mindestens für vier Zeilen Platz auf der Seite ist und .bp +1, wenn die Seite schon fast voll ist.

Die aktuelle Position des rechten Druckrandes wird angegeben mittels

.rm n (right margin).

Zentrierungen der nächsten n Zeilen können erzeugt werden mit

.ce n (center).

Will man ein Zeilenzählen sparen, läßt sich mit Hilfe des Kommandos .ce 255 der Zentriermodus einschalten für 255 Zeilen und mit .ce 0 wieder ausschalten. Bei diesem Kommando erfolgt kein Randausgleich.

Ähnlich wirken die Kommandos zum Fettschreiben und zur Unterstreichung. Sie bewirken aber keinen Zeilenabbruch und können damit auch innerhalb einer Zeile benutzt werden:

.fa n (fat)

.ul n (underline) .

Zur Beeinflussung des linken Randes gibt es zwei Anweisungen.

.in n (indent)

rückt alle nachfolgenden Ausgabezeilen um n Zeichen ein, während

.ti n (temporary indent)

lediglich die nächste Zeile um n Zeichen einrückt.

Die Kopf - und Fußzeile werden über die Kommandos

.he ' Seite # (header) und

.fo " - # - (footer)

eingegeben. Der Anfang des Titels wird durch das erste nichtleere Zeichen gekennzeichnet. Jedoch wird das einfache bzw. doppelte Anführungszeichen ignoriert und erlaubt die Eingabe einer definierten Zahl von Blanks vor den druckbaren Zeichen. Das Nummerzeichen in der Kopf - bzw. Fußzeile ist ein Platzhalter. Beim Ausdrucken wird an dieser Stelle die aktuelle Seitenzahl eingesetzt.

Für die Texterstellungsphase sind die zwei folgenden Kommandos sehr wertvoll. Mit

.bs (begin skip)

wird die Textausgabe eines Files unterdrückt, bis man zu dem Kapitel gekommen ist, das man gerade bearbeitet. Dort kann mit

.es (end skip)

das Überspringen der Textausgabe beendet werden.

2.1 Befehlsübersicht

Befehl	Break	Init.	Funktion	Beschreibung
.fi	j	on	fill	Blocksatz einschalten
.nf	j	off	no fill	Blocksatz ausschalten
.br	j		break	Abbruch einer Zeile
.sp n	j	n=1	space	Einfügungen von Leerzeilen
.ls n	n	n=1	line space	Zeilenabstand
.bp n	j	n=+1	begin page	Neue Seite anfangen
.pl n	n	n=66	page length	Seitenlänge einstellen
.ne n	n	n=0	need	Eventueller Seitenwechsel
.pp	j		paragraph	Absatz erzeugen
.rm n	n	n=70	right margin	Zeilenlänge einstellen
.ce n	j	n=0	center	Zentrierung
.fa n	n	n=0	fat	Fett schreiben
.ul n	n	n=0	underline	Unterstreichen
.in n	j	n=0	indent	Einrückung um n Spalten
.ti n	j	n=0	temp. indent	Einmalige Einrückung
.he	n	CR	header	Kopftitel angeben
.fo	n	CR	footer	Fußtitel angeben
.bs	n		begin skip	Textausgabe überspringen
.es	n		end skip	Ende des Überspringens

2.2 Besonderheiten

Manchmal ist es wichtig, im Blocksatz eine genaue Anzahl von Blanks zwischen zwei Worten zu haben oder die Trennung eines Wortes zu verhindern, z. B. bei der Abkürzung DIN A 4. Aus diesem Grund wurde das Dachsymbol hexadezimal 5E benutzt. Es ist ein nichtleeres Zeichen und wird dementsprechend bei der Auffüllung auch als solches behandelt. D.h., daß zwei Worte, die über das Dachsymbol getrennt sind, von dem Formatierer als ein einziges gelesen werden. Erst vor der unmittelbaren Ausgabe erfolgt die Umwandlung in ein Blank.

4.0 Programmlisting

Das Programm ist implementiert auf einem AIM 65 und läuft seit einem halben Jahr sehr zufriedenstellend.

