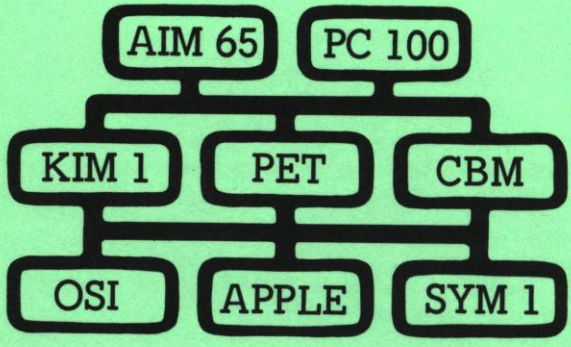
65xx MICRO MAG

COMPUTING · SOFTWARE · HOBBY

DM 9,50 Nr. 30 April 1983



Inhaltsverzeichnis

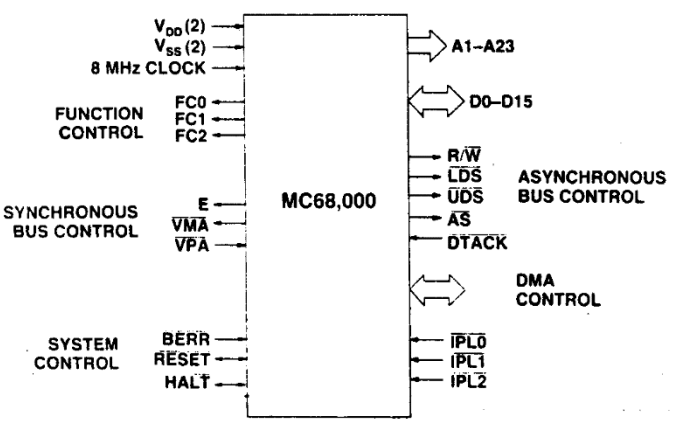
Wichtige Merkmale des MC 68000

Mikroprozessoren mit 16 Bit-Datenbus gewinnen zunehmend an Bedeutung. In Abweichung von der üblichen Darstellung in (englischen) Datenblättern und auch in Büchern, soll hier versucht werden, den MC68000 von Motorola aus den erkennbaren Grundideen seines Aufbaues (Designs) zu beschreiben und nur die Besonderheiten seines Befehlssatzes hervorzuheben. Wir gehen dabei von der Voraussetzung aus, daß der interessierte Leser ja sowieso ein Datenblatt beim Hersteller oder bei seinen Distributoren bezieht. Für das Studium dieser umfangreichen Vorlage hier ein Überblick:

Ein Mikroprozessor ist eine in der Zeit arbeitende Maschine, die durch einen Quarz oder einen Ersatzschwingkreis getaktet wird und die elektrische Signale aussendet, um binäre Programm- oder Dateninformation zu holen, zu verarbeiten und zu speichern. Bei den 8 Bit-Mikros waren wir es gewohnt, daß das gesamte System durch den Takt synchronisiert ist (synchrone Systembausteine - von den hier und da schon gegebenen Möglichkeiten der Taktdehnung einmal abgesehen).

Asynchrone Maschine mit Handshake

Der MC68000 ist eine asynchrone Maschine. Sie ist nicht darauf angewiesen, daß Programm- oder Dateninformation binnen einer streng bemessenen Taktzeit auf den Bus kommt. Sie wartet auf das Data Acknowledge-Signal (DTACK), das vom angesprochenen Speicher oder von der Peripherie abgegeben werden muß, geduldig auch sehr lange. DTACK (active low) ist damit auf dem Prozessorbus das Handshake dafür, daß die Botschaft beim R/W angekommen ist - ein wesentliches Merkmal für Programm- und Datensicherheit. Ein entwickeltes Prozessorsystem achtet mit externer Beschaltung darauf, daß nach einer überzogenen Antwortzeit das Signal Bus Error (BERR) gezogen wird, womit eine 'Exception', ein Interrupt eintritt, der einen eigenen Vektor auf eine Auffangroutine des Anwenders oder der Systemfirmware hat.. Dieses TIMEOUT ist damit eine Versicherung gegen die Ansprache nicht implementierter Programm- und Datenspeicher. Es ist sogar sichergestellt, daß nach einem zweiten Bus Error in dieser Behandlungsphase die CPU auf HALT geht. Das Abstürzen des Systems ist damit gar nicht mehr so einfach zu vollziehen.



Signalgruppen am MC 68000

Für die sog. 'synchronen' Bausteine (vorwiegend für die der 8 Bit 68xx und 65xx) ist eine von DTACK verschiedene Antwortweise vorgesehen, auf die wir im Zusammenhange des Interruptsystems eingehen werden (in den Folgeheften).

Die innere Registerbreite

Mit der Abb. 2 'Programmers Model' kommen wir auf die Maschinenregister. Wir finden 8 'Akkumulatoren' von 32 Bit Breite, nämlich die Datenregister D0 bis D7 sowie 9 gleich breite Register, die vorwiegend der direkten und indirekten Adressierung dienen, die Adreßregister A0-A6 mit einem ziemlich vollen Befehlssatz für die Rechenhaftigkeit. Daneben gibt es den System-Stackpointer A7' und den Anwender-Stackpointer A7. Alle Register können zur Indizierung (auch mit Indizierung und Offset) herangezogen werden. Im Statusregister (16 Bit) gibt es das Anwenderbyte mit den üblichen Anzeigen sowie das Systembyte, das die Maschine auf Supervisor-/Anwendermodus setzt, das den TRACE-Modus und die Interrupt-Prioritäten (7 Ebenen) bewacht .

Die Register sind 32 Bit breit, der Datenbus hat 16 Pin für die Modelle L4, L8, L10, L12 (für 4, 8, 10, 12,5 MHz), er ist für das erwartete Modell L20 32 Bit breit. Es gibt das eingeschränkte Modell 68010 mit vollkompatiblem Befehlssatz und 8 Bit Datenbus.

Der Adreßbus ist immer 23 Bit breit und gestattet die Ansprache von 8 Megabyte Wörtern, wohl mehr, als man in vielen Anwendungen braucht.

Transparenz der Signale

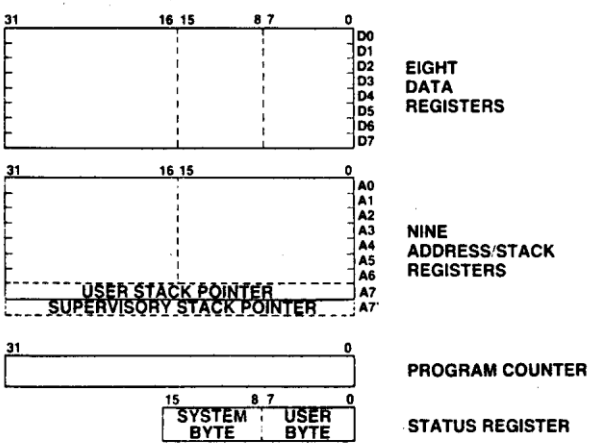
Es wurde bereits erwähnt, daß das Signal DTACK die asynchrone Maschine möglich macht. Folgende Signale dienen der Transparenz für Adreß- und Datenbus: R/W (Read/Write), LDS, UDS (Lower und Upper Data Strobe, Ansprache des unteren oder oberen Bytes im Wort), AS (Address Strobe, gültige angelegte Adresse).

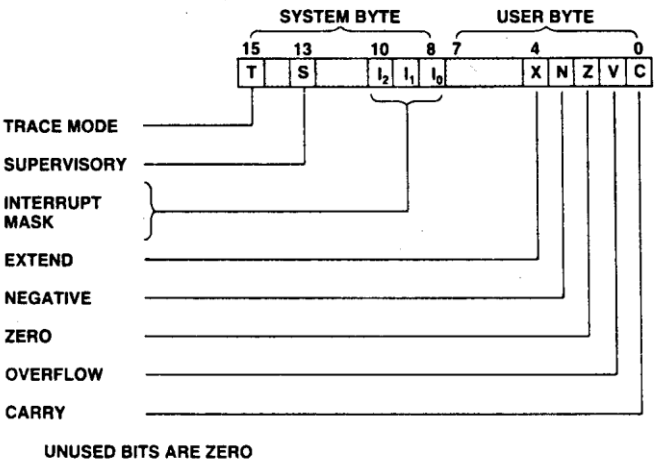
Der Prozessor gibt aber auch immer Auskunft, was er z.Zt. macht, und zwar über die Functions-Codes FC0-FC2, auf denen abzulesen ist, ob er im Supervisor-Mode oder im Anwendermodus ist und ob er in diesen Modes Programm- oder Datenwörter bearbeitet. Bei einem Interrupt gehen alle 3 Pins auf HIGH. Hiervon wird das Signal IACK (Interrupt Acknowledge) extern abgeleitet.

Weitere Pins dienen dem Umgang mit den 'älteren synchronen Bausteinen', nämlich VPA (Valid Peripheral Address), VMA (Valid Memory Address) und das Enable E. Diese werden im Zusammenhang mit den Interrupten behandelt werden.

Die CPU ist imstande, über den Pin RESET einen elektrischen RESET entgegenzunehmen. Der Befehl RESET setzt dagegen nur die Peripherie zurück, indem dieser Pin für einige Zeit von innen her auf Low gezogen wird. Die Übergabe der Buskontrolle für externe Einheiten im DMA (Direct Memory Access) ist über die Handshake-Pins Bus Request (BR), Bus Grant (BG) und das Eingabesignal BGACK (Bus Grant Acknowledge) geregelt.

MC68,000 PROGRAMMING MODEL





Drei Pins IPL0-IPL2 dienen zur Eingabe elektrischer Interrupts nach Prioritätsebene. Auch sie werden später ausführlicher beschrieben. Man darf aber auch hier schon darauf hinweisen, daß das Interruptsystem sehr ausgedacht ist, es gibt 'auto vectored' Interrupts, die durch Beschaltung sofort auf den spezifischen Interruptvektor und damit auf die Behandlungsroutine führen (sie arbeiten ohne weiteres 'polling'), sowie anwenderbestimmte 'vectored' Interrupts. Bei diesen gibt der interruptende Baustein nach einem dekodierten Interrupt Acknowledge (IACK) die Nummer des ihm in einem Register mitgeteilten Interrupts auf den Datenbus ab. Die CPU verzweigt nach der Erkennung (DTACK) sofort auf die vektorierte Behandlungsroutine. Das gewährleistet schnelle Bearbeitung. Klar ist auch, daß ein Interface im Programmverlauf verschiedene Vektor-Nummern abgeben kann z.B. für Einschalten/Ausschalten/Wiedereinschalten u.ä.. - Die Behandlung von Interrupts ist für VME-Bus-Systeme einheitlich elektrisch und einsichtig geregelt.

Der Befehlssatz

Der 68000 kennt folgende Datentypen:

1. Bit
2. Byte
3. BCD-Zahl
4. Word
5. Long Word (32 Bit)

Gruppen 1 und 3 wirken nur bei einigen wichtigen Befehlen, die anderen werden dem Assembler durch das Anhängen (suffix) von B, .W oder .L mitgeteilt. Man muß herausstellen, daß der 68000 bei den meisten Befehlen eine 2 Adreß-Maschine ist. Im ersten Operanden des Quelltextes teilt man die Datenquelle (Source) mit, im zweiten das Datenziel (Destination). Der Befehl ADD D0,D1 z.B. addiert den Inhalt des Datenregisters D0 zum bisherigen Inhalt des Registers D1 mit Ergebnis in D1. Es sind fast alle gemischten Sources und Destinations zugelassen, Register to Register, Memory to Register Register to Memory, Memory to Memory.

Der Befehlsatz ist damit vielseitig und beschränkt sich auf übersichtliche Grundbefehle, mit verschiedenen Adressierungsarten immediate, m.effektiver Adresse, mit Adresse in einem Register, mit Indizierung durch ein Register, Indizierung mit Offset und soweit sinnvoll natürlich auch alles indirekt adressiert. Insbesondere ist hervorzuheben, daß eine speicherplatz- oder adressenunabhängige Adressierung relativ zum Programmzähler möglich ist. Das Modul führt dann an jeder

Stelle aus und kann mühelos eingebunden werden. Weitere relokative Tricks sind über eine Benutzung/Indizierung mit einem Basis Adreßregister möglich.

Neben den bekannten Grundbefehlen (Laden, Speichern, Vergleich, arithmetische, logische, Verzweigungs und Sprungbefehle) sind folgende Verarbeitungsmöglichkeiten gegeben:

Dezimale Addition, Subtraktion, Negation
Bit-Manipulation: Testen, Setzen, Löschen, Invertieren
Verschiebebefehle mit Zahl der Verschiebeschritte
Binäre Multiplikation 16*16 Bit, Division 32/16 Bit
Load oder Push effektive Adresse
Move Multiple Register List, Block-Move von Bytes
Teste Bedingung dekrementiere und verzweige für Schleifenkonstrukte
Teste und setze Bit 7 im Operanden, prüfe Operanden auf Null

Befehle und Bedingungen, die eine 'Exception , einen Interrupt auslösen

TRAP mit Vektor Nummer. Einleitung einer exception. Mit dieser Instruktion können noch nicht implementierte Befehle (Micro-Codes) simuliert werden.
Trap On Overflow
Trap-Bedingungen treten auch bei versuchter Division durch 0 ein,
ferner bei Operationscodes mit 1010 und 1111
bei Benutzung privilegierter Befehle des Supervisors im Anwendermodus
beim Befehl CHK wenn ein Register einen festgelegten Wert überschreitet.

Ein besonderer Exception Vector wurde ferner für 'Adreßfehler' geschaffen. Er wird angesprungen, wenn ein Word-Operand auf einer ungeraden Adresse angesprochen ist.

Zur bequemen Diagnose werden bei Bus- und Adreßfehlern mehrere Werte auf dem Systemstack abgelegt, Programmzähler, Status vor dem Ereignis, Befehlsregister-Inhalt, zugegriffene Adresse sowie zusäztliche Information über den abgebrochenen Buszyklus.

Der Prozessor kann ferner in einen TRACE-Modus versetzt werden. Nach jeder einzelnen Befehlsausführung wird dann ein spezifischer Vektor zur Weiterbehandlung angesprochen, der vom Systemprogramm für entsprechende Anzeigen genutzt wird.

Zusammenfassung

Die Erwähnung der oben genannten wichtigsten Merkmale des 68000 zeigt, daß von der Hardware und vom Befehlssatz her neue Wege beschritten wurden. Es ist eine asynchrone Maschine, die im Handshake-Betrieb arbeitet und die auf Ausnahmebedingungen elektrischer oder programm-mäßiger Art in einer transparenten Weise eindeutig reagiert, und zwar durch ein gut überlegtes System von 256 Ausnahme-Vektoren. Mit diesen Merkmalen wird man die 68000 sowohl bei der Programmerprobung (TRACE, Adreßfehler, nicht antwortende Einheiten), wie auch im praktischen Betrieb in der Mehrzahl der Fälle definiert auffangen können. Mit ihren Traps für noch nicht implementierte und für wirklich 'illegale' Befehle sind Aufwärtskompatibilität und zusätzliche Ablaufsicherheit gewährleistet.

Der Entwurf der CPU darf damit als hervorragend durchdacht bezeichnet werden. Mit der inzwischen schon als ausreichend zu bezeichnenden Dokumentation (s.u.) ist ferner eine Einarbeitung gut möglich, vor allem wenn man von 68xx oder 65xx aufbaut.

Andere wichtige Merkmale stehen nicht im Datenblatt: Second Sources (Zweitlieferanten für die CPU) mit Maskenaustausch sind Hitachi, Mostek, Philips/Signetics, Thomson Efcis und Rockwell, die auch Eigenentwicklungen zur 68000-Familie betreiben und so die Konfigurationsmöglichkeiten verbessern.

Es muß auch der VME-Bus für den 68000 erwähnt werden. Er stellt eine strenge elektrische und mechanische freiwillige Norm dar die von den Herstellern Motorola, Mostek, Philips und Thomson unterstützt und eingehalten wird. Jetzige und künfige boards dieser Hersteller sind daher miteinander kombinierbar Diese Norm beinhaltet nicht nur die Beschaltung der VG Stecker an den

Ports, sondern auch den mechanischen Aufbau bis hin zum Sitz der Bohrlöcher und der Schraubengröße. Wir werden auch darauf eingehen.