Bei Verwendung auf anderen Rechnern müssen lediglich zwei Unterprogramme geändert werden.

5.0 Zusammenfassung

ROFF ist beim Autor seit mehreren Monaten erfolgreich in Betrieb genommen worden. Für den unvoreingenommenen Benutzer scheint das Arbeiten mit den eingestreuten Kommandos zuerst etwas ungewohnt. Man beherrscht die Bedienung allerdings sehr schnell. So ist z.B. diese Beschreibung unter Verwendung des hier beschriebenen Programms entwickelt worden. Es ist zu keiner Zeit zu Schwierigkeiten gekommen.

Ein artverwandtes Programm mit der Bezeichnung NROFF wird seit ca. einem Jahr an der Arbeitsstelle des Autors im universitären Bereich verwendet für die Erstellung von Forschungsberichten und Diplomarbeiten. Selbst Benutzern, die mit Rechenanlagen nicht sehr vertraut sind, fällt die Anwendung dieses Programms sehr leicht, so daß es auf eine große Akzeptanz gestoßen ist.

6.0 Literatur

/1/ Kernighan B.W., Plauger P.L.: Programmierwerkzeuge.
Springer-Verlag Berlin, Heidelberg, New York 1980.

/2/ Redder A., Wollenberg R.: EDEDIT-Extended Editor für
AIM 65. 65xx MICRO MAG Nr.31, S.??-?? , Juni 1983.

/3/ Wollenberg R., Redder A.: REMON-Redigierter Monitor.
65xx MICRO MAG Nr. 28, Seite 3-14, Dezember 1982.

/4/ Wollenberg R., Redder A.: SCREEN-Bildschirmeditor für
AIM 65. 65xx MICRO MAG Nr. 30, Seite 10-22, April 1983.

Hinweis des Herausgebers: Der in diesem Heft noch fehlende Abschnitt 3 mit den Hinweisen zum Programmaufbau sowie die umfangreiche Programm-Liste werden in den folgenden Heften abgedruckt.

Ein FORTH-Splitter

Herr Werner Mäser aus CH-9042 Speicher weist darauf hin, daß das FORTH-Wort ?TERMINAL beim AIM 65 das X-Register umsetzt und damit den Stackpointer des FORTH zerstört.

Editorial

Ohne Floppy Disk kann man als Computerbetreiber eigentlich gar nicht mehr leben. Die Abspeicherung von Programmen und Daten auf Cassette oder Datasette konnte nur ein Provisorium sein. Mühseliges Nachtippen wurde zudem erforderlich, wenn ein nicht doppelt abgespeichertes File durch Behandlungs- oder Lesefehler unbrauchbar wurde. Die vielen telefonischen Rückfragen beim Herausgeber zeigten immer wieder das große Interesse, das bei allen Betreibern an einer Floppy Disk besteht. Haupthindernis für die Anschaffung war bisher jedoch der recht hohe Preis.

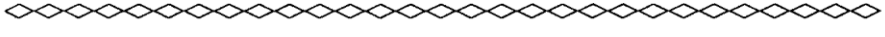
Dieses Heft enthält aus der Feder von Michael Zimmermann den Vorschlag für einen preiswerten Aufbau. Im weiteren Verlauf wird das Betriebsprogramm für die Hardware veröffentlicht werden. Es wird zeigen, wie man die Diskette verwaltet und in das Betriebssystem einbezieht.

Ob man andererseits eine fertige Floppy anschafft, ist sicher auch eine Frage der Kosten, der Bequemlichkeit und der weiteren Planung. Nach den seit Juni 1981 veröffentlichten Routinen für den IEEE 488-Bus hat der Herausgeber in vielen Gesprächen dafür plädiert, Floppies mit diesem Busanschluß zu betreiben, weil sie mit eigener Intelligenz alle Aktionen korrekt verwalten. Sie 'verstehen' ASCII-Strings, die man ihnen als Befehl übergibt. Das ist die Frage der Bequemlichkeit. Es ist Herrn Peter Rix zu danken, daß er mit der zweiten größeren Arbeit in diesem Heft eine konsequente Fortentwicklung des Betriebssystems für solche Floppies vorlegt. Mit dieser Dokumentation in der Hand wird man auch einen eventuellen künftigen Systemwechsel leicht meistern und eine vorhandene Floppy weiterbenutzen können. Das ist die Frage der Planung. Auch der Verbund mehrerer Rechner via Bus und gemeinsamer Flopyy ist leicht möglich. Bei einer solchen Konfiguration hat man den Vorteil, daß der eine Rechner dem anderen mit seinen vielleicht besseren Entwicklungshilfsmitteln unterstützen kann. So ist z.B. das in diesem Heft abgedruckte Programm zum Banking beim CBM 710 in den Assembler des AIM 65 eingegeben worden.

Zur Preisfrage bei Floppies ist festzustellen, daß auch diese Geräte in diesem Jahr billiger geworden sind. Die mit seriellem Anschluß an den Volkscomputern betriebene CBM 1541 kostet heuer nur noch DM 675,-. Wenn man sich vor Augen hält daß man für diesen Preis ein fertiges Gerät mit Laufwerk, Elektronik, Netzteil und Gehäuse erhält, dann wird in vielen Fällen die Mühe eines Selbstaufbaues nicht mehr lohnen. Ihr Betriebssystem hat alle guten Eigenschaften des bei der 4040 bekannten DOS 2, einschließlich der Verwaltung relativer Dateien. Allerdings: Sie läßt sich nicht am parallelen IEEE-Bus betreiben. Der dafür geeigneten Single Drive hat noch etwa den doppelten Preis. Es ist daher bei ausreichendem Interesse geplant, zur Lieferung ab November einen Interface-Prozessor mit einfachster Beschaltung herauszubringen, der diese Floppy 1541 am parallelen IEEE-Bus zu betreiben gestattet. Das ist sicher nicht nur für Betreiber eines CBM interessant, vielmehr auch für Besitzer anderer Rechner, die dann die von P. Rix entwickelten Routinen benutzen.

In das vorliegende Heft konnten wiederum nicht alle vorliegenden Arbeiten ganz oder geteilt aufgenommen werden. Der Herausgeber bittet die betroffenen Autoren da noch um etwas Geduld. Auch in einer solchen Zeitschrift gibt es ja immer einige aktuelle Bezüge, die berücksichtigt werden wollen. So ist es u.a. sehr zu begrüßen, hier schon einen Anwenderbericht über die LISA zu finden, obwohl dieses Gerät in diesem Land wohl bisher kaum ausgeliefert wurde. Mit dem Eintreffen von Datenblättern wurde es ferner möglich, die neuen Commodore-Chips zu dokumentieren, um dem bestehenden Informationsbedürfnis aktuell zu entsprechen.

Diesem Heft ist auf grünen Seiten einmal wieder ein Gesamt-Inhaltsverzeichnis beigefügt, das die zusammenfassenden Bücher 1-6 und 7-13 sowie die Hefte 14 bis 32 überspannt. Der Leser möge es mit Gewinn für seine eigene Orientierung benutzen. Er ist aber auch herzlich gebeten, ggfs. Fotokopien im Bekanntenkreis empfehlend weiterzureichen.



SYSTEMS 83 in München vom 17. bis 21. Oktober. Der Herausgeber ist am 17. und 18. auf dem Messegelände. Mit einigem Glück ist ggfs. eine Verabredung über den Rockwell-Stand möglich. Man wende sich bitte an Herrn Christian Streicher.

Offenes, zwangloses Meeting von Computerfreunden am Montag, den 17. Oktober ab 18 Uhr in der Gaststätte Heide Volm, Bahnhofstraße 51 in 8033 Planegg, direkt am S-Bahnhof. Planegg ist 8 Minuten Fahrzeit von der Innenstadt entfernt, Linie S6 nach Starnberg. Wenn alles klappt, wird man dort nicht nur den Herausgeber (sicher!) treffen, sondern auch Jim Butterfield aus Toronto, der die Messe besuchen will.