Ferner: In den Wintermonaten nahmen viele hundert Designer aus anderen Firmen als den erwähnten an VME-Bus-Seminaren teil. Dieses Interesse spricht nicht nur dafür, daß der 68000 ein Industriestandard wird, sondern daß auch der zu ihm gehörende VME Bus Akzeptanz findet und auch von diesen Designern um Module bereichert wird.

Literatur:

Datenblatt MC6800L4 bis L12, Motorola Inc., 1982
MC68000 16 Bit User's Manual (Motorola)
MC68000 Systems Cross Macro Assembler Reference Manual (Motorola)
SC68000 16 Bit Mikroprozessorfamilie, Datenblätter von Valvo
MC68008 16 Bit Microprocessor With 8 Bit Data Bus, Voraus-Datenblatt von Motorola, Nov. 82
MC68010 16 Bit Virtual Memory Microprocessor, Motorola-Datenblatt, Dez. 82
VMEbus Specification Manual Rev. B, Signetics, Aug. 82
Kane, G., Hawkins, D., Leventhal, L.: 68000 Assembly Language Programming. Verlag Osborn McGraw-Hill, Berkely, CA. 1981, ISBN 0-931988-62-4
Koch, J., VALVO GmbH Hrsg.: Der 16-bit-Mikroprozessor SC68000 - Eigenschaften. Hamburg 1982, ISBN 3-87095-255-5, ca. DM 20,-
Ein weiteres VALVO-Buch soll im April 1983 erscheinen.

Vorstehende Abbildungen wurden der Motorola-Dokumentation entnommen.

R.L. □□

Interfacebaustein PI/T MC 68230

In Heft 7 dieser Zeitschrift wurde die VIA 6522 als ein intelligenter Peripheriebaustein für die Leserschaft hilfreich dargestellt. Es ist an der Zeit, den 68230 zu besprechen, der von Motorola als 'Parallel Interface/Timer' bezeichnet wird und der speziell für 68000-Systeme geschaffen wurde. Man wird ihn aber auch in anderen Systemen benutzen können, zumal er unter einer eigenen Clock asynchron betrieben werden kann, die von der CPU-Clock abweicht. Überlappungsprobleme mögen sich nur dann ergeben, wenn erstere schneller als die des Systems ist. Und natürlich muß die Impulsspezifikation eingehalten werden.

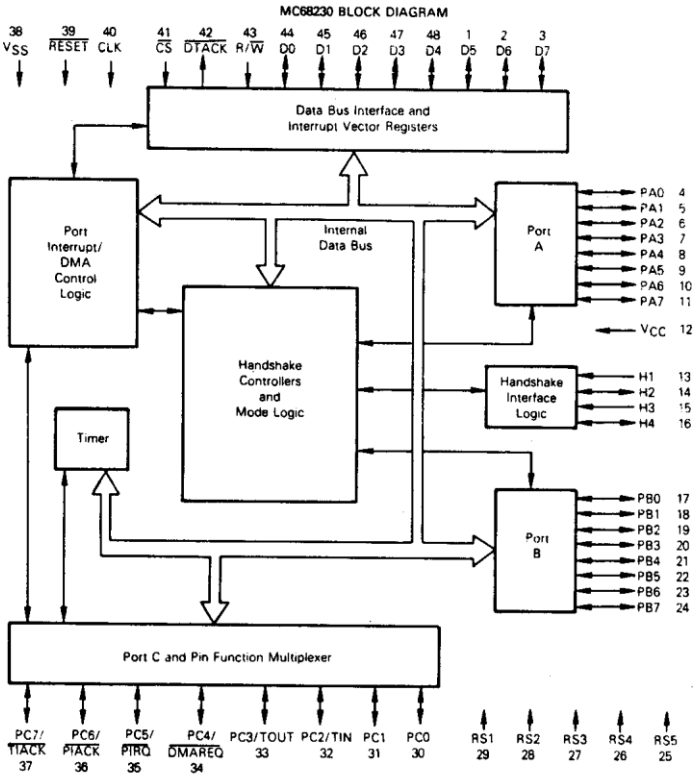
Zunächst zum Blockdiagramm der Schaltung:

Zuoberst sehen wir den Datenbusanschluß D0-D7, den Chip Select, das R/W (Read/Write) und das DTACK-Antwortsignal im Handshakebetrieb. Man beachte, daß das Datenbus-Interface 8 Bit breit ist und nicht 16, wie von manchen Anwendern schon gewünscht wurde.

Auf der Unterseite der Abbildung sind die prozessorseitigen Eingangsleitungen RS1-RS5 bezeichnet, die mit dem unteren Adreßbus verbunden werden und die den 'Register Select' bewirken. Die Steuerung des Bausteines erfolgt wie bei der VIA und bei anderen durch initialisierendes Beschreiben der Register. Von den 32 möglichen Registern sind 23 belegt, und zwar 14 für die Steuerung der Ports A, B und C sowie 9 für die Kontrolle des Timers.

Steuerung der Ports

Die Ports A, B und C können in gewohnter Weise betrieben werden: Initialisierendes Beschreiben der Datenrichtungsregister und dananch Schreiben in die Ports (genannt PADR, PBDR, PCDR, Port Daten Register). In manchen Konfigurationen kann man also eine E/A in 24 Bit betreiben. Hinzu kommen die Handshake-Leitungen H1 bis H4, die auch 'von Hand' in ihrer Funktion gesteuert werden können zur Ausgabe eines High/Low-Pegels, als Impulsausgabe zur Quittung oder als Eingabepins zur Erkennung abfallender oder steigender Impulsflanken bzw. zur Erkennung der Level High oder Low. Bei diesen vier Pins gehört natürlich auch das Interrupt Enable dazu. Das Interruptsystem ist sehr ausgeklügelt und wird noch näher dargestellt. Soweit diese Pins nicht für Handshake oder Interrupt benutzt werden, erhöhen sie die Zahl der Port-Pins also auf mögliche 28.



Port C stellt einen Sonderfall dar. Die Pins PC0 und PC1 sind ständig normale Port-Pins. Die höherwertigen Pins können entweder auch normale Port-Pins sein oder sie werden als Multifunktionspins für das Interruptsystem der Ports/des Timers (mit Handshake) oder für einen über den Peripheriebaustein gesteuerten zweiten externen und schnellen Datenbus im DMA-Verfahren (DMAREQ) benutzt. Alles wird natürlich über die Register auf dem Chip gesteuert. Im Timerbetrieb ist auch die Ausgabe von Rechteckimpulsen an Pin PC3/TOUT möglich, wie wir sie an der VIA 6522 an Pin PB7 kennen.

Das Datenblatt des Bausteines umfaßt 32 eng bedruckte Seiten, deren Inhalt hier natürlich nicht vollständig wiedergegeben wird. Es kommt mehr darauf an, die Betreibungsmöglichkeiten so zu beschreiben, daß der Leser sich ein erstes Bild machen kann.

Im Voraus ist zu bemerken, daß der PI/T ein sehr leistungsfähiges Subsystem für die Peripherie bildet und daß er den bisherigen Interfacebausteinen der Familie artverwandt ist. Er wird entweder mit der System-Clock von 8 MHz betrieben oder mit einer externen asynchronen Clock. Das Signal DTACK sorgt für die Synchronisierung zum System.

Für die Ports A und B gibt es die Betreibungs-Modes 0 bis 3 für 8 oder 16 Bit-Transfers in der E/A. Dazu kommen Submodes (Spielarten) für die Funktion der Handshake-Pins H1-H4 mit Interrupt Enable oder ohne.

Es wird unterschieden zwischen dem 'unidirektionalen Modus' (das Datenrichtungsregister bestimmt die Richtung des Datentransfers) und dem 'bidirektionalen Modus', bei dem die Handshake-Pins H1-H4 bestimmen, in welcher Richtung der Datentransfer erfolgt. Der Betriebszustand kann also von innen her (CPU) oder von außen (DMA) 'umgeswapt' werden. Es ergeben sich damit schnell ausführende Möglichkeiten für einen 'zweiten Datenbus'.

Das Timerkonzept

Der PI/T enthält einen in 24 Bit geführten Timer. Dazu kommt ein Vorteiler (Prescaler) von 32 Impulsen, der in 3 Betriebsarten (CPU-Clock, externe Clock an TIN) benutzt werden kann und der z.B. bei der Ereigniszählung nicht benutzt wird.

Der Timer kann per Programm oder von außen her ein- und ausgeschaltet/unterbrochen werden. Es gibt weitere Optionen: Wie bei der VIA 6522 kann er jeweils aus Vorspeichern (3 Byte) nachgeladen werden, wenn er 00 erreicht hat, oder er wird mit FFFFFF nachgeladen (zur späteren Bestimmung der verstrichenen Zeit). Natürlich kann man die Zählerinhalte unterwegs lesen, exakt aber nur beim Halten des Timers.

Das Interrupt-System des PI/T

Für die Handshake-Pins H1-H4 sowie für den Timer kann man Interrupt zulassen oder auch nicht. Zwischen den vier Quellen H1-H4 kann man die Priorität der Bearbeitung ordnen, und zwar durch das PSRR (Port Service Request Register). Bei einem mit allen Signalen in das System integrierten PI/T, der über die Multifunktions-Pins des Port C seinen Interrupt Service Request herausgibt, kann die Nummer des Interruptvektors (4 für die Ports, eine für den Timer) von der CPU am Datenbus abgelesen werden, wenn auch die Signale IACK, und A1-A3 in entsprechend dekodierter Form auf PIACK bzw. TIACK zugeleitet werden (vectored interrupt). In einer anderen Art der externen Beschaltung (die weniger zu empfehlen ist) kann man unter Verwertung der für 'synchrone' Interfaces (68xx, 65xx) gedachten Signale VMA, VPA und E an der CPU auch die sog. 'auto vectored' Interrupts benutzen. In jedem Falle erfolgt eine gradlinige Vektorierung in der CPU auf die Behandlungsroutine des Anwenders, ohne daß ein weiteres 'polling' (Abfrage) nötig ist.

Das VME-Bus-Prinzip sieht vor, daß eine höher prioritisierte Einheit im 'daisy chain' (Gänseblümchen-Kette) niedere Interrupts solange vom IACK (Interrupt Acknowledge) abschneiden kann, bis der höhere Interrupt abgearbeitet ist (Sperrung und Durchleitung von IACK auf dem VME-Bus).

Zusammenfassung

Der PI/T 68230 läßt eine sehr schnelle Bearbeitung der Peripherie in Ports und Timer zu. In den meisten Fällen wird man seine auf 8 Datenleitungen beschränkte Architektur akzeptieren können. Tricks sind mit externer Beschaltung natürlich durch das Parallelschalten von Bausteinen möglich, um zB. die Peripherie in Wortbreite simultan zu steuern. Bemerkenswert ist, daß Datentransfers 'double buffered' sind, d.h. im unidirektionalen Modus können 2 Informationen in den Port-Latches zwischengespeichert sein, so daß sich Hol- oder Abnahmezyklen der CPU mit Abnahme- oder Bereitstellungszyklen der Peripherie überlappen können, wobei in den Registern Transparenz gewahrt ist, 'was noch geht' oder was noch zu tun ist. In späteren Heften werden typische Programmsequenzen für die Steuerung des PI/T enthalten sein.

R.L. □□

Dipl.-Ing. Rüdiger Wollenberg und Dipl.-Phys. Artur Redder

SCREEN

Bildschirmeditor für AIM 65

1.0 Einleitung

Der Editor des AIM 65 bzw. PC 100 ist ursprünglich konzipiert worden, um Assemblerprogramme zu erstellen. Er genügt daher nur geringen Anforderungen. Viele Betreiber haben ihren AIM aber inzwischen mit einem Bildschirmeditor und mit einem Drucker ausgerüstet und besitzen damit sämtliche Komponenten eines Textverarbeitungssystems. Leider erfüllt der Standard AIM Editor die Ansprüchen eines komfortablen Systems nicht. Aus diesem Grunde sind auch im Rahmen dieser Zeitschrift Ansätze zur Erweiterung des Befehlssatzes vorgenommen worden. Diese nutzen aber vornehmlich eine Zeilennumerierung des Schriftsatzes aus.

Von allen denkbaren Möglichkeiten des Editing ist aber die des Bildschirm-Editors am schönsten und komfortabelsten. Auf einem Bildschirm ist ein Fenster des Textbuffers zu sehen. Über Kontrollkommandos läßt sich eine bestimmte Textstelle anfahren und dort z.B. Zeichenveränderungen, -einschiebungen, -löschungen und auch Zeilenlöschungen und -einschiebungen vornehmen. Weil ständig die Umgebung der aktuellen Zeile zu sehen ist und die Cursorsteuerung sich hervorragend zur vertikalen und horizontalen Positionierung eignet, wird ein äußerst hoher Komfort erreicht.

In dieser Veröffentlichung wird ein Bildschirmeditor vorgestellt, der zum AIM Texteditor vollkompatibel ist und der über die folgenden Eigenschaften verfügt:

Durch Assemblerprogrammierung ist Unabhängigkeit von Compilern und Interpretern gegeben.

Es wird ein Fenster des Textbuffers auf dem Monitor dargestellt, das ständig die acht Zeilen vor der aktuellen Zeile, diese und die sieben darauffolgenden anzeigt.

Die Beschreibung bezieht sich auf ein Videointerface, das im Speicherraum des AIM 65 liegt (memory mapped) und 16 Zeilen * 64 Zeichen umfaßt. Die Autoren bauten eine Schaltung nach /2/ nach, die mit dem CRT-Controller 6545 bzw. 6845 arbeitet. Andere Interfacekarten sind softwaremäßig jedoch leicht anpaßbar, weil durch eine strukturierte Programmierung erreicht wurde, daß nur wenige Unterprogramme geändert werden müssen.

Die Kommandosteuerung des Screeneditors erfolgt über Control-Kommandos, d.h. durch gleichzeitiges Drücken der CTRL- und einer alphanumerischen Taste. Selbstverständlich wurde die Tastenbelegung den Befehlskürzeln des AIM-Editors angepaßt. CTRL I bedeutet beispielsweise die Einfügung einer Zeile.

Lediglich für die Cursorsteuerung wurden die vier rautenförmig angeordneten Tasten U, J, N und H gewählt, weil dadurch eine leichtere Bedienbarkeit erreicht wird. In der Testphase wurde die Rechts- und Linksbewegung über CRTL-R und CTRK-L durchgeführt. Dies ist zwar einprägsam, aber ergonomisch ungünstig, weil die Cursorbewegung nach rechts mit der linken Hand ausgeführt wird und umgekehrt.

Nach jeder Betätigung der Return-Taste, einer vertikalen Cursorsteuerung oder des Zeilen-Insert wird eine Einspeicherung der aktuellen Bildschirmzeile vorgenommen. Vorher werden alle links stehenden Blanks unterdrückt. Bei leeren Zeilen wird genau ein Blank übrig gelassen, um keine Kollisionen mit dem AIM-Editor zu bekommen. Ein versehentliches Betätigen der Escape-Taste führt daher auch nicht zum Verlust des gesamten Schirminhaltes.

Die horizontale Cursorposition wird, soweit es geht, nicht verändert. Nur bei Abschluß einer Zeile mit Return oder einem Zeileninsert erschien es angebracht, sich auf das erste Zeichen einer neuen Zeile zu sezen. Insgesamt wird dadurch die Programmqualität weiter gesteigert.

Trotz der Verwendung eines Videointerface, das 64 Zeichen pro Zeile darstellen kann, wird die maximale Zeilenlänge von 61 inklusive Return beibehalten, weil der Original-Editor diese Beschränkung vorgibt.

Der AIM stellt für die Textbearbeitung eine Reihe von Routinen zur Verfügung, von denen die mit REPLAC benannte eine zentrale Bedeutung besitzt. Es wurde versucht, diese Hilfsprogramme weitestgehend auszunutzen.

Für die unten beschriebenen Bedienfunktionen werden lediglich ca. 850 Byte Speicherplatz benötigt, wenn der redigierte Monitor REMON 3.2 der Verfasser benutzt wird. Die Umschaltung der Groß-/Kleinschreibung, das Tastenrepeat, der deutsche Zeichensatz und die Kommandoerweiterung sind dort schon vorbereitet und brauchen nicht mehr programmiert zu werden.