Aus der Branche

Rockwell International stellt einen elektrisch löschbaren PROM, den R5213 mit 16K vor. Die möglichen Einsatzbereiche beinhalten die Kalibrierung von Meßsystemen, programmierbare Zeichengeneratoren, Telefonwählsysteme und Fahrzeugkontrollen. Bytes oder Chips können on board gelöscht (21-V) und wieder beschrieben. werden .Daneben gibt es die Betriebsarten Schreibsperre, Lesen und Standby. Eine sicher interessante Kombination von nicht flüchtigem und Schreib-Lesespeicher. - Lieferbar ist jetzt auch ein 8Kx8 CMOS - EPROM mit geringer Leistungsaufnahme (0.5 mW im standby, max. 132 mW im Betrieb), das mit R87C32 bezeichnet wird. Eine 64KVersion ist für das 1. Quart 1984 angekündigt.

CCU 8086 heißt ein Tochterprozessor der ancon Ingenieurgesellschaft mbH in 1000 Berlin 62, Kärntener Str. 8, Tel. 030 -7 84 10 20/29. Es handelt sich um eine Prozessorplatine mit CPU 8086 (5 oder 8 MHz), 128 kB RAM, einem Urlader und Interfacebaustein PIO 8522. Der Preis ist mit DM 999,- inkl. MWSt (o.Gew.) interessant. Die Europakarte kann an fast jeden Host angeschlossen werden und wird mit CBIOS auf der Floppy des Host geliefert, für diverse Rechner kann MS-DOS 2.0 und CP/M-86 geliefert werden. Mit einer solchen Kombination steht dem Hostrechner das weite Softwareangebot des IBM PC und vergleichbarer Computer zur Verfügung. - Es wäre interessant, von vielleicht schon vorliegenden Erfahrungen zu hören.

Die FORTH-Produkte des Verlages MountainView werden ab 1. Nov. 1983 in Deutschland ab Lager lieferbar sein, und zwar bei der Firma Angelika Flesch, Schützenstr. 3, 7820 Titisee-Neustadt, Tel.: 07651-16 65. Es sollen auch weitere FORTH-Bücher dort vertrieben werden, nebst Erweiterungen für Floating Point und Datenbankverwaltung, Cross-Compiler.

Die GWK Ges. für technische Elektronik aus Herzogenrath stellt auf der Systems in Halle 18, Stand 18000 aus. Neu sind einige Produkte für den 68000/FORCE: Relaiskarte zum VME-Bus und für den GWK-Bus und FORCE Kit 2 mit Real Time Clock. Es gibt ein Interface zwischen VMEbus und GWK-Bus, so daß alle vorhandenen 8-Bit-Karten für den 68000 zur Verfügung stehen.

Kleinanzeigen

AIM 65 aus Europakarten mit Netzteil, Philips Mini-DCR in 3 HE 19"-Gehäuse und abgesetzte Tastatur. Mit 20K CMOS-RAM, 26K ROM, RS 232, 20 mA, Centronics, 2 User-VIA, echtes Videointerface, EPROMMER, Handbücher, 8K-12K BASIC, 4K Assembl., PL/65, FORTH, PASCAL, Schach, REMON u.v.m.. VB 1950,- DM. G. Wilde, Tel.: 021 34 - 35 504.

Gelegenheit: AIM 65/40. Komplett mit Netzteil im Gehäuse. Betriebsbereit für Terminal und Drucker (Centronics-Schnittstelle). Ohne Display und Thermodrucker. Firmware: I/O und Monitor-ROM, Assembler und Terminal-Betriebssystem in EPROMs. Ausführliche Dokumentation. VB DM 1300,-. Kuhlmey, Tel.: 0421 - 50 22 97.

Verkaufe PC 100 mit BASIC, Assembl., PL/65 , 28 K RAM, Zusatznetzteil, 19" Einschub, EPROM-Programmiergerät, Handbücher u. org. Siemens-Software, DM 1100,-. **Video-Interface** CRT2 DM 180,-. N. Gerfelder, Stormstr. 5, 6453 Seligenstadt, Tel.: 061 82 - 13 37.

ITT Mikroprozessorlehrgang 8080 DM 350,-. Mehrere TV-Geräte als Monitore (12 MHz) umgebaut je DM 100,- zu Verkaufen. Angele, Bruchwettern 3c, 2800 Bremen 33, Tel.: 0421 - 27 30 82.

Textverarbeitung mit AIM 65: Groß- und Kleinschreibung umschaltbar mit CNTL A, deutsche Tastaturbelegung, automatisches Tastenrepeat, Schnelles REPLACE, Delete über DILINK, Treibersoftware für serielle Druckerschnittstelle, Editorerweiterungen einfach über Vektoren u.v.a.m.. Zwei geänderte Monitor-EPROMs REMON 3.5 für 98,- DM inkl. ausführlicher Dokumentation.
Textformatierer ROFF 3.2 für jeden AIM 65. 19 Formatierkommandos für Blocksatz, Seitenumbruch, automatische Seitennumerierung, Zentrierung u.v.a.m.. Software im 2532-EPROM für Steckplatz $C000 auf der AIM-Hauptplatine für 98,- DM inkl. ausführlicher Dokumentation.
Paketangebot: REMON 3.5. + ROFF 3.2 + EXEDIT 5.3. Erweiterter Editor mit Blockmove, -copy, -erase, Change to End, Help-Funktion, zweiter Editor aktiv u.a.. Drei 2532 EPROMs in den C, E und F-Sockeln der AIM-Hauptplatine für komplett 178,- DM inkl. ausführl. Dokumentation.
Bestellungen bei Dipl.-Ing Rüdiger Wollenberg, Stockumer Str. 234, 4600 Dortmund 50, Tel. 0231 - 75 34 77.

IICC Schnittstellen-Rechner

IICC: Intelligent Interface Controller Card

für **Anwendungen** wie

BCD Interface, Centronics Interface (.. für Drucker)
V.24/RS232 Interface (.. potentialfreie Rechnerkopplung)
IEEE 488 (IEC 625) Instrument Bus Interface
Codewandler, Echtzeituhr, Datenlogger, Puffer etc..

- als single board computer, eigener Taktgenerator
- als slave system am MCS Bus, synchron zum MCS Takt

Aufbau: : Europakarte, 5V, VG64 Messerleiste [2]

6504	CPU	8kByte Adreßraum (opt. CMOS, erweiterter Befehlssatz)

Peripherie [1]

6522	VIA	2 parallele Ports, Handshake Logik, Latches, 2 Timer, Schieberegister (I/O über 22pol. Stiftleiste)
6551	ACIA	asynchroner serieller Baustein Taktgenerator, progr. Baudraten, voll duplex, Echo mode, Modemsteuerung (I/O über 11pol. Stiftleiste)
68488	GPIA	IEEE488 GPIB Talker/Listener open collector/tri-state Treiber (I/O über 24pol. IEEE488 Buchse)

Speicher

6116	RAM	2kByte (pseudo dual port) [2] CMOS RAM
2716	PROM	2 Steckplätze

[1] alle Interface Bausteine liegen code- und rechenzeit-optimiert auf page 0

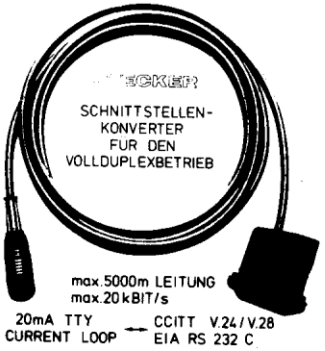
[2] als slave system beansprucht die IICC 2kByte (beliebig wählbar) aus dem Adreßraum des master system.
Auch mehrere slaves möglich! Keine Taktverlängerung!

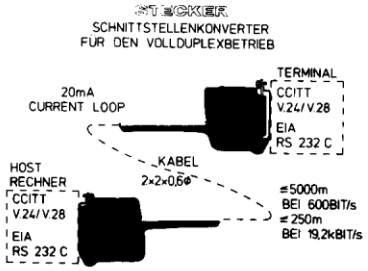
Div. Optionen (System Takt, Baud Takt, EPROMs, CMOS)
ab DM 590,- plus MWSt.
Entwicklungs - EPROM DM 23,- zzgl. MWSt.

K. W. Angele, Bruchwettern 3c, 2800 Bremen 33, Tel.: 0421 -27 30 82

Die Stecker
von Stecker haben es in sich!