Unter Verwendung des REMON-Betriebssystemes /1/ ist der Einsprung in den Screneditor und ein Rücksprung in den AIM-Monitor ohne Veränderung des DILINK-Vektors möglich.

Durch den modularen Aufbau kann zunächst eine Befehls-Untermenge augebaut werden, die man dann schrittweise komfortabler erweitert.

2.0 Hardwareumgebung

Das Videointerface der Betreiber muß Memory Mapped sein, d.h. daß der Bildwiederholspeicher vom Mikroprozessor wie ein ganz gewöhnliches RAM beschrieben und gelesen werden kann. Weiterhin muß das Interface über Befehle zur Placierung des Cursors verfügen. Ein Bildschirmformat von 64 Zeichen * 16 Zeilen sollte mindestens verfügbar sein. - Erst durch die Verwendung des REMON /1/ ist eine einfache Anbindung des Screeneditors an den AIM-Editor möglich geworden. Gleichzeitig wird die Bedienungsqualität noch gesteiegert, weil die Programmierung von so nützlichen Routinen wie Tastenrepeat, Groß- und Kleinschreibung und des deutschen Zeichensatzes entfällt.

3.0 Bedienungsfunktionen

Die Kommandos des Screeneditors werden mittels Controlcodes eingegeben. Dabei ist gleichzeitig die CTRL—Taste und eine alphanumerische Taste zu drücken. Alle ASCII-Zeichen kleiner als $20 und das Delete $7F werden als Befehle interpretiert. Der Rest sind druckbare Zeichen. Controlcodes, die momentan nicht belegt sind, werden ignoriert.

3.1 Cursor-Kommandos

TEXT—CURSOR UP CTRL-U
Bewegt den sichtbaren Text um eine Zeile nach unten und damit den Cursor relativ nach oben. Die horizontale Cursorposition wird beibehalten. Die TOP-Zeile wird nicht überschritten.

TEXT-CURSOR DOWN CTRL-D bzw. CTRL-N
Bewegt den sichtbaren Text um eine Zeile nach oben und damit den Cursor relativ nach unten. Die horizontale Cursorposition wird beibehalten. Nach Überschreitung der Bottom-Zeile werden automatisch leere Folgezeilen erzeugt.

MOVE CURSOR RIGHT CTRL-J
Bewegt den Cursor um eine Stelle nach rechts, überschreitet aber nicht den rechten Rand.

MOVE CURSOR LEFT CTRL-H
Bewegt den Cursor um eine Stelle nach links, überschreitet aber nicht den linken Rand.

CURSOR TO THE RIGHT CTRL-R
Setzt den Cursor an den rechten Rand.

3.2. Zeilen-Kommandos

CARRIAGE RETURN CR bzw. CTRL-M
Übernimmt die aktuelle Zeile vollständig nach vorheriger Reduktion von rechtsbündigen Leerzeichen und schiebt den Text um eine Zeile nach oben, bzw. den Cursor relativ nach unten. Der Cursor wird auf die erste Position der neuen Zeile gesetzt.

INSERT A LINE CTRL-I
Übernimmt die jetzige Zeile vollständig und erzeugt eine Leerzeile vor dieser, die zur aktuellen wird. Der Cursor wird auf die erste Position der Leerzeile plaziert.

DELETE A LINE CZRL-K
Löscht die aktuelle Zeile. Die horizontale Cursorposition wird beibehalten.

DUPLICATE LINE CTRL-D
Dupliziert die aktuelle Zeile. Der Cursor bleibt an der selben Position.

SPLIT A LINE CTRL-S
Der Zeileninhalt ab der Cursorposition wird in die nächste Zeile gebracht. Der Cursor bleibt auf seiner Position.

LINK A LINE CTRL-L
Die aktuelle Zeile wird ab der Cursorposition mit dem Inhalt der nächsten Zeile aufgefüllt. Wird aus der Folgezeile wegen des Datentransports eine Leerzeile, so wird diese gelöscht. Der Cursor bleibt auf seiner Position.

3.3 Spezial-Befehle

TEXT-CURSOR TO TOP CTRL-T
Nimmt die TOP-Zeile als aktuelle Zeile. Behält die horizontale Cursorposition.

TEXT-CURSOR TO BOTTOM CTRL-B
Nimmt die letzte Textzeile als aktuelle Zeile. Behält die horizontale Cursorposition bei.

QUIT SCREENEDITOR CTRL-Q
Rückkehr zum normalen AIM 65-Editor. Setzt die Cursorzeile von der Bildmitte auf die untere Zeile.

DELETE $7F
Löscht das zuletzt eingegebene Zeichen und bewegt den Cursor um eine Position nach links. Geht nicht über den linken Rand hinaus.

ERASE A CHARACTER CTRL-E
Löscht das Zeichen, das durch den Cursor markiert ist und schiebt die Zeichenkette rechts vom Cursor um eine Stelle nach links. Die letzte Stelle der Zeile wird mit dem Leerzeichen aufgefüllt.

TOGGLE INSERT-FLAG CTRL-X
Invertiert das Insert-Flag. Bei gesetztem Flag wird bei der Eingabe eines druckbaren Zeichens die Textkette rechts vom Cursor um eine Stelle nach rechts versetzt. Zeichen, die über den rechten Rand hinausgeschoben werden, sind verloren. Das Insert-Flag wird lediglich für den Aufruf des Bildschirmeditors gelöscht.

TOGGLE CAPS LOCK CTRL-A
Umschaltung des Groß-/Kleinschreib-Flags, vgl. /1/.

3.4 Tabulator-Befehle

ZERO TABULATORPOSITION CTRL-Z
Acht Tabulatorpositionen werden mit voreingestellten Werten besetzt. Von einer automatischen Initialisierung beim Aufruf des Bildschirmeditors wurde abgegangen, um beim wiederholten Wechsel von der normalen Editorebene zum Screeneditor und zurück die eingestellten individuellen Tabulatorpositionen nicht zu verlieren.

GENERATE TABULATOR CTRL-G
Die gegenwärtige horizontale Cursorposition wird zur Tabulatorposition, wenn noch Tabulatormerker frei sind.

CLEAR TABULATOR CTRL-C
Die Tabulatorposition, die der gegenwärtigen Cursorstelle entspricht, wird gelöscht.

TABULATOR FORWARD CTRL-F

Die nächste Tabulatorposition der Bildschirmzeile wird angesprungen. Existiert keine frühere mehr, verharrt der Cursor auf der Stelle.

4.0 Programmaufbau und -listing

Das Programm gliedert sich in vier Teile:

1. Anbindung des Sreeneditors an den AIM Text-Editor.
2. Befehlsdekodierung der Screenkommandos
3. Routinen zur Zeichenumspeicherung im Bildwiederholspeicher und zum Datentransport zwischen AIM Textbuffer und der Screen.
4. Routinen zur Initialisierung und Cursorsteuerung des Videointerface.

Lediglich die Unterprogramme des vierten Abschnittes, die auf die genaue Hardwareumgebung des Videointerface eingehen, müssen von den Betreibern auf die eigenen Bedürfnisse angepaßt werden.

Literaturhinweise

/1/ Wollenberg R., Redder, A.: REMON - Redigierter Monitor. 65xx MICRO MAG Nr. 28, S. 3
/2/ Zimmermannn M.: Gedanken zum Video-AIM. 65xx MICRO MAG, Nr. 8, S. 16.

```
0000          0000 ;*****************************************************
0000          0001 ;*
0000          0002 ;*   SCREENEDITOR   V 3.6   21.02.1983
0000          0003 ;*   WOLLENBERG/REDDER
0000          0004 ;*****************************************************
0000          0005 ;*   DEFINITION DER MONITOR - UND EDITORROUTINEN
0000          0006 SAVNOW=$F934  ;Save current line
0000          0007 ATTOP  =$F8DB ;Current line at top? (C set if so)
0000          0008 DOWN1  =$F6E3 ;1 line down in memory
0000          0009 ATBOT  =$F8E9 ;Current line at bottom? (C set if so)
0000          0010 UPNO   =$F709 ;1 line up in memory
0000          0011 UP1    =$F713 ;1 line up in memory
0000          0012 RESNOW=$F8D0  ;Restore current line adress
0000          0013 REPLAC=$F93F  ;Replaces new line for old one
0000          0014 TOPNO  =$F8BC ;Set current line to top
0000          0015 SETBOT=$F8C5  ;Set current line to bottom
0000          0016 PLNE   =$F727 ;Print current line
0000          0017 CFLG   =$F8B2 ;Set flag for C-command
0000          0018 KIFLG  =$F8B6 ;Clr K or I command flag
0000          0019 JUMP   =$A47D ;Simulate JSR (COMMAND)
0000          0020 DIBUFF=$A438  ;Display buffer
0000          0021 READ   =$E93C ;Read 1 character from KB/TTY
0000          0022 CRLOW  =$EA13 ;CRLF to TTY or D/P
0000          0023 QM     =$E7D4 ;Outputs a '?'
0000          0024 ;*****************************************************
0000          0025 ;*   DEFINITION DER VIDEOINTERFACE-VARIABLEN
0000          0026 VIDADR=$A300  ;Adressregister of CRT-controller
0000          0027 VIDREG=$A301  ;Register of CRT-Controller
0000          0028 VID8   =$9A00 ;Absolute adress of TV-line 8
0000          0029 VID9   =VID8+LINLEN ;Absolute adress of TV-line 9
0000          0030 CUR8   =$200 ;Relative adress of TV-line 8
0000          0031 CUR15  =$3C0 ;Relative adress of TV-line 15
0000          0032 LINLEN=64  ;# of TV-characters/line
0000          0033 LINNBR=16  ;# of TV-lines
0000          0034 WIDTH  =60 ;# of usable characters/line
0000          0035 CR     =$D ;ASCII of carriage return
0000          0036 TABMAX=8  ;Max. Tabulationpositions
0000          0037 ;*****************************************************
```

```
0000          0038 ;*   ZERO-PAGE-VARIABLEN
0000          0039        *=$D1
00D1          0040 TABNR  *=*+1 ;Aktuelle Tabulatoranzahl
00D2          0041 TABBUF *=*+8  ;Tabulatorbuffer
00DA          0042 TVLIN  *=*+1 ;Current TV-Line
00DB          0043 CURADR *=*+2  ;Cursor-adress absolut
00DD          0044 CURREL *=*+1  ;Cursor-adress relativ
00DE          0045 IFLG   *=*+1 ;Insert-flag
00DF          0046        * =$DF
00DF          0047 NOWLN  *=*+2 ;Current line
00E1          0048 BOTLN  *=*+2 ;Last active, so far
00E3          0049 TEXT   *=*+2 ;Limits of buffer (start)
00E5          0050 END    *=*+2 ;Limits of buffer (end)
00E7          0051 SAVE   *=*+2 ;Used by REPLAC
00E9          0052 OLDLEN *=*+1  ;Original length
00EA          0053 LENGTH *=*+1  ;New length
00EB          0054 ;*****************************************************
00EB          0055 ;*   EDITORERWEITERUNG
00EB          0056        * =$A40C ;Einsprung für REMON (MIKRO MAG 2B)
A40C 00C0     0057        .WOR EXEDIT
A40E          0058        * =$C000
C000 C953     0059 EXEDIT CMP #'S'  ;Screen-Aufruf?
C002 F004     0060        BEQ SCREEN
C004 20D4E7   0061 FEHLER JSR QM  ;Decode other commands or display '?'
C007 60       0062        RTS
C008          0063 ;*****************************************************
C008          0064 ;*   SCREENEINTRITT
C008 A000     0065 SCREEN LDY #0  ;Cursor to CUR8
C00A 206AC3   0066        JSR CURR1
C00D 84DE     0067        STY IFLG ;I-Flag off
C00F 20D5C1   0068 SCREE1 JSR BILD  ;New screen
C012 203CE9   0069 SCREE2 JSR READ  ;Read new screen-command
C015 201BC0   0070        JSR SCREE3 ;Decode it
C018 4C12C0   0071        JMP SCREE2
C01B C920     0072 SCREE3 CMP #$20  ;Cntl-code?
C01D B051     0073        BCS DELETE ;No
C01F 0A       0074        ASL A ;Yes, but which one
C020 AA       0075        TAX
C021 BD30C0   0076        LDA SCNTAB,X ;Enter command-adress
C024 8D7DA4   0077        STA JUMP
C027 BD31C0   0078        LDA SCNTAB+1,X
C02A 8D7EA4   0079        STA JUMP+1
C02D 6C7DA4   0080        JMP (JUMP) ;Execute screen-command. End with RTS
C030 9CC0     0081 SCNTAB .WORD RETURN, GROSKL, BOTTOM, TABCLR, CURDWN, ERASE
C032 EEC0     0081
C034 A1C1     0081
C036 48C1     0081
C038 A2C0     0081
C03A ADC1     0081
C03C 6AC1     0082        .WORD TABFOR, TABGEN, CURLEF, LININS, CURRIG, LINKIL
C03E 19C1     0082
C040 FAC0     0082
C042 BBC0     0082
C044 F7C0     0082
C046 CCC0     0082
C048 E1C0     0083        .WORD LINK, CARRI, CURDWN, DUPLIC, TABPRE, QUIT
C04A 9DC0     0083
C04C A2C0     0083
```

```
C0B5 20E3F6  0128        JSR DOW1 ;No, 1 line down in memory
C0B8 4CD5C1  0129 CURUP2 JMP BILD  ;New screen
C0BB         0130 ;*********************************************************
C0BB         0131 ;*   INSERT A LINE          CNTL I
C0BB 2070C2  0132 LININS JSR RESTOR  ;Restore TV-line
C0BE A000    0133        LDY #0 ;Cursor to CUR8
C0C0 206AC3  0134        JSR CURR1
C0C3 2064C2  0135        JSR LINBLK ;Fill line with blanks
C0C6 2084C2  0136        JSR INSERT ;Insert the line
C0C9 4CD5C1  0137        JMP BILD ;New screen
C0CC         0138 ;*********************************************************
C0CC         0139 ;*   DELETE A LINE          CNTL K
C0CC 201BC3  0140 LINKIL JSR KILL  ;Kill the line
C0CF 4CD5C1  0141        JMP BILD ;New screen
C0D2         0142 ;*********************************************************
C0D2         0143 ;*   DUPLICATE LINE         CNTL O
C0D2 2070C2  0144 DUPLIC JSR RESTOR  ;Restore TV-line
C0D5 2084C2  0145        JSR INSERT ;Insert the line
C0D8 4CD5C1  0146        JMP BILD ;New screen
C0DB         0147 ;*********************************************************
C0DB         0148 ;*   SPLIT A LINE           CNTL S
C0DB 2094C2  0149 SPLIT  JSR SPLIN ;Split the line
C0DE 4CD5C1  0150        JMP BILD ;New screen
C0E1         0151 ;*********************************************************
C0E1         0152 ;*   LINK A LINE            CNTL L
C0E1 20C6C2  0153 LINK   JSR LINKLI ;Pack the next line to the current
C0E4 4CD5C1  0154        JMP BILD ;New screen
C0E7         0155 ;*********************************************************
C0E7         0156 ;*   TOGGLE INSERT-FLAG     CNTL X
C0E7 A980    0157 CHARIN LDA #$80  ;Toggle bit 7 of IFLG
C0E9 45DE    0158        EOR IFLG
C0EB 85DE    0159        STA IFLG
C0ED 60      0160        RTS
C0EE         0161 ;*********************************************************
C0EE         0162 ;*   TOGGLE CAPS LOCK       CNTL A
C0EE AD08A4  0163 GROSKL LDA $A408 ;Toggle bit 6 of $A408
C0F1 4940    0164        EOR #$40
C0F3 8D08A4  0165        STA $A408
C0F6 60      0166        RTS
C0F7         0167 ;*********************************************************
C0F7         0168 ;*   MOVE CURSOR RIGHT      CNTL J
C0F7         0169
C0F7 4C63C3  0170 CURRIG JMP CURR  ;Move cursor right once
C0FA         0171 ;*********************************************************
C0FA         0172 ;*   MOVE CURSOR LEFT       CNTL H
C0FA 4C5CC3  0173 CURLEF JMP CURL  ;Move cursor left once
C0FD         0174 ;*********************************************************
C0FD         0175 ;*   CURSOR TO LAST LOCAT.  CNTL R
C0FD A03B    0176 CURLAS LDY #WIDTH-1  ;Move cursor to last location
C0FF 4C6AC3  0177        JMP CURR1
C102         0178 ;*********************************************************
C102         0179 ;*   ZERO TABULATORPOSITION CNTL I
C102 A008    0180 TABZER LDY #TABMAX  ;Max. Tabulatoranzahl
C104 84D1    0181        STY TABNR
C106 88      0182        DEY
C107 B911C1  0183 TABZE1 LDA TABPOS,Y  ;Setze alle Tabulatoren
C10A 99D200  0184        STA TABBUF,Y
C10D 88      0185        DEY
```

```
C04E D2C0    0083
C050 83C1    0083
C052 C3C1    0083
C054 FDC0    0084        .WORD CURLAS, SPLIT, TOP, CURUP, RETURN, RETURN
C056 DBC0    0084
C058 98C1    0084
C05A ADC0    0084
C05C 9CC0    0084
C05E 9CC0    0084
C060 E7C0    0085        .WORD CHARIN, RETURN, TABZER, RETURN, RETURN, RETURN
C062 9CC0    0085
C064 02C1    0085
C066 9CC0    0085
C068 9CC0    0085
C06A 9CC0    0085
C06C 9CC0    0086        .WORD RETURN, RETURN
C06E 9CC0    0086
C070         0087 ;##################################################
C070         0088 ;*   DELETE                $7F
C070 C97F    0089 DELETE CMP #$7F  ;Delete?
C072 D00B    0090        BNE WRITE ;No
C074 205CC3  0091        JSR CURL ;Yes. Delete last character
C077 A920    0092        LDA #' '
C079 2094C0  0093        JSR WRIT2
C07C 4C5CC3  0094        JMP CURL
C07F         0095 ;##################################################
C07F         0096 ;*   DISPLAYABLE CHARACTERS
C07F 24DE    0097 WRITE  BIT IFL6 ;I-flag on?
C081 1011    0098        BPL WRIT2 ;No
C083 48      0099        PHA ;Yes, do a right-shift
C084 A03A    0100        LDY #WIDTH-2 ;Begin at the end of the line
C086 B9009A  0101 WRIT1  LDA VID8,Y
C089 C8      0102        INY
C08A 99009A  0103        STA VID8,Y
C08D 88      0104        DEY
C08E 88      0105        DEY
C08F C4DD    0106        CPY CURREL ;Reached cursor-position?
C091 10F3    0107        BPL WRIT1 ;No
C093 68      0108        PLA ;Yes
C094 A4DD    0109 WRIT2  LDY CURREL ;Write character to cursorposition
C096 99009A  0110        STA VID8,Y
C099 4C63C3  0111        JMP CURR ;Move cursor right
C09C         0112 ;##################################################
C09C         0113 ;*   RETURN
C09C 60      0114 RETURN RTS  ;Dummy-routine
C09D         0115 ;##################################################
C09D         0116 ;*   CARRIAGE-RETURN BIW.  CNTL M
C09D A000    0117 CARRI  LDY #0 ;Move cursor to CURB
C09F 206AC3  0118        JSR CURR1
C0A2 2070C2  0119 CURDWN JSR RESTOR  ;Restore TV-line
C0A5 A000    0120        LDY #0
C0A7 2013F7  0121        JSR UP1 ;1 line up in memory
C0AA 4CD5C1  0122 CARR2  JMP BILD ;New screen
C0AD         0123 ;##################################################
C0AD         0124 ;*   TEXT-CURSOR UP        CNTL U
C0AD 2070C2  0125 CURUP  JSR RESTOR ;Restore TV-line
C0B0 20DBF8  0126 CURUP1 JSR ATTOP  ;At top?
C0B3 B003    0127        BCS CURUP2
```

```
C10E 10F7   0186        BPL TABZE1
C110 60     0187        RTS
C111 00     0188 TABPOS .BYT 0,7,11,22,29,39,49,59
C112 07     0188
C113 0B     0188
C114 16     0188
C115 1D     0188
C116 27     0188
C117 31     0188
C118 3B     0188
C119        0189 ;**************************************************
C119        0190 ;*   GENERATE TABULATOR     CNTL G
C119 A000   0191 TABGEN LDY #0
C11B A5D1   0192        LDA TABNR ;Nr of current tabulatorpositions
C11D F021   0193        BEQ TABGE4 ;Zero?
C11F C908   0194        CMP #TABMAX ;All full
C121 B024   0195        BCS TABGE5 ;Yes
C123 B9D200 0196 TABGE1 LDA TABBUF,Y  ;No, tabulator already set?
C126 C5DD   0197        CMP CURREL
C128 F01D   0198        BEQ TABGE5 ;Yes
C12A C8     0199        INY
C12B C4D1   0200        CPY TABNR
C12D D0F4   0201        BNE TABGE1
C12F A4D1   0202        LDY TABNR ;No, Set tabulator
C131 88     0203        DEY
C132 B9D200 0204 TABGE2 LDA TABBUF,Y  ;Load tabulatorposition
C135 C5DD   0205        CMP CURREL ;Greater than current position?
C137 9006   0206        BCC TABGE3 ;No, store tabulator
C139 99D300 0207        STA TABBUF+1,Y ;Yes, move position forward
C13C 88     0208        DEY
C13D 10F3   0209        BPL TABGE2
C13F C8     0210 TABGE3 INY  ;Store tabulator
C140 A5DD   0211 TABGE4 LDA CURREL
C142 99D200 0212        STA TABBUF,Y
C145 E6D1   0213        INC TABNR ;Increment # of tabulators
C147 60     0214 TABGE5 RTS
C148        0215 ;**************************************************
C148        0216 ;*   CLEAR TABULATOR        CNTL C
C148 A5D1   0217 TABCLR LDA TABNR  ;# of tabulators = 0?
C14A F01D   0218        BEQ TABCL3 ;Yes
C14C A000   0219        LDY #0 ;No
C14E B9D200 0220 TABCL1 LDA TABBUF,Y  ;Tabulatorposition = current cursor?
C151 C5DD   0221        CMP CURREL
C153 F007   0222        BEQ TABCL2 ;Yes
C155 C8     0223        INY
C156 C4D1   0224        CPY TABNR ;All done?
C158 D0F4   0225        BNE TABCL1 ;No
C15A F00D   0226        BEQ TABCL3 ;There was no tabulator set
C15C B9D300 0227 TABCL2 LDA TABBUF+1,Y  ;Move next tabpos. to current tabpos.
C15F 99D200 0228        STA TABBUF,Y
C162 C8     0229        INY
C163 C4D1   0230        CPY TABNR ;All done?
C165 D0F5   0231        BNE TABCL2 ;No
C167 C6D1   0232        DEC TABNR ;Decrement # of tabulators
C169 60     0233 TABCL3 RTS
C16A        0234 ;**************************************************
C16A        0235 ;*   TABULATOR FORWARD      CNTL F
C16A A000   0236 TABFOR LDY #0
```

```
C16C B9D200  0237 TABF01 LDA TABBUF,Y  ;Tabulatorpos. less equal cursor?
C16F C5DD    0238        CMP CURREL
C171 F00A    0239        BEQ TABF02 ;Yes
C173 9008    0240        BCC TABF02 ;Yes
C175 C93C    0241        CMP #WIDTH ;No, move cursor when in TV-line
C177 B009    0242        BCS TABF03
C179 A8      0243        TAY
C17A 4C6AC3  0244        JMP CURR1 ;Move cursor
C17D C8      0245 TABF02 INY
C17E C4D1    0246        CPY TABNR ;All positions done?
C180 D0EA    0247        BNE TABF01 ;No
C182 60      0248 TABF03 RTS  ;Yes
C183         0249 ;*****************************************************
C183         0250 ;*   PREVIOUS TABULATOR     CNTL P
C183 A4D1    0251 TABPRE LDY TABNR
C185 B9D200  0252 TABPR1 LDA TABBUF,Y  ;Tabulatorpos. greater cursorpos?
C188 C5DD    0253        CMP CURREL
C18A B008    0254        BCS TABPR2 ;Yes, next tabpos.
C18C C900    0255        CMP #0 ;No, tabpos greater 0?
C18E 3007    0256        BMI TABPR3 ;No
C190 A8      0257        TAY
C191 4C6AC3  0258        JMP CURR1 ;Yes, move cursor
C194 88      0259 TABPR2 DEY  ;All positions done?
C195 10EE    0260        BPL TABPR1 ;No
C197 60      0261 TABPR3 RTS  ;Yes
C198         0262 ;*****************************************************
C198         0263 ;*   TEXT-CURSOR TO TOP     CNTL T
C198 2070C2  0264 TOP    JSR RESTOR ;Restore TV-line
C19B 20BCF8  0265        JSR TOPNO ;Top-line becomes current line
C19E 4CD5C1  0266        JMP BILD ;New screen
C1A1         0267 ;*****************************************************
C1A1         0268 ;*    TEXT-CURSOR TO BOTTOM CNTL B
C1A1 2070C2  0269 BOTTOM JSR RESTOR  ;Restore TV-line
C1A4 20C5F8  0270        JSR SETBOT ;Bottom-line becomes current line
C1A7 20B0C0  0271        JSR CURUP1 ;1 line down in memory
C1AA 4CD5C1  0272        JMP BILD ;New screen
C1AD         0273 ;*****************************************************
C1AD         0274 ;*    ERASE A CHARACTER     CNTL E
C1AD A4DD    0275 ERASE  LDY CURREL ;Load cursor-position
C1AF C8      0276 ERASE1 INY  ;Do a left-shift, begin with cursor
C1B0 B9009A  0277        LDA VID8,Y
C1B3 88      0278        DEY
C1B4 99009A  0279        STA VID8,Y
C1B7 C8      0280        INY
C1B8 C03C    0281        CPY #WIDTH ;Reached the right position?
C1BA D0F3    0282        BNE ERASE1 ;No
C1BC A920    0283        LDA #' ' ;Yes, blank in last location
C1BE 88      0284        DEY
C1BF 99009A  0285        STA VID8,Y
C1C2 60      0286        RTS
C1C3         0287 ;*****************************************************
C1C3         0288 ;*   QUIT SCREENEDITOR      CNTL Q
C1C3 2070C2  0289 QUIT   JSR RESTOR ;Restore TV-line
C1C6 203DC2  0290        JSR CUREND ;Cursor to CUR15
C1C9 2013EA  0291        JSR CRLOW ;Empty line
C1CC 2013EA  0292        JSR CRLOW ;Empty line
C1CF 2027F7  0293        JSR PLNE ;Print current line
C1D2 68      0294        PLA ;Return to EDITOR-commands
```

```
C1D3 68      0295        PLA ;Delete PC from last JSR from stack
C1D4 60      0296        RTS
C1D5         0297 ;*****************************************************
C1D5         0298 ;*  SUBROUTINES WHICH HANDELS WITH VIDEO-INTERFACE
C1D5         0299 ;*
C1D5         0300 ;*****************************************************
C1D5         0301 ;*  WRITES A SCREEN: 8 LINES BEFORE AND 7 AFTER ACTUAL LINE
C1D5 2034F9  0302 BILD   JSR SAVNOW ;Save current line
C1D8 A907    0303        LDA #7 ;8 lines before actual TV-line
C1DA 85DA    0304        STA TVLIN
C1DC 2034C2  0305        JSR CURINI ;Initialize CURADR to VID8
C1DF 2052C2  0306        JSR LINMOV ;Move 1 line to TV
C1E2 38      0307 BIL0   SEC ;Decrement CURADR with LINLEN
C1E3 A5DB    0308        LDA CURADR
C1E5 E940    0309        SBC #LINLEN
C1E7 85DB    0310        STA CURADR
C1E9 B002    0311        BCS *+4
C1EB C6DC    0312        DEC CURADR+1
C1ED 20DBFB  0313        JSR ATTOP ;At top?
C1F0 B009    0314        BCS BIL1 ;Yes, fill a blank line
C1F2 20E3F6  0315        JSR DOWN1 ;No, down in memory
C1F5 2052C2  0316        JSR LINMOV ;Move next line to TV
C1F8 4CFEC1  0317        JMP BIL2
C1FB 2064C2  0318 BIL1   JSR LINBLK ;Move blanks to TV
C1FE C6DA    0319 BIL2   DEC TVLIN ;Next TV-line
C200 10E0    0320        BPL BIL0 ;All done?
C202         0321
C202 A906    0322        LDA #6 ;7 lines after actual TV-line
C204 85DA    0323        STA TVLIN
C206 2034C2  0324        JSR CURINI ;Initialize CURADR to VID8
C209 20D0F8  0325        JSR RESNOW ;Restore current line
C20C 18      0326 BIL3   CLC ;Increment CURADR with LINLEN
C20D A940    0327        LDA #LINLEN
C20F 65DB    0328        ADC CURADR
C211 85DB    0329        STA CURADR
C213 9002    0330        BCC *+4
C215 E6DC    0331        INC CURADR+1
C217 20E9FB  0332        JSR ATBOT ;At bottom?
C21A B00B    0333        BCS BIL4 ;Yes, fill a blank line
C21C A000    0334        LDY #0 ;No, up in memory
C21E 2013F7  0335        JSR UP1
C221 2052C2  0336        JSR LINMOV ;Move next line to TV
C224 4C2AC2  0337        JMP BIL5
C227 2064C2  0338 BIL4   JSR LINBLK ;Move blanks to TV
C22A C6DA    0339 BIL5   DEC TVLIN ;Next TV-line
C22C 10DE    0340        BPL BIL3 ;All done?
C22E         0341
C22E 2034C2  0342        JSR CURINI ;Initialize CURADR to VID8
C231 4CD0FB  0343        JMP RESNOW ;Original current line
C234         0344 ;*****************************************************
C234         0345 ;*  INITIALIZE ABSOLUTE CURSORADRESS TV-LINE 8
C234 A99A    0346 CURINI LDA #>VID8  ;Initialize CURADR to VID8
C236 85DC    0347        STA CURADR+1
C238 A900    0348        LDA #<VID8
C23A 85DB    0349        STA CURADR
C23C 60      0350        RTS
C23D         0351 ;*****************************************************
C23D         0352 ;*  INITIALIZE RELATIVE CURSORADRESS AT LAST TV-LINE
```

```
C23D A90E    0353 CUREND LDA #14  ;Set cursor to CUR15
C23F 8D00A3  0354        STA VIDADR
C242 A903    0355        LDA #>CUR15
C244 8D01A3  0356        STA VIDREG
C247 A90F    0357        LDA #15
C249 8D00A3  0358        STA VIDADR
C24C A9C0    0359        LDA #<CUR15
C24E 8D01A3  0360        STA VIDREG
C251 60      0361        RTS
C252         0362 ;*****************************************************
C252         0363 ;*   MOVE 1 TEXTLINE TO ACTUAL TV-LINE
C252 A000    0364 LINMOV LDY #0
C254 B1DF    0365 LINMO1 LDA (NOWLN),Y  ;From memory
C256 F00E    0366        BEQ LINBL1 ;End of text $00?
C258 C90D    0367        CMP #CR ;Carriage return?
C25A F00A    0368        BEQ LINBL1
C25C 91DB    0369        STA (CURADR),Y ;To actual TV-line
C25E C8      0370        INY
C25F C03C    0371        CPY #WIDTH ;Line full?
C261 D0F1    0372        BNE LINMO1
C263 60      0373        RTS
C264         0374 ;*****************************************************
C264         0375 ;*   FILL TV-LINE WITH BLANKS
C264 A000    0376 LINBLK LDY #0
C266 A920    0377 LINBL1 LDA #' '  ;Blank
C268 91DB    0378 LINBL2 STA (CURADR),Y  ;To TV-line
C26A C8      0379        INY
C26B C040    0380        CPY #LINLEN ;TV-line full?
C26D D0F9    0381        BNE LINBL2
C26F 60      0382        RTS
C270         0383 ;*****************************************************
C270         0384 ;*   CALL REPLAC OF ORIGINAL EDITOR
C270 20E9F8  0385 RESTOR JSR ATBOT  ;At bottom?
C273 B00F    0386        BCS INSERT ;Yes, than insert a line
C275 202EC3  0387        JSR CHARME ;# of characters in memory
C278 203EC3  0388        JSR CHARTV ;# of characters in TV
C27B 204EC3  0389        JSR MOVECH ;TV-line to DIBUFF
C27E 20B2F8  0390        JSR CFLG ;C-command-flag
C281 4C3FF9  0391        JMP REPLAC ;Replace from AIM-Editor
C284         0392 ;*****************************************************
C284         0393 ;*   INSERT A LINE IN AIM TEXT-BUFFER
C284 203EC3  0394 INSERT JSR CHARTV  ;# of characters in TV
C287 A000    0395 INSER1 LDY #0  ;OLDLEN = 0
C289 84E9    0396        STY OLDLEN
C28B 204EC3  0397        JSR MOVECH ;TV-line to DIBUFF
C28E 20B6F8  0398        JSR KIFLG ;I-command-flag
C291 4C3FF9  0399        JMP REPLAC ;Replace from AIM-Editor
C294         0400 ;*****************************************************
C294         0401 ;*   SPLITS A LINE
C294 A4DD    0402 SPLIN  LDY CURREL ;Cursorposition = 0?
C296 C000    0403        CPY #0
C298 D003    0404        BNE SPLIN1
C29A 4CB8C0  0405        JMP LININS ;Yes, normal INSERT
C29D A200    0406 SPLIN1 LDX #0  ;No
C29F B9009A  0407 SPLIN2 LDA VID8,Y  ;Characters right from cursor
C2A2 9D38A4  0408        STA DIBUFF,X ;To DIBUFF
C2A5 C8      0409        INY
C2A6 E8      0410        INX
```

```
C2A7 C03C    0411        CPY #WIDTH ;End of line?
C2A9 D0F4    0412        BNE SPLIN2
C2AB A920    0413        LDA #' ' ;Reduce blanks
C2AD CA      0414 SPLIN3 DEX
C2AE DD38A4  0415        CMP DIBUFF,X
C2B1 F0FA    0416        BEQ SPLIN3
C2B3 E8      0417        INX
C2B4 86EA    0418        STX LENGTH ;# of non-blank characters cursorright
C2B6 202EC3  0419        JSR CHARME ;# of characters in memory
C2B9 20B2F8  0420        JSR CFLG ;C-command-flag
C2BC 203FF9  0421        JSR REPLAC ;Replace from AIM-Editor
C2BF A4DD    0422        LDY CURREL ;# of TV-characters till cursor
C2C1 84EA    0423        STY LENGTH
C2C3 4C87C2  0424        JMP INSER1 ;Insert the TV-line
C2C6         0425 ;**************************************************
C2C6         0426 ;*   LINK A LINE
C2C6 A4DD    0427 LINKLI LDY CURREL  ;Cursorposition = 0?
C2C8 D003    0428        BNE LINKL0
C2CA 4C1BC3  0429        JMP KILL ;Yes, kill the line
C2CD A200    0430 LINKL0 LDX #0  ;No
C2CF BD409A  0431 LINKL1 LDA VID9,X  ;From TV-line 9
C2D2 99009A  0432        STA VID8,Y ;To TV-line 8 right the cursor
C2D5 C8      0433        INY
C2D6 E8      0434        INX
C2D7 C03C    0435        CPY #WIDTH ;Line 8 full?
C2D9 D0F4    0436        BNE LINKL1
C2DB 2070C2  0437        JSR RESTOR ;Restore the line 8
C2DE A93C    0438        LDA #WIDTH ;Rest # of characters in TV-line 9
C2E0 38      0439        SEC
C2E1 E5DD    0440        SBC CURREL
C2E3 A8      0441        TAY
C2E4 A200    0442        LDX #0
C2E6 B9409A  0443 LINKL2 LDA VID9,Y  ;Rest characters from TV-line 9
C2E9 9D38A4  0444        STA DIBUFF,X ;To DIBUFF
C2EC C8      0445        INY
C2ED E8      0446        INX
C2EE C03C    0447        CPY #WIDTH ;All done?
C2F0 D0F4    0448        BNE LINKL2
C2F2 A920    0449        LDA #' ' ;Reduce blanks in DIBUFF
C2F4 CA      0450 LINKL3 DEX
C2F5 3019    0451        BMI LINKL4 ;Kill line when full of blanks
C2F7 DD38A4  0452        CMP DIBUFF,X
C2FA F0F8    0453        BEQ LINKL3
C2FC E8      0454        INX ;# of non-blank characters
C2FD 86EA    0455        STX LENGTH
C2FF A000    0456        LDY #0
C301 2013F7  0457        JSR UP1 ;1 line up in memory
C304 202EC3  0458        JSR CHARME ;# of characters in memory
C307 20B2F8  0459        JSR CFLG ;C-command-flag
C30A 203FF9  0460        JSR REPLAC ;Replace of AIM-Editor
C30D 4CE3F6  0461        JMP DOW1 ;1 line down in memory
C310 A000    0462 LINKL4 LDY #0  ;1 line up in memory
C312 2013F7  0463        JSR UP1
C315 2018C3  0464        JSR KILL ;Kill the line
C318 4CE3F6  0465        JMP DOW1 ;1 line down in memory
C31B         0466 ;**************************************************
C31B         0467 ;*   KILL A LINE FROM AIM TEXT-BUFFER
C31B 20E9F8  0468 KILL   JSR ATBOT ;At bottom?
```

```
C31E B00D    0469        BCS KILL1
C320 A000    0470        LDY #0 ;No
C322 84EA    0471        STY LENGTH ;# of characters = 0
C324 202EC3  0472        JSR CHARME ;# of characters in memory
C327 20B6F8  0473        JSR KIFLG ;K-command-flag
C32A 203FF9  0474        JSR REPLAC ;Replace from AIM-Editor
C32D 60      0475 KILL1  RTS
C32E         0476 ;*****************************************************
C32E         0477 ;*   CALCULATE # OF CHARACTERS IN MEMORY-LINE
C32E A000    0478 CHARME LDY #0
C330 B1DF    0479 CHARM1 LDA (NOWLN),Y  ;Load character
C332 F007    0480        BEQ CHARM2 ;End of text $0 ?
C334 C90D    0481        CMP #CR ;Carriage return?
C336 F003    0482        BEQ CHARM2
C338 C8      0483        INY
C339 D0F5    0484        BNE CHARM1
C33B 84E9    0485 CHARM2 STY OLDLEN  ;# of characters
C33D 60      0486        RTS
C33E         0487 ;*****************************************************
C33E         0488 ;*   CALCULATE # OF CHARACTERS IN TV-LINE (MIN. 1)
C33E A03B    0489 CHARTV LDY #WIDTH-1  ;Begin on right side
C340 B9009A  0490 CHART1 LDA VIDB,Y  ;Load character
C343 C920    0491        CMP #' ' ;Reduce blanks
C345 D003    0492        BNE CHART2
C347 88      0493        DEY
C348 D0F6    0494        BNE CHART1
C34A C8      0495 CHART2 INY  ;# of non-blank-characters in TV-line
C34B 84EA    0496        STY LENGTH
C34D 60      0497        RTS
C34E         0498 ;*****************************************************
C34E         0499 ;*   MOVE TV-LINE TO DIBUFF
C34E A000    0500 MOVECH LDY #0
C350 B9009A  0501 MOVEC1 LDA VIDB,Y  ;From TV-line
C353 9938A4  0502        STA DIBUFF,Y ;To DIBUFF
C356 C8      0503        INY
C357 C03C    0504        CPY #WIDTH ;All done?
C359 D0F5    0505        BNE MOVEC1
C35B 60      0506        RTS
C35C         0507 ;*****************************************************
C35C         0508 ;*   MOVE CURSOR LEFT ONCE
C35C A4DD    0509 CURL   LDY CURREL ;Cursor-position = 0 ?
C35E F024    0510        BEQ CURR2 ;Yes, RTS
C360 88      0511        DEY ;No, decrement Y
C361 1007    0512        BPL CURR1 ;Branch always
C363         0513 ;*****************************************************
C363         0514 ;*   MOVE CURSOR RIGHT ONCE
C363 A4DD    0515 CURR   LDY CURREL
C365 C03B    0516        CPY #WIDTH-1 ;Cursorposition = left side ?
C367 F01B    0517        BEQ CURR2 ;Yes, RTS
C369 C8      0518        INY ;No, increment
C36A 84DD    0519 CURR1  STY CURREL ;New cursorposition
C36C 98      0520        TYA ;Also to CRT-controller
C36D 18      0521        CLC
C36E 6900    0522        ADC #<CUR8
C370 A20F    0523        LDX #15
C372 8E00A3  0524        STX VIDADR
C375 8D01A3  0525        STA VIDREG
C378 A902    0526        LDA #>CUR8
C37A 6900    0527        ADC #0
C37C A20E    0528        LDX #14
C37E 8E00A3  0529        STX VIDADR
C381 8D01A3  0530        STA VIDREG
C384 60      0531 CURR2  RTS
C385         0532
C385         0533        .END

  ERRORS= 0000
```