Die Wandlung der V.24/V.28 Spannungspegel in einen 20 mA Strompegel erfolgt durch eine Schaltung, die in die serienmäßigen Griffschalen der 25-poligen Subminiatur D-Stecker von AMP, Amphenol, Harting usw. eingebaut wird. Sie haben keine lästigen externen Geräte mehr herumstehen und müssen sich auch nicht mehr um deren Spannungsversorgung kümmern.





Preis komplett wie abgebildet
mit 2,5 Meter Kabel
DM 438,90

Preis je Stück ohne Kabel
DM 342,--
Alle Preise inklusive MWSt

Der Stecker
von Stecker wird einfach in die V.24 Buchsenleiste eingesteckt und schon können Sie mit der Datenübertragung bis zu

5 km

-je nach Widerstand und Kapazität der Leitung und der Baudrate- ihre Daten zwischen Rechner und Datenendgerät übertragen. Die Spannungsversorgung für
die Stecker
von Stecker erfolgt mit plus und minus 12 Volt über die in der DIN 66020 nicht genormten Steckkontakte 9 und 10. Beide Spannungen werden nach CCITT Empfehlung für V.24 und V.28 typischerweise in Ihren Datenendgeräten und Computern bereitgehalten.
Alle Leitungen sind über Optokoppler galvanisch getrennt, so daß Einstreuungen über die Leitung nicht zur Zerstörung Ihrer Geräte und Anlagen führt. Die Datenübertragungsgeschwindigkeit beträgt max. 20 Kilobits/sec im vollduplex Mode.

1 Jahr Garantie.

Gebrauchsmuster angemeldet.

STECKER
INGENIEURBÜRO

Postfach 60 07 66 5000 Köln 60 Tel.: (0221) 7 12 40 18
Neue Telefon-Nummer!!

Assembler unter FORTH

CROSS–09

Cross-Assembler für MC6809 (Motorola). Tonbandcassette (6,5K), Diskette (CBM 3040/4040) oder 2 EPROMs, deren Inhalt nach hex 200-1B00 kopiert werden muß. DM 380,-- + MWSt.

CROSS–05

Cross-Assembler für MC6805/68705xx (Motorola), auf Cassette, Diskette oder EPROM für D000: DM 320,-- + MWSt.

Alle Assembler sind komfortabel ausgelegt, benutzen die Mnemonics des Herstellers und laufen unter FORTH des AIM 65 (für CBM-FORTH von Lowinski oder PHS ggfs. auf Anfrage). Sie erzeugen EPROM-fähigen Code für beliebige Speicherräume und enthalten alle üblichen Kontrollstrukturen wie BEGIN, WHILE, REPEAT, IF, ELSE, THEN usw. aber auch alle Branches (6809: Longbranches). Es kann mit externen Symbolen und Labels im Programmablauf gearbeitet werden. Zusätzlich sind viele Assembler-Anweisungen implementiert: Byte- Word- und Stringablage, .OPTLIS, Reserve Memory Bytes und binäre Argumente.

Mathe–ROM für 6502

Implementierung von P. Rix (s. Textteil) für Sockel D000. Fließkommaarithmetik und höhere mathem. Funktionen (wie in Microsoft-BASIC) für AIM 65-FORTH(komfortabler eigener Fließkommastapel) und für jedes 6502-Assemblerprogramm (dokumentierte Einsprungspunkte und Argumente). EPROM DM 110,- + MWSt.

Commodore Serie 700 und 600 ROM-Listing DM 74,--

Ca. 300 Seiten, gebunden, ausführliche Dokumentation mit original CBM-Labeln, Belegung der Zero Page, des RAM und des I/O, Cross-Referenzliste. Assemblierfähiger Quelltext. Die Benutzung der Banks/Segmente wird erklärt.

LASER 110

PERSONAL-COMPUTER

Prompt lieferbar: Computer-Konsole mit Microsoft-BASIC und Betriebssystem in 16 KB für Anschluß an TV-Gerät oder Daten-Monitor (TV- und Video-Ausgang), BASIC mit 6 signifikanten Ziffern und PRINT USING.