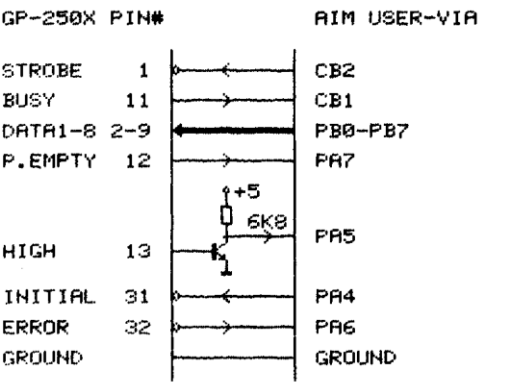
Wolfgang Radeloff, 2080 Pinneberg

Editor mit Steuerzeichen

AIM 65 steuert Drucker Seikosha GP 250

Diese Ausgaberoutine enststand mit dem Ziel, aus dem Text des Editors heraus alle Funktionen des Druckers mit Control-Codes zu steuern und diese auf dem Bildschirm für eine Editierung sichtbar zu machen.

Der Drucker wird an die User-VIA des AIM 65 gem. nebenstehender Abbildung angeschlossen (auch diese Grafik ist per Programm auf dem Drucker erzeugt worden). Es werden die Fehlersignale PAPER EMPTY, HIGH und ERROR ausgewertet. Der INITIAL-Anschluß wird zur Steuerung durch das Programm ebenfalls an Port A geschaltet. Eine Kabellänge von 2 m hat sich als unkritisch erwiesen.



Die Treiberroutinen für den Drukker sind so ausgelegt, daß sie von beliebigen Ausgabeprogrammen als Unterprogramme aufgerufen werden können. INITYP initialisiert VIA und Drucker. WRITE sendet ein Zeichen im Akku an den Drukker und wertet die Fehler-Meldeleitungen aus. Die STOP-Taste des GP 250 gibt 'Paper EMPTY/ STOP' an das Display und führt zum Programmabbruch und Eintritt in den Monitor bei COMIN. Weitere Meldungen sind: PRINTER DOWN und PRINTER ERROR.

Der Protokollbetrieb über den DILINK-Vektor wird mit F2 initialisiert. INITDI übernimmt die Initialisierung, DIPRIN gibt ein Zeichen im Akku an den Drucker und weiter an das Videodisplay aus. Die Taste <PRINT> schaltet wie gewohnt den Protokollbetrieb ein und aus.

Alle an den Drucker oder das Programm zu sendenden Control-Codes sind mit dem Prefix '↑' ($5E) als Klein- oder Großbuchstaben im Bereich A bis] (a bis rechte Akkolade) in den Text aufzunehmen. Sie werden nicht ausgedruckt, können aber im Protokollbetrieb dokumentiert werden. Dazu ein Beispiel:

```
^CA Texteditor AIM-65 steuert Drucker SEIKOSHA GP 250 ^M^U
```

Die Control-Codes DC1, DC2 und DC3 steuern Programmfunktionen und gelangen nicht zum Drucker. CTRL-Q initialisiert Drucker und VIA neu. CTRL-R schaltet in den Hexbetrieb. Nach ↑R folgende ASCII-Zeichen 0-9 und A-F werden gepackt als 1 Byte an den Drucker übermittelt. SPACE wird ignoriert. Ein weiteres CTRL-R schaltet in den Normalzustand zurück (Toggle Funktion). CTRL-S steuert den automatischen Seitenvorschub. Nach dem Initialisieren ist er 'EIN' und kann mit CTRL-S aus- und wieder eingeschaltet werden. CTRL-L (FF) im Text löst unabhängig von der Autopaging-Funktion Formfeed aus. Bei abgeschaltetem Autopaging kann mit Druckerkommandos ein beliebiges Paging definiert werden. Beispiele zum Hexbetrieb:

```
^CW^R 91 1C 1C FF 00 00 00 ^R
^CW^R 92 08 08 FF 00 00 00 ^R
 GP-250X PIN#            AIM USER-VIA
^CL^R02^R
              ^R81202020202020202081^R
 STROBE     1 ^R80828282858282828292^R CB2
```

Ausschnitt aus obiger Grafik

Ohm-Zeichen
```
^CW^R 80 4E 71 01 71 4E 00 ^R
75^R80^R GESAMTWIDERSTAND.

75Ω GESAMTWIDERSTAND.
```

Mapping der Control-Codes: Die Initialisierungstabelle INIT legt im RAM eine Tabelle CTLTAB an. Alle mit Prefix definierten CTRL-Codes werden über sie gemappt. Jeder Code kann durch Eintrag einer '00' in seiner Funktion abgeschaltet werden und jedem Prefix-Code kann durch Eintrag ein anderer Wert zugewiesen werden. Die Tabelle wird mit F2 angelegt und durch OUT=U nicht geändert.

Ist das Zeichen] von der Tastatur nicht zu erreichen (mit Prefix=ESC), so wird in die 12. Stelle der Tabelle $1B eingetragen und CTRL-B im Text sendet den Code ESC an den Drucker. Als Beispiel: Nach Änderung der CTLTAB wird CR vom Texteditor zum Drucker unterdrückt und CTRL-D im Text als LF an den Drucker ausgegeben.

Text:
```
1) CNTL D WIRD ALS LF DEFINIERT: IN DER REL.ADDR.
 DER CTLTAB 04 WIRD 0A NOTIERT^D
2) CR WIRD AUSGEBLENDET: IN DER REL.ADDR. DER CTLTAB
 0D WIRD 00 EINGETRAGEN.^D
3) ^D UEBERNIMMT IM TEXT DIE MARKIERUNG ZEILENENDE
/NAECHSTE ZEILE. ^D
```

CTLTAB:
```
<M>=A35A 00 00 02 03
< > A35E 0A 00 00 07
< > A362 00 00 0A 00
< > A366 0C 00 0E 0F
```

Der an den Drucker in langer Zeile ausgegebene Text wird nachstehend aus drucktechnischen Gründen in jeweils 2 Zeilen geteilt-

```
1) CNTL D WIRD ALS LF DEFINIERT: IN DER REL.ADDR.
2) CR WIRD AUSGEBLENDET: IN DER REL.ADDR.
3) ^D UEBERNIMMT IM TEXT DIE MARKIERUNG
                                    DER CTLTAB 04 WIRD 0A NOTIERT.
                                DER CTLTAB 0D WIRD 00 EINGETRAGEN.
                              ZEILENENDE/NAECHSTE ZEILE.
```

Sollte das Zeichen '/' von der Tastatur nicht erreichbar sein, so kann stattdessen '@' als Prefix benutzt werden. In der Tabelle CODES ist dann $5E gegen $40 zu tauschen.

Weitere Programmfunktionen können unaktiven Codes zugewiesen werden: Der Code wird in der CTLTAB freigegeben, in der Tabelle CODES notiert und die Adresse der Routine in der Tabelle AKTION hinterlegt. Nach Erhöhen der Parser-Loopvariablen ist die Erweiterung eingebunden.

Die Init.-Routine fragt mit VTAB= nach der Anzahl der Leerstellen am linken Rand und erwartet die zweistellige Eingabe einer Dezimalzahl. <SPACE> stellt den Defaultwert 06 ein, CR beläßt den vorherigen Wert.

```
0000                                ;SYSTEM ROUTINEN          IM A
0000 PACK       =$EA84              ;PACK 2 ASCII IM A IN 1 BYTE
0000 OUTPUT     =$E97A              ;AUSGABE AN DISPLAY/DRUCKER
0000 COMIN      =$E1A1              ;RESTOR STACK & WARMSTART
0000 CRLOW      =$EA13              ;CR TO D/P
0000 VIDEO      =$8C0F              ;CHAR. IM ACCU ZUM BILDSCH.
0000 RD2        =$EA5D              ;HOLT 2 BYTES VOM KB
0000
0000 DILINK     =$A406              ;SYSTEM VARIABLEN
0000 PRIFLG     =$A411
0000 SAVEX      =$A422
0000 UOUT       =$010A
```

```
0000 JUMP           =$A47D
0000 F1             =$010C
0000
0000 DRB            =$A000          ;USERVIA
0000 DRA            =$A001
0000 DDRB           =$A002
0000 DDRA           =$A003
0000 PCR            =$A00C
0000 IFR            =$A00D
0000                                ;CONTROL CODES
0000 DC1            =$11            ;DRUCKER NEU INIT.        DRUCKER
0000 DC2            =$12            ;JE 2 BYTES ALS 1 HEXBYTE AN
0000 DC3            =$13            ;AUTOPAGING EIN-AUS
0000 DC4            =$14            ;NUR CR
0000 CR             =$0D            ;CR & JE NACH DRUCKEREINSTELLUNG
0000 LF             =$0A            ;CR & LF                    LF
0000 SPC            =$20            ;SPACE
0000 FF             =$0C            ;FORM FEED
0000 ESC            =$1B            ;ESCAPE, PREFIX FUER DRUCKER-CMDS
0000                                                            CMDS
0000                *=$A35A         ;VARIABLEN
A35A VARRAM         *=*             ;START VARIABLEN
A35A CTLTAB         *=*+32          ;CTL-CODE SCHALTERTABELLE
A37A VTAB           *=*+1           ;TABULATOR LINKER RAND       6
A37B HX             *=*+1           ;ZAEHLER 2 ASCII-BYTES       1
A37C FLAG           *=*+1           ;F7=1 HEXMODE                0
A37D                                ;F6=1 CTL-CODE FOLLOWS       0
A37D                                ;F5=1 AUTO PAGING           $201
A37D LNPAG          *=*+1           ;ZEILEN/SEITE                66
A37E DILIN2         *=*+2           ;2.DILINK AUF VIDEO        $8C0F
A380
A380                *=UOUT          ;SYSTEM VECTOREN
010A          5B02  .WOR UTEXT      ;UOUT - USER OUTPUT VECTOR
010C                *=F1
010C          4C6402 JMP INIT       ;F1 - INIT UTEXT VON TASTATUR
010F          4CBD03 JMP INITDI     ;F2 - INIT OUTPUT VIA DILINK-
0112                                                      VECTOR
0112                *=$0200         ;PROGRAMMSTART
0200                                ; DRUCKER-INITIALISIERUNG
0200                                ; UND UEBERGABE EINES ZEICHENS
0200 INITVP   A2FF    LDX #$FF      ;INITIALISIERUNG VIA    IM ACCU
0202          8E02A0  STX DDRB
0205          E8      INX
0206          8E00A0  STX DRB
0209          A218    LDX #$18
020B          8E03A0  STX DDRA
020E          8E01A0  STX DRA
0211          A2A0    LDX #$A0
0213          8E0CA0  STX PCR
0216          203102  JSR TEST      ;PRINTER 'ON' ?

0219 INIPR    A208    LDX #$08      ;SENDE INITIAL-PULS AN DRUCKER
021B          8E01A0  STX DRA
021E          A218    LDX #$18
0220          8E01A0  STX DRA
0223          202A02  JSR BUSY
0226          60      RTS
0227
```

```
0227 WRITE    8D00A0 STA DRB
022A BUSY     A910   LDA #%00010000     ;TEST, OB DRUCKER IN BUSY-ZUST'
022C          2C0DA0 BIT IFR
022F          F0F9   BEQ BUSY
0231 TEST     A920   LDA #%00100000     ;TEST DER FEHLERLEITUNGEN
0233          2C01A0 BIT DRA
0236          D005   BNE HIGH
0238          5007   BVC ERROR
023A          3009   BMI PEMPTY
023C          60     RTS
023D
023D HIGH     A200   LDX #0
023F          1006   BPL ERREX
0241 ERROR    A20D   LDX #M2-M1
0243          1002   BPL ERREX
0245 PEMPTY   A21B   LDX #M3-M1
0247 ERREX    204D02 JSR KEPU
024A          4CA1E1 JMP COMIN
024D
024D KEPU     BD6303 LDA M1,X
0250          48     PHA
0251          297F   AND #$7F
0253          207AE9 JSR OUTPUT
0256          E8     INX
0257          68     PLA
0258          10F3   BPL KEPU
025A          60     RTS
025B                                   ;INITIAL. TEXTEDITOR AN DRUCKER
025B                                   ; ********* [F2] **************
025B UTEXT    B037   BCS UPRINT
025D          A9A0   LDA #$A0          ;TEST, OB SCHON INITIALISIERT.
025F          CD0CA0 CMP PCR
0262          F013   BEQ UGCNT
0264 INIT     200002 JSR INITVP        ;INIT VIA & DRUCKER
0267          A200   LDX #0            ;SEND LINES/PAGE AN DRUCKER
0269          203903 JSR THERMS
026C          A223   LDX #35           ;INIT VARIABLEN
026E UTX1     BD9503 LDA TABL1,X
0271          9D5AA3 STA VARRAM,X
0274          CA     DEX
0275          10F7   BPL UTX1
0277 UGCNT    A22C   LDX #M4-M1
0279          204D02 JSR KEPU
027C          205DEA JSR RD2           ;GET VERTIKAL TABULATOR
027F          900A   BCC UGCNT1
0281          C90D   CMP #CR           ;WENN CR...
0283          F009   BEQ SETVC         ;DANN VTAB NICHT AENDERN
0285          C920   CMP #SPC          ;WENN SPACE...
0287          D0EE   BNE UGCNT
0289          A906   LDA #6            ;DANN VTAB = 6
028B UGCNT1   8D7AA3 STA VTAB
028E SETVC    2013EA JSR CRLOW         ;CR AN D/P
0291          4C0303 JMP TAB1          ;RAND EINRUECKEN
0294                                   ;AUSGABE TEXTEDITOR AN DRUCKER
0294 UPRINT   8E22A4 STX SAVEX         ;SAVE X-REG
0297          68     PLA               ;GET CHARACTER
0298          C90D   CMP #CR           ;MAP CR
029A          D004   BNE PARSER
```

```
029C           AA       TAX
029D           BD5AA3   LDA CTLTAB,X
02A0 PARSER    A209     LDX #9
02A2 P1        DD4503   CMP CODES,X
02A5           F011     BEQ DO
02A7           CA       DEX
02A8           10F8     BPL P1
02AA OTHER     2C7CA3   BIT FLAG
02AD           707D     BVS CTL0
02AF           3067     BMI HEX0
02B1 EXIT1     202702   JSR WRITE
02B4 EXIT0     AE22A4   LDX SAVEX
02B7           60       RTS
02B8                                ;PARSER FINDET STEUERZEICHEN UND
02B8                                ;SETZT ZEIGER AUF DIE ENTSPR.
02B8 DO        48       PHA         ;SAVE ACCU              ROUTINE
02B9           8A       TXA         ;INDEX * 2
02BA           0A       ASL A
02BB           AA       TAX
02BC           BD4F03   LDA AKTION,X    ;VECTOR AUS AKTION-TABELLE
02BF           8D7DA4   STA JUMP        ;... IN RAM SCHREIBEN.
02C2           BD5003   LDA AKTION+1,X
02C5           8D7EA4   STA JUMP+1
02C8           68       PLA             ;RESTORE ACCU
02C9           6C7DA4   JMP (JUMP)      ;UND ROUTINE AUSFUEHREN!
02CC                                ;VOM PARSER ANGESTEUERTE ROUTI
02CC TOGHEX    A980     LDA #$80        ;HEXMODE EIN-AUS          NEN
02CE           D006     BNE SET
02D0 TOGCTL    A940     LDA #$40        ;FOLGENDER CODE IST CTL-CODE
02D2           D002     BNE SET
02D4 TOGPAG    A920     LDA #$20        ;PAGING EIN-AUS
02D6 SET       4D7CA3   EOR FLAG
02D9           8D7CA3   STA FLAG
02DC           4CB402   JMP EXIT0
02DF NEWINI    206402   JSR INIT        ;DRUCKER NEU INITIALISIEREN
02E2           4CB402   JMP EXIT0
02E5 PAG0      202702   JSR WRITE       ;LF ,CR AN DRUCKER
02E8           A920     LDA #%00100000  ;TEST AUTO-PAGING FLAG
02EA           2C7CA3   BIT FLAG
02ED           F014     BEQ TAB1        ;AUS! TABULATOR AUSFUEHREN
02EF           CE7DA3   DEC LNPAG       ;ZAEHLER ZEILEN/SEITE -1
02F2           100F     BPL TAB1        ;<>0, TABULATOR
02F4 FF0       A90C     LDA #FF         ;DO FORMFEED
02F6           202702   JSR WRITE
02F9           A942     LDA #66         ;RESTORE LINES/PAGE
02FB           8D7DA3   STA LNPAG
02FE           A90A     LDA #LF         ;DO CR & LF
0300 TAB0      202702   JSR WRITE       ;ODER DC4 AN DRUCKER
0303 TAB1      AE7AA3   LDX VTAB        ;TABULATOR LINKER RAND..
0306           F0AC     BEQ EXIT0       ;..IST =0
0308 TAB2      A920     LDA #SPC        ;..IST>0
030A           202702   JSR WRITE       ;SPACE AN DRUCKER
030D           F8       SED             ;VTAB IST BCD-ZAHL!
030E           8A       TXA             ;VTAB-1
030F           38       SEC
0310           E901     SBC #1
0312           D8       CLD
0313           AA       TAX
```

```
0314           D0F2    BNE TAB2            ;NICHT GENUG SPACE AN DRUCKER
0316           F09C    BEQ EXIT0           ;GENUG!                    1 HEXBYTE
0318                                       ;2 ASCII BYTES WERDEN ALS
0318 .                                     ;AN DEN DRUCKER AUSGEGEBEN.
0318 HEX0      C920    CMP #SPC            ;IGNORIERE SPACE
031A           F098    BEQ EXIT0
031C           2084EA  JSR PACK

031F           CE7BA3  DEC HX              ;COUNTER 2 BYTES
0322           1090    BPL EXIT0           ;1.BYTE NICHT AN DRUCKER
0324           A201    LDX #1              ;2.BYTE
0326           8E7BA3  STX HX              ;RESET COUNTER
0329           4CB102  JMP EXIT1           ;UND AN DRUCKER
032C                                       ;CNTR A-Z WERDEN ZU $01-$1D CON
032C CTL0      291F    AND #%00011111      ;CONVERT A-Z TO #01-$1A
032E           AA      TAX                 ;CTL CODE MAPPING
032F           BD5AA3  LDA CTLTAB,X
0332           48      PHA                 ;CODE TO STACK
0333           20D002  JSR TOGCTL          ;RESET CTL FLAG
0336           4C9402  JMP UPRINT          ;PASS2 FOR ALL CTL CODES
0339
0339 THERMS    BDB903  LDA THERM0,X        ;SEND THERMES TO PRINTER
033C           F006    BEQ TH1
033E           202702  JSR WRITE
0341           E8      INX
0342           D0F5    BNE THERMS
0344 TH1       60      RTS
0345                                       ;STEUERZEICHEN-TABELLE
0345 CODES     00      .BYT $00,$FF,$5E,DC2
0346           FF
0347           5E
0348           12
0349           13      .BYT DC3,DC1,CR,LF
034A           11
034B           0D
034C           0A
034D           14      .BYT DC4,FF
034E           0C
034F                                       ;PARSER ZIEL-TABELLE
034F AKTION    B402    .WOR EXIT0,EXIT0,TOGCTL,TOGHEX
0351           B402
0353           D002
0355           CC02
0357           D402    .WOR TOGPAG,NEWINI,PAG0,PAG0
0359           DF02
035B           E502
035D           E502
035F           0003    .WOR TAB0,FF0
0361           F402
0363                                       ;TEXTSTRINGS ZUM DISPLAY
0363 M1        0D      .BYT $0D,'PRINTER DOWN',$CE
0364           5052
036F           CE
0370 M2        0D      .BYT $0D,'PRINTER ERRO',$D2
0371           5052
037D           D2
037E M3        0D      .BYT $0D,'PAPER EMPTY-STO',$D0
037F           5041
038E           D0
```

```
038F M4         0D      .BYT $0D,'VTAB',$BD
0390            5654
0394            BD
0395                                    ;CNTRL-CODE MAPPING TABELLE
0395 TABL1      00      .BYT 0,0,2,3,0,0,0,7
0396            00
0397            02
0398            03
0399            00
039A            00
039B            00
039C            07
039D            00      .BYT 0,0,10,0,12,13,14,15
039E            00
039F            0A
03A0            00
03A1            0C
03A2            0D
03A3            0E
03A4            0F
03A5            10      .BYT 16,17,18,19,20,0,0,0
03A6            11
03A7            12
03A8            13
03A9            14
03AA            00
03AB            00
03AC            00
03AD            00      .BYT 0,0,0,27,28,0,0,0
03AE            00
03AF            00
03B0            1B
03B1            1C
03B2            00
03B3            00
03B4            00
03B5            06      .BYT 6,1,$20,66   ;INIT. TABELLE
03B6            01
03B7            20
03B8            42
03B9 THERM0     1B      .BYT ESC,'F',72,0
03BA            46
03BB            48
03BC            00
03BD                                    ;INITIALISIERUNG FUER PROTOKOLLB
03BD                                    ;************* [F2] ************.
03BD INITD1     A9D4    LDA #<DIPRIN    ;VECTOR PRINTROUTINE IN DILINK
03BF            8D06A4  STA DILINK
03C2            A903    LDA #>DIPRIN
03C4            8D07A4  STA DILINK+1
03C7            A90F    LDA #<VIDEO     ;VECTOR 'VIDEO' IN DILIN2
03C9            8D7EA3  STA DILIN2
03CC            A98C    LDA #>VIDEO
03CE            8D7FA3  STA DILIN2+1
03D1            4C0082  JMP INITVP      ;INITIALISIERE VIA & DRUCKER
03D4                                    ;CHARACTER IM ACCU AN DRUCKER
03D4 DIPRIN     2C11A4  BIT PRIFLG      ;PRINTER 'ON' ?        & DISPLAY
03D7            1007    BPL DP1
```

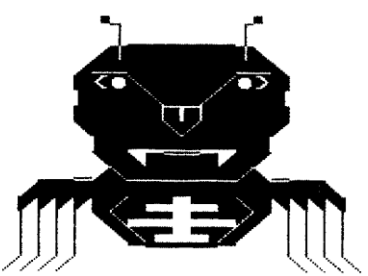
```
03D9            48      PHA
03DA            297F    AND #%01111111     ;MASK BIT 7
03DC            202702  JSR WRITE          ;AN DRUCKER
03DF            68      PLA
03E0 DP1        6C7EA3  JMP (DILIN2)       ;WEITER AN VIDEO DISPLAY
03E3
03E3                    .END
```

Ein Anwendungsbeispiel mit Quelltext, Ausdruck und CTLTAB

```
<L>IN=ON
<T>
;^EW^R 80 3E 49 49 49 3E 00 ^R
;^EW^R 81 63 5D 49 49 63 00 ^R
;^R 81 80 ^R
;^S    TOGGLE FUNKTION : AUTO FORM FEED ON
;^A    ILLEGAL CONTROL CODE, MUST BE IGNORED
;^P20 TABULATOR TESTZEILE TAB(20)
;^EA TESTPROGRAMM ^M
;^EB ****** RECHNUNG *********^M^O
; WOLFGANG ^N RADELOFF ^O PINNEBERG
;^G   BELL
;
;E8
;    TOGGLE FUNKTION : AUTO FORM FEED ON
;    ILLEGAL CONTROL CODE, MUST BE IGNORED
;                    TABULATOR TESTZEILE TAB(20)
; TESTPROGRAMM
; ****** RECHNUNG *********
; WOLFGANG   RADELOFF   PINNEBERG
;   BELL
;
CTLTAB .BYT 0,0,2,3,0,0,0,7
       .BYT 0,0,10,0,12,13,14,15
       .BYT 16,17,18,19,20,0,0,0
       .BYT 0,0,0,27,28,0,0,0
       ;
```

□□

SYS30464



High Resolution Screen Dump

Siehe im Heftinneren

Martin Albrecht, 4350 Recklinghausen

MOVE und RELOCATE

Das vorliegende Programm dient dem Verschieben und Umrechnen von 6502 Maschinencode. Es verwendet die im 'First Book of KIM' erschienenen Dienstprogramme MOVIT von Lew Edwards und RELOCATE von Jim Butterfield. Die mächtigen Eigenschaften dieser Programme kommen erst durch eine interaktive Benutzerführung zur Geltung.