CPU Z80A, 4KB RAM. Siehe auch die Besprechung in Heft 32. Mit externem Netzteil, TV-Anschlußkabel, Kabel für Cassetten-Rekorder, Demo-Cassette. Deutsches Anwenderhandbuch.

Preis inkl. MWSt, Verp. und Versand DM255,

Floppy-Schnittstelle

Umsetzung des parallelen IEEE 488 Busses auf die preiswerte CBM-Floppy 1541. Gestattet den Betrieb von CBM und PET mit der Floppy ebenso aller anderen Rechner mit IEEE-Bus Routinen (s. Heft 33). Einfacher Anschluß des Schnittstellenprozessors (28 Pin). Lieferbar ab November 1983. DM 198,- inkl. MWSt und Zusendung.

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

R. S. Microcomputer GmbH
Gagernstraße 4
8580 Bayreuth
Telefon (0921) 68877

HARD + SOFT

PROFILA

Programmsystem Fakturierung und integrierte Lager- und Adreßverwaltung

Dieses Programmsystem zeichnet sich durch seinen Bedienungskomfort, seine große Leistungsfähigkeit und durch seine hohe Bearbeitungsgeschwindigkeit aus. Es ist ein deutsches Programm für CBM 8032, 8096 und 600/700, variabel mit frei einrichtbaren Druckertreibern für alle am Markt befindlichen Drucker.

PROFILA Fakturierung

Dieser Programmteil erstellt Lieferscheine, Rechnungen und Gutschriften aus dem Lagerbestand heraus. Die Artikel werden automatisch abgebucht, die Umsätze werden bei den Artikeln, Lieferanten und Kunden (Adreßverwaltung) automatisch fortgeschrieben. Das Rechnungsformular ist frei definierbar. Artikel haben eine max. zweizeilige Bezeichnung. Preise sind bei Rechnungserstellung veränderbar. Auch Artikel außerhalb der Lagerverwaltung können verarbeitet werden. PROFILA rechnet die ganze Rechnung aus mit Porto und auch mehreren MWSt-Sätzen. Die Fälligkeit wird aus dem Datum berechnet. Im Adreßbestand wird automatisch nach der vollständigen Anschrift, Rabattstufe und nach den Zahlungsbedingungen gesucht. Neue Adressen werden dem Bestand hinzugefügt. Es können Einzel- und Stapelrechnungen erstellt werden. Auf Wunsch entsteht ein Rechnungsausgangsbuch mit Fälligkeiten, ferner Etiketten und Postausgangslisten. Ausgangsrechnungen können in die Finanzbuchhaltung übergeleitet werden. Lieferscheine können an Filialen und Sammelrechnungen an die Stammhäuser gesandt werden.

PROFILA Lager

Der Artikelsatz umfaßt u.a. folgende Informationen: Artikelbezeichnung mit Nummer, Gruppe, Einheit, Bestell-Nummer, Spanne, Lagerort, EK, letzter EK, VK netto und brutto, MWSt-Satz, Umsatz, Deckung, verkaufte Einheiten, Bestand, Mindestbestand, Bestellvorschlag, Bestelldatum u. Menge, Lieferant und aktueller Warenwert. Jede denkbare Liste kann angelegt und erzeugt werden.

PROFILA Adreßverwaltung

Der Datensatz ist umfangreich mit frei definierbaren Feldern. Er wird ständig sortiert gehalten. Es können beliebige Listen unter unterschiedlichen Auswahlbedingungen erstellt werden. Adressen können auch an das Textprogramm z.B. für Werbeaktionen übergeben werden.

PROFILA Nebenbuchhaltung/Mahnung

Der Zahlungsverkehr mit Debitoren und Kreditoren wird abgestimmt und mit Buchungstext gegengebucht. Offene Posten und Kontostände können gelistet werden. Mahnbriefe können in drei Mahnstufen erzeugt werden.