Das Programm ist auf jedem 6502-Rechner lauffähig, der über eine ASCII Eingabe- und Druckroutine verfügt (siehe INPUT und OUTPUT). Insofern liegt eine systemfreie Bearbeitung vor, die sicher für die EPROM-Programmierung und auch für das Experimentieren mit ROM-Routinen interessant ist, z.B. mit dem AIM-Monitor nach REMON.

In der Seite Null werden 10 Byte belegt, alle anderen Variablenadressen sind frei wählbar. Besitzer eines AIM können den Bereich von $4C7 bis $537 streichen und stattdessen die Monitorroutinen CRLOW, WRAX und ADDNE und die Variablen ADDR=$A41C und CKSUM=$A41E benutzen. Im ungünstigsten Fall ist das Programm 1051 Byte lang. Es sei auch auf eine andere frühere Bearbeitung hingewiesen /1/.

Zum Arbeiten mit dem Programm: Der Programmstart erfolgt mit JMP ALMAR (JMP $0200). Die folgenden Dienstleistungen werden angeboten: Verschieben, Verschieben und Umrechnen am neuen Platz, Umrechnen am alten oder neuen Platz. Beim Umrechnen besteht jeweils die Möglichkeit, enthaltene Tabellen auszusparen und hinterher Adreßtabellen umzurechnen.

Die Programmbedienung geht aus dem Kopf des Listings hervor. Alle Adreßeingaben beziehen sich immer auf den alten Platz. Grundsätzlich ist die Adresse des ersten und des letzten Bytes einzugeben.