PROTEXT, Programmsystem Textverarbeitung

Es ist leicht bedienbar, man arbeitet sich schnell ein. Auf Wunsch hat man eine Hilfe-Funktion auf dem Bildschirm. Auch ohne Formatierbefehle sieht man Briefe, wie von der Schreibmaschine geschrieben. Weitere Merkmale: Autom. Silbentrennung, Proportionalspacing individuelle Druckeranpassung, Fettschrift, Unterstreichnung und 10 freie Sonderfunktionen. Daneben Job-Verarbeitung, Kopf- und Fußzeilen, Seitenzähler, Randausgleich, Trennung, Zentrierung, Archivieren, Suchen und Ersetzen. Speicherung der Tabulatoren. Zeilenbreite bis 120 Zeichen mit horizontalem Rollen u.a.m..

PROTEXT Textkorrektur

Dieses Programm prüft auf Rechtschreibefehler. Der Grundwortschatz hat 1600 Worte. Er wird selbstlernend erweitert.

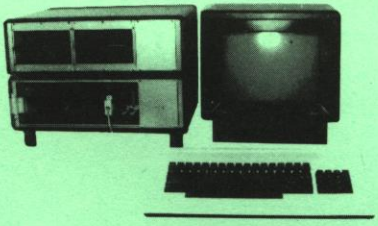
Wir senden Ihnen für Ihrem CBM gerne vollständige Unterlagen und Angebote.

GWK

GESELLSCHAFT FÜR TECHNISCHE ELEKTRONIK mbH.
HARDWARE SOFTWARE SYSTEMENTWICKLUNG

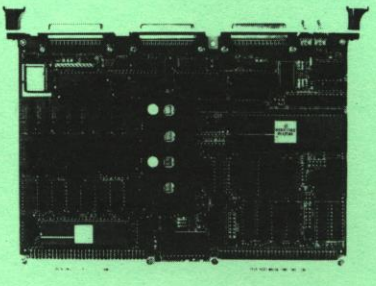
MICROCOMPUTER

FÜR DEN EINSATZ IN TECHNIK U. WISSENSCHAFT



GWK EURO BOARD COMPUTER SYSTEM

Modulares Europakarten System für die 8 Bit CPU s 6809 und 6502



SYS 68K VME

Modulares VME-BUS System mit 16 Bit CPU 68000

SYSTEMS 83
17.-21. Oktober
München
Wir stellen aus: Halle 18, Stand 18000

FORTUNE 32:16

Hochleistungs System CPU 68000 und UNIX Betriebssystem
Multiuser Multitasking

D 5120 Herzogenrath Asternstr. 2
Tel.: 02406/6035 Telex: 832109 gwk d

65xx MICRO MAG

COMPUTING · SOFTWARE · HOBBY

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

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

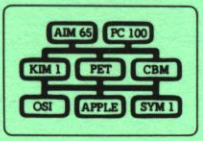
Bezugsbedingungen: Abonnement ab laufender Ausgabe für 6 Hefte DM 54,-- (Inlandsendpreis). Ausland/foreign via surface mail DM 59,--, USA air 26 Dollar. Abonnements laufen bis auf Widerruf mit Kündigungsmöglichkeit bis zu zwei Wochen vor deren Ablauf.

Nachliefermöglichkeiten: Hefte 1-6 und 7-13 sind als Buch nachlieferbar (s. Anzeige unten). Hefte 14-32 können alle als Einzelhefte zu DM 7,80/St. + DM 2,50 je Sendung geliefert werden. Einzelpreis bei 10 und mehr Heften je Sendung DM 6,--.

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

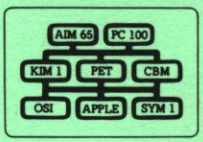
Leser-Service des Herausgebers

Das Buch 1-6 des 65.. MICRO MAG



230 Seiten, DM 26,--

Das Buch 7-13 des 65.. MICRO MAG



340 Seiten, DM 42,--

Thermokopf (Printerplatte für AIM 65 und PC 100
Artikel läuft aus. Lieferung nur solange Vorrat. DM 28,--

FORTH User's Guide
Rockwell-Handbuch für das FIG-FORTH des AIM 65. Mit der Erläuterung des Befehlssatzes auch für andere FORTH-Betreiber geeignet. Ca. 300 S., engl. DM 24,--

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

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

Cross-Assembler für Motorola 6809 DM 380,-- und für 6805 DM 320,-- + MWSt, inkl. Dokumentation (DM 10,--), beide unter FORTH des AIM laufend.