Literatur: The First Book of KIM, Argonne, Ill., 1977, ferner /1/ Dohmann I.. MOVE And RELOCATE. 65xx MICRO MAG Nr. 7, 1979.

```
LOC. SYMBOL   CODE

              PASS 1
0115          PASS 2
0000
0000          ;VERSCHIEBEN UND UMRECHNEN VON 6502 MASCHINENCODE
0000          ;(C) MARTIN ALBRECHT,AM BAERENBACH 18,4350 RECKLINGHAUSEN
0000
0000          ;PROGRAMMBEDIENUNG:
0000          ;VON (ADRESSE)(CR) BIS (ADRESSE)(CR):
0000          ;ALTER BEREICH
0000          ;NACH (ADRESSE)(CR): NEUER BEGINN
0000          ;MENUE: AUSWAHL MIT 'Y','J', WEITER MIT ANDERER TASTE
0000          ;ALT ---> NEU         NUR VERSCHIEBEN
0000          ;ALT ---> NEU UMR.    VERSCHIEBEN, UMRECHNEN AM NEUEN PLATZ
0000          ;ALT UMR. ---> NEU    UMRECHNEN AM ALTEN PLATZ, VERSCHIEBEN
0000          ;ALT UMR.             NUR UMRECHNEN AM ALTEN PLATZ
0000          ;NEU UMR.             NUR UMRECHNEN AM NEUEN PLATZ
0000
0000          ;UMR. BIETET DIE FOLGENDEN LEISTUNGEN:
0000          ;UEBERSPRINGEN VON TABELLEN:
0000          ;TAB VON (ADRESSE,CR)(CR) BIS (ADRESSE)(CR)
0000          ;ZU UEBERSPRINGENDE TABELLEN IN REIHENFOLGE EINGEBEN
0000          ;'CR' WENN KEINE (WEITERE) TABELLE VORLIEGT
0000          ;VOM PROGRAMM ERKANNTE TABELLEN WERDEN ANGEMAHNT MIT
0000          ;(ERKANNTE ADRESSE) ? TAB BIS (ADRESSE)(CR)
0000          ;DAS TABELLENENDE IST EINZUGEBEN
```

```
0000          ;UMRECHNEN VON ADRESSTABELLEN:
0000          ;TABELLEN UMRECHNEN ? (J,Y,BEL. ANDERES ZEICHEN)
0000          ;AUSWAHL MIT 'Y' ODER 'J', SONST ANDERE TASTE
0000          ;TAB VON (ADRESSE,CR)(CR) BIS (ADRESSE)(CR)
0000          ;TABELLEN IN REIHENFOLGE EINGEBEN
0000          ;'CR' WENN KEINE WEITERE TABELLE VORLIEGT
0000          ;INKREMENT ? (WERT,CR)(CR)
0000          ;ABSTAND DER ADRESSEN IN DER TABELLE EINGEBEN
0000          ;'CR' NIMMT DEN WERT 2 AN FUER REINE ADRESSTABELLEN
0000
0000          ;'CR' MEINT 'CR' ODER 'SPACE'
0000          ;FEHLERHAFTE ADRESSEN WERDEN NICHT ERKANNT
0000
0000          ;ASCII INPUT UND OUTPUT (UEBERGABE IM ACCU)
0000          ;JE NACH RECHNER ANPASSEN !
0000 INPUT          =$E973          ;EIN ZEICHEN HOLEN UND ANZEIGEN
0000 OUTPUT         =$E97A          ;EIN ZEICHEN DRUCKEN
0000
0000          ;PROGRAMMVARIABLEN
0000          ;ALLE FREI WAEHLBAR, NUR ASTART,AEND,NSTART,NEND UND ZEIG
0000          ;MUESSEN 0-PAGE ADRESSEN SEIN.
0000          ;DIE ERSTEN 4 MUESSEN UNBEDINGT IN FOLGE STEHEN !
0000                *=$C0
00C0 ASTART         *=*+2           ;ALTER BEGINN
00C2 AEND           *=*+2           ;ALTES ENDE
00C4 NSTART         *=*+2           ;NEUER BEGINN
00C6 NEND           *=*+2           ;NEUES ENDE
00C8 ZEIG           *=*+2           ;AKTUELLER ZEIGER
00CA VDIST          *=*+2           ;VERSCHIEBEDISTANZ
00CC ADIST          *=*+2           ;PLATZDISTANZ
00CE CKAD           *=*+2           ;TESTADRESSE F. TABELLEN
00D0 GOON           *=*+2           ;WEITERMACHADRESSE F. TABELLEN
00D2 CNTL           *=*+1           ;BYTEZAEHLER LOW
00D3 CNTH           *=*+1           ;BYTEZAEHLER HIGH
00D4 NBOUND         *=*+2           ;NEUER BEGINN
00D6 REND           *=*+2           ;ENDE F. UMR.
00D8 ABOUND         *=*+2           ;ALTER BEGINN
00DA EBOUND         *=*+2           ;ALTES ENDE
00DC Z1             *=*+1           ;BYTE F. UMR.
00DD Z2             *=*+1           ;BYTE F. UMR.
00DE ADDR           =$A41C
00DE CKSUM          =$A41E
00DE
00DE                * =$200
0200
0200          ;HOLE DIE BEREICHSGRENZEN
0200 ALMAR    D8    CLD
0201 V1       20C704 JSR CRLOW      ;CR
0204          A000   LDY £0         ;VON
0206          206E04 JSR TXTAD      ;DRUCKE TEXT, HOLE ADRESSE
0209          B0F6   BCS V1         ;FEHLER
020B          D0F4   BNE V1         ;KEINE EINGABE
020D          204F04 JSR GETVAL     ;ADRESSE IN A UND X
0210
0210          ;ALTER BEGINN:
0210          85C0   STA ASTART     ;NACH ASTART
0212          86C1   STX ASTART+1
0214          85D8   STA ABOUND     ;UND ABOUND
```

```
0216          86D9    STX ABOUND+1
0218
0218 ERR2     A004    LDY £BIS-TEXT  ;BIS
021A          206E04  JSR TXTAD
021D          B0F9    BCS ERR2
021F          D0F7    BNE ERR2
0221          204F04  JSR GETVAL
0224
0224          ;ALTES ENDE:
0224          85C2    STA AEND       ;NACH AEND
0226          86C3    STX AEND+1
0228          85DA    STA EBOUND     ;UND EBOUND
022A          86DB    STX EBOUND+1
022C
022C          ;BERECHNE BLOCKLAENGE DES ALTEN BEREICHS
022C          38      SEC
022D          E5C0    SBC ASTART
022F          85D2    STA CNTL       ;ERGEBNIS NACH CNTL,CNTH
0231          8A      TXA
0232          E5C1    SBC ASTART+1
0234          85D3    STA CNTH
0236
0236 V3       20C704  JSR CRLOW
0239 ERR3     A009    LDY £NACH-TEXT   ;NACH
023B          206E04  JSR TXTAD
023E          B0F6    BCS V3
0240          D0F4    BNE V3
0242          204F04  JSR GETVAL
0245
0245          ;NEUER ANFANG:
0245          85D4    STA NBOUND     ;NACH NBOUND
0247          86D5    STX NBOUND+1
0249          85C4    STA NSTART     ;UND NSTART
024B          86C5    STX NSTART+1
024D          85C8    STA ZEIG       ;ZEIGER FUER UMRECHNEN
024F          86C9    STX ZEIG+1
0251
0251          ;BERECHNE NEUES ENDE
0251          18      CLC
0252          65D2    ADC CNTL
0254          85C6    STA NEND       ;NACH NEND UND REND
0256          85D6    STA REND
0258          8A      TXA
0259          65D3    ADC CNTH
025B          85C7    STA NEND+1
025D          85D7    STA REND+1
025F
025F          ;WELCHE LEISTUNG ? :
025F HMENUE   A035    LDY £MSG1-TEXT   ; ALT --> NEU
0261          20BE04  JSR QUEST      ;CR,TEXT, HOLE ADRESSE
0264          D003    BNE M2
0266          4C9802  JMP MOVE       ;VERSCHIEBEN
0269
0269 M2       A041    LDY £MSG2-TEXT   ;ALT --> NEU UMR.
026B          20BE04  JSR QUEST
026E          D006    BNE M3
0270          209802  JSR MOVE       ;VERSCHIEBEN
0273          4CF202  JMP NEUREL     ;UMRECHNEN AM NEUEN PLATZ
```

```
0276 M3       A052   LDY £MSG3-TEXT   ;ALT UMR. --> NEU
0278          20BE04 JSR QUEST
027B          D006   BNE M4
027D          20D602 JSR ALTREL       ;UMRECHNEN AM ALTEN PLATZ
0280          4C9802 JMP MOVE         ;VERSCHIEBEN
0283
0283 M4       A063   LDY £MSG4-TEXT   ;ALT UMR.
0285          20BE04 JSR QUEST
0288          D003   BNE M5
028A          4CD602 JMP ALTREL       ;UMRECHNEN AM ALTEN PLATZ
028D
028D M5       A06C   LDY £MSG5-TEXT   ;NEU UMR.
028F          20BE04 JSR QUEST
0292          D003   BNE NOTASK
0294          4CF202 JMP NEUREL       ;UMRECHNEN AM NEUEN PLATZ
0297 NOTASK   60     RTS
0298
0298          ;VERSCHIEBE ALT --> NEU
0298          ;MIT MOVIT VON LEW EDWARDS
0298          ;SIEHE 65XX MICRO MAG NR.7 S. 24 FF
0298 MOVE     E6D2   INC CNTL
029A          E6D3   INC CNTH
029C          A0FF   LDY £$FF
029E
029E          ;NEUER BEREICH - ALTER BEREICH
029E          200003 JSR CVDIST
02A1
02A1 LOOP     A200   LDX £0           ;VORWAERTSVERSCHIEBEN
02A3          9002   BCC VA
02A5          A202   LDX £2           ;RUECKWAERTSVERSCHIEBEN
02A7 VA       A1C0   LDA (ASTART,X)   ;VERSCHIEBE
02A9          81C4   STA (NSTART,X)
02AB          9014   BCC VA2
02AD
02AD          ;VERMINDERE ADRESSEN FUER RUECKWAERTS
02AD          C6C2   DEC AEND
02AF          98     TYA
02B0          45C2   EOR AEND
02B2          D002   BNE NOTFF1
02B4          C6C3   DEC AEND+1
02B6 NOTFF1   C6C6   DEC NEND
02B8          98     TYA
02B9          45C6   EOR NEND
02BB          D002   BNE NOTFF2
02BD          C6C7   DEC NEND+1
02BF NOTFF2   B00C   BCS CNTDEC
02C1
02C1          ;ERHOEHE ADRESSEN FÜR VORWAERTS
02C1 VA2      E6C0   INC ASTART
02C3          D002   BNE NOT01
02C5          E6C1   INC ASTART+1
02C7 NOT01    E6C4   INC NSTART
02C9          D002   BNE CNTDEC
02CB          E6C5   INC NSTART+1
02CD
02CD          ;DEKREMENTIERE BYTEZAEHLER
02CD CNTDEC   C6D2   DEC CNTL
02CF          D0D0   BNE LOOP         ;SCHLEIFE
```

```
02D1          C6D3    DEC CNTH
02D3          D0CC    BNE LOOP        ;SCHLEIFE
02D5          60      RTS
02D6
02D6          ;VORBEREITUNG F. UMRECHNEN AM ALTEN PLATZ
02D6 ALTREL   200003  JSR CVDIST      ;BERECHNE VERSCHIEBEDISTANZ
02D9          A900    LDA £0          ;PLATZDISTANZ AUF 0
02DB          85CC    STA ADIST
02DD          85CD    STA ADIST+1
02DF          A5D8    LDA ABOUND      ;ZEIG=UMRECHENZEIGER
02E1          A6D9    LDX ABOUND+1    ;AUF ALTEN BEGINN
02E3          85C8    STA ZEIG
02E5          86C9    STX ZEIG+1
02E7          A5DA    LDA EBOUND      ;REND=ENDE FUER UMRECHNEN
02E9          A6DB    LDX EBOUND+1    ;AUF ALTES ENDE
02EB          85D6    STA REND
02ED          86D7    STX REND+1
02EF          4C0E03  JMP RELOC       ;ZUM UMRECHNEN
02F2
02F2          ;VORBEREITUNG F. UMRECHNEN AM NEUEN PLATZ
02F2 NEUREL   200003  JSR CVDIST      ;VERSCHIEBEDISTANZ
02F5          A5CA    LDA VDIST
02F7          A6CB    LDX VDIST+1
02F9          85CC    STA ADIST       ;PLATZDISTANZ ZUM ALTEN PLATZ
02FB          86CD    STX ADIST+1
02FD          4C0E03  JMP RELOC       ;ZUM UMRECHNEN
0300
0300          ;BERECHNE VERSCHIEBEDISTANZ
0300 CVDIST   38      SEC             ;NEUER START - ALTER START
0301          A5D4    LDA NBOUND
0303          E5D8    SBC ABOUND
0305          85CA    STA VDIST
0307          A5D5    LDA NBOUND+1
0309          E5D9    SBC ABOUND+1
030B          85CB    STA VDIST+1
030D          60      RTS
030E
030E          ;ZUM UMRECHNEN
030E RELOC    201403  JSR RELOZ1
0311
0311          ;ZUM TABELLEN-UMRECHNEN
0311          4CE703  JMP RADR
0314
0314          ;UMRECHNEN:
0314 RELOZ1
0314          ;HOLE ZU UEBERSPRINGENDE TABELLE
0314          ;TABELLENGRENZEN IN CKAD UND GOON
0314 TAB      207804  JSR UTAB        ;HOLE GRENZEN
0317 RCHECK   206404  JSR ZUENDE      ;ENDE (REND) ERREICHT ?
031A          9001    BCC CK          ;NEIN
031C          60      RTS
031D          ;TABELLENANFANG ERREICHT ?
031D CK       38      SEC             ;UMRECHENZEIGER SCHON AUF CKAD ?
031E          A5C8    LDA ZEIG
0320          E5CE    SBC CKAD
0322          A5C9    LDA ZEIG+1
0324          E5CF    SBC CKAD+1
0326          900E    BCC RECALC      ;NEIN, WEITER UMRECHNEN
```

```
0328           ;TABELLENANFANG ERREICHT
0328           ;SETZE ZEIGER AUF TABELLENENDE
0328           A5D0    LDA GOON        ;TABELLENENDE
032A           A6D1    LDX GOON+1
032C           85C8    STA ZEIG        ;NACH ZEIG
032E           86C9    STX ZEIG+1
0330           206404  JSR ZUENDE      ;ENDE (REND) ERREICHT ?
0333           90DF    BCC TAB         ;NEIN
0335           60      RTS
0336
0336           ;JIM BUTTERFIELD'S RELOCATE
0336           ;SIEHE 65XX MICRO MAG NR.7 S.24 FF
0336 RECALC    A000    LDY £0
0338           B1C8    LDA (ZEIG),Y
033A           A8      TAY
033B           A207    LDX £7
033D RC0       98      TYA
033E           3DB805  AND TB-1,X
0341           5DBF05  EOR TB+6,X
0344           F003    BEQ RC1
0346           CA      DEX
0347           D0F4    BNE RC0
0349 RC1       BCC705  LDY TB+14,X
034C           300D    BMI RC2
034E           F022    BEQ RC3
0350 RC5       E6C8    INC ZEIG
0352           D002    BNE RC4
0354           E6C9    INC ZEIG+1
0356 RC4       88      DEY
0357           D0F7    BNE RC5
0359           F0BC    BEQ RCHECK
035B RC2       C8      INY
035C           3060    BMI EVTAB       ;OFFENBAR LIEGT EINE TABELLE VOR
035E           C8      INY
035F           B1C8    LDA (ZEIG),Y
0361           AA      TAX
0362           C8      INY
0363           B1C8    LDA (ZEIG),Y
0365           209E03  JSR ADDIST
0368           91C8    STA (ZEIG),Y
036A           88      DEY
036B           8A      TXA
036C           91C8    STA (ZEIG),Y
036E           A003    LDY £3
0370           10DE    BPL RC5
0372 RC3       C8      INY
0373           A6C8    LDX ZEIG
0375           A5C9    LDA ZEIG+1
0377           209E03  JSR ADDIST
037A           86DC    STX Z1
037C           A2FF    LDX £$FF
037E           B1C8    LDA (ZEIG),Y
0380           18      CLC
0381           6902    ADC £2
0383           3001    BMI RC6
0385           E8      INX
0386 RC6       86DD    STX Z2
0388           18      CLC
```

```
0389          65C8    ADC ZEIG
038B          AA      TAX
038C          A5DD    LDA Z2
038E          65C9    ADC ZEIG+1
0390          209E03  JSR ADDIST
0393          CA      DEX
0394          CA      DEX
0395          8A      TXA
0396          38      SEC
0397          E5DC    SBC Z1
0399          91C8    STA (ZEIG),Y
039B          C8      INY
039C          10B2    BPL RC5
039E
039E          ;RECHNE ADRESSE UM, WENN IN ABOUND,EBOUND GRENZEN
039E ADDIST   C5DB    CMP EBOUND+1
03A0          900A    BCC KE
03A2          D019    BNE RET
03A4          E4DA    CPX EBOUND
03A6          9004    BCC KE
03A8          F002    BEQ KE
03AA          D011    BNE RET
03AC KE       C5D9    CMP ABOUND+1
03AE          D002    BNE RC7
03B0          E4D8    CPX ABOUND
03B2 RC7      9009    BCC RET
03B4          48      PHA
03B5          8A      TXA
03B6          18      CLC
03B7          65CA    ADC VDIST
03B9          AA      TAX
03BA          68      PLA
03BB          65CB    ADC VDIST+1
03BD RET      60      RTS
03BE
03BE          ;EINE NICHT ANGEGEBENE TABELLE WURDE ERKANNT
03BE          ;FRAGE TABELLENENDE AB
03BE EVTAB    20C704  JSR CRLOW
03C1          38      SEC             ;BERECHNE ALTE ADRESSE
03C2          A5C8    LDA ZEIG
03C4          E5CC    SBC ADIST
03C6          AA      TAX
03C7          A5C9    LDA ZEIG+1
03C9          E5CD    SBC ADIST+1
03CB          20CC04  JSR WRAX        ;ZEIGE SIE AN
03CE          A016    LDY £TABBIS-TEXT   ;TAB BIS ?
03D0          206E04  JSR TXTAD       ;HOLE TABELLENENDE
03D3          B0E9    BCS EVTAB
03D5          D0E7    BNE EVTAB
03D7          204F04  JSR GETVAL
03DA          38      SEC             ;ADDIERE PLATZDISTANZ
03DB          65CC    ADC ADIST
03DD          85C8    STA ZEIG        ;NACH ZEIG
03DF          8A      TXA
03E0          65CD    ADC ADIST+1
03E2          85C9    STA ZEIG+1
03E4          4C1703  JMP RCHECK      ;WEITER UMRECHNEN
```

```
03E7          ;TABELLEN UMRECHNEN
03E7 RADR     A021    LDY £ADRTAB-TEXT   ;TABELLEN UMRECHNEN ?
03E9          20BE04  JSR QUEST
03EC          F001    BEQ RECH           ;JA
03EE          60      RTS
03EF
03EF          ;HOLE TABELLENADRESSE
03EF RECH     207804  JSR UTAB           ;TABELLENGRENZEN HOLEN
03F2          A5CE    LDA CKAD
03F4          A6CF    LDX CKAD+1
03F6          85C8    STA ZEIG           ;ANFANG IN ZEIG
03F8          86C9    STX ZEIG+1
03FA
03FA          ;HOLE INKREMENT FUER UMRECHNEN
03FA          20C704  JSR CRLOW
03FD I1       A075    LDY £INCR-TEXT     ;INCREMENT ?
03FF          206E04  JSR TXTAD          ;FORDERE EINGABE AN
0402          B0F9    BCS I1             ;FEHLER
0404          F00A    BEQ GOTINC         ;ERFOLGTE EINGABE
0406          A902    LDA £2             ;KEINE EINGABE, NIMM 2 AN
0408          A200    LDX £0
040A          85DC    STA Z1
040C          86DD    STX Z2
040E          F007    BEQ NXT
0410 GOTINC   204F04  JSR GETVAL         ;HOLE EINGABE
0413          85DC    STA Z1             ;NACH Z1,Z2
0415          86DD    STX Z2
0417 NXT      206404  JSR ZUENDE         ;ENDE ERREICHT ?
041A          9001    BCC LKEND          ;NEIN
041C          60      RTS
041D
041D          ;TESTE, OB ZEIG TABELLENENDE ERREICHT HAT
041D LKEND    38      SEC
041E          A5C8    LDA ZEIG
0420          E5D0    SBC GOON
0422          A5C9    LDA ZEIG+1
0424          E5D1    SBC GOON+1
0426          9002    BCC NOTYET         ;NEIN, WEITER UMRECHNEN
0428          B0C5    BCS RECH           ;NAECHSTE TABELLE
042A          ;RECHNE TABELLENEINTRAG UM
042A NOTYET   A000    LDY £0             ;TABELLENEINTRAG IN A UND X
042C          B1C8    LDA (ZEIG),Y
042E          AA      TAX
042F          C8      INY
0430          B1C8    LDA (ZEIG),Y
0432          209E03  JSR ADDIST         ;UMRECHNEN MIT ADDIST
0435          91C8    STA (ZEIG),Y       ;UND WIEDER EINSETZEN
0437          88      DEY
0438          8A      TXA
0439          91C8    STA (ZEIG),Y
043B          204104  JSR INZEIG         ;ZEIG+INKREMENT
043E          4C1704  JMP NXT            ;NAECHSTER EINTRAG
0441          ;INKREMENTIERE ZEIG UM INKREMENT
0441 INZEIG   18      CLC
0442          A5C8    LDA ZEIG
0444          65DC    ADC Z1
0446          85C8    STA ZEIG
0448          A5C9    LDA ZEIG+1
```

```
044A          65DD    ADC Z2
044C          85C9    STA ZEIG+1
044E          60      RTS
044F
044F          ;LADE ADRESSE AUS ADDR IN A UND X
044F GETVAL   AD1CA4  LDA ADDR
0452          AE1DA4  LDX ADDR+1
0455          60      RTS
0456
0456          ;GIB TEXT AUS
0456          ;LETZTES ZEICHEN HAT BIT 7=1
0456 KEPU     B93805  LDA TEXT,Y      ;HOLE ZEICHEN
0459          48      PHA
045A          297F    AND £$7F        ;BIT 7 AUF 0
045C          207AE9  JSR OUTPUT      ;DRUCKEN
045F          C8      INY
0460          68      PLA
0461          10F3    BPL KEPU        ;WEITER, WENN BIT 7=0 WAR
0463          60      RTS
0464
0464          ;TESTE OB ZEIG ALTES ENDE ERREICHT HAT
0464 ZUENDE   38      SEC             ;ZEIG-REND
0465          A5C8    LDA ZEIG
0467          E5D6    SBC REND
0469          A5C9    LDA ZEIG+1
046B          E5D7    SBC REND+1
046D          60      RTS
046E
046E          ;SENDE TEXT, HOLE ADRESSE
046E TXTAD    205604  JSR KEPU        ;SENDE TEXT
0471          20FF04  JSR ADDNE       ;HOLE ADRESSE
0474          AD1EA4  LDA CKSUM
0477          60      RTS
0478
0478          ;TABELLENGRENZEN HOLEN UND ANPASSEN
0478 UTAB     20C704  JSR CRLOW
047B R1       A00E    LDY £TABV-TEXT   ;TAB VON
047D          206E04  JSR TXTAD
0480          B0F9    BCS R1
0482          F00B    BEQ ISAD        ;ERFOLGTE EINGABE
0484          A5D6    LDA REND        ;KEINE EINGABE
0486          A6D7    LDX REND+1      ;SETZE REND EIN
0488          85CE    STA CKAD
048A          86CF    STX CKAD+1
048C          38      SEC
048D          B024    BCS TAE         ;FERTIG
048F ISAD     204F04  JSR GETVAL      ;HOLE START
0492          18      CLC             ;ADDIERE PLATZDISTANZ
0493          65CC    ADC ADIST
0495          85CE    STA CKAD        ;NACH CKAD
0497          8A      TXA
0498          65CD    ADC ADIST+1
049A          85CF    STA CKAD+1
049C R2       A004    LDY £BIS-TEXT   ;BIS
049E          206E04  JSR TXTAD       ;HOLE ADRESSE
04A1          B0F9    BCS R2
04A3          D0F7    BNE R2
04A5          204F04  JSR GETVAL
```

```
04A8          38      SEC             ;ADDIERE PLATZDISTANZ
04A9          65CC    ADC ADIST
04AB          85D0    STA GOON        ;NACH GOON
04AD          8A      TXA
04AE          65CD    ADC ADIST+1
04B0          85D1    STA GOON+1
04B2          18      CLC
04B3 TAE      60      RTS
04B4
04B4          ;TESTE AUF 'J' ODER 'Y'
04B4 YESNO    2073E9  JSR INPUT
04B7          C94A    CMP £'J'
04B9          F002    BEQ YESRET
04BB          C959    CMP £'Y'
04BD YESRET   60      RTS
04BE
04BE          ;SENDE CR,TEXT; LIES TASTATUR
04BE QUEST    20C704  JSR CRLOW
04C1          205604  JSR KEPU
04C4          4CB404  JMP YESNO
04C7
04C7          ;'CR' DRUCKEN
04C7 CRLOW    A90D    LDA £13
04C9          4C7AE9  JMP OUTPUT

04CC          ;DRUCKE A DANN X IN ASCII
04CC WRAX     20D004  JSR NUMA
04CF          8A      TXA
04D0          ;EIN BYTE=2 ASCII ZEICHEN DRUCKEN
04D0 NUMA     48      PHA
04D1          4A      LSR A
04D2          4A      LSR A
04D3          4A      LSR A
04D4          4A      LSR A
04D5          20DB04  JSR NOUT
04D8          68      PLA
04D9          290F    AND £$F
04DB NOUT     18      CLC
04DC          6930    ADC £$30
04DE          C93A    CMP £$3A
04E0          9002    BCC OUTIT
04E2          6906    ADC £6          ;CARRY=1
04E4 OUTIT    4C7AE9  JMP OUTPUT
04E7          ;ACCU VON ASCII IN HEX UMFORMEN
04E7 HEX      C930    CMP £$30        ;≙ 30 ?
04E9          9012    BCC NHEX
04EB          C947    CMP £$47        ;≙ 47 ?
04ED          B00E    BCS NHEX
04EF          C93A    CMP £$3A        ;≙ 10 ?
04F1          9006    BCC H1
04F3          C941    CMP £$41        ;≙ 10 ?
04F5          9006    BCC NHEX
04F7          6908    ADC £8          ;+9 (C=1)
04F9 H1       290F    AND £$F         ;4 MSB AUSBLENDEN
04FB          18      CLC
04FC          60      RTS
04FD NHEX     38      SEC
04FE          60      RTS
```

```
04FF           ;4 BYTE ADRESSE HOLEN
04FF           ;DIE 4 LETZTEN ZEICHEN GELTEN
04FF ADDNE     A000    LDY £0
0501           8C1CA4  STY ADDR        ;LOESCHE ERGEBNISSPEICHER
0504           8C1DA4  STY ADDR+1
0507           8C1EA4  STY CKSUM       ;FLAG, GESETZT WENN NUR CR
050A ADDN1     2073E9  JSR INPUT       ;HOLE ZEICHEN
050D           C90D    CMP £13         ;TESTE OB CR ODER SPACE
050F           F01E    BEQ DONE
0511           C920    CMP £' '
0513           F01A    BEQ DONE
0515           C8      INY
0516           20E704  JSR HEX         ;UMWANDELN
0519           B01C    BCS ADRET       ;NICHT HEX
051B           A204    LDX £4
051D LEFT      0E1CA4  ASL ADDR        ;ERGEBNIS 4* LINKSSCHIEBEN
0520           2E1DA4  ROL ADDR+1
0523           CA      DEX
0524           D0F7    BNE LEFT
0526           0D1CA4  ORA ADDR        ;ERGEBNIS VON HEX DAZU
0529           8D1CA4  STA ADDR
052C           18      CLC
052D           90DB    BCC ADDN1
052F DONE      C000    CPY £0
0531           D003    BNE VALAD       ;GUELTIGE ADRESSE
0533           8D1EA4  STA CKSUM       ;FLAG, DA KEINE EINGABE
0536 VALAD     18      CLC
0537 ADRET     60      RTS
0538
0538
0538
0538           ;TEXTTABELLEN FUER KEPU
0538 TEXT      564F4E .BYT 'VON',$20+$80
053B           A0
053C BIS       2042   .BYT ' BIS',$20+$80
0540           A0
0541 NACH      4E41   .BYT 'NACH',$20+$80
0545           A0
0546 TABV      5441   .BYT 'TAB VON',$20+$80
054D           A0
054E TABBIS    203F   .BYT ' ? TAB BIS',$20+$80
0558           A0
0559 ADRTAB    5441   .BYT 'TABELLEN '
0562           554D   .BYT 'UMRECHNEN ',$3F+$80
056C           BF
056D MSG1      414C   .BYT 'ALT --> NEU',$20+$80
0578           A0
0579 MSG2      414C   .BYT 'ALT --> NEU UMR.',$20+$80
0589           A0
058A MSG3      414C   .BYT 'ALT UMR. --> NEU',$20+$80
059A           A0
059B MSG4      414C   .BYT 'ALT UMR.',$20+$80
05A3           A0
05A4 MSG5      4E45   .BYT 'NEU UMR.',$20+$80
05AC           A0
05AD INCR      494E   .BYT 'INKREMENT ?',$20+$80
05B8           A0
```

```
05B9            ;TABELLEN FUER RELOCATE
05B9 TB   0C      .BYT $0C,$1F,$0D,$87,$1F,$FF,3,$0C
05BA      1F
05BB      0D
05BC      87
05BD      1F
05BE      FF
05BF      03
05C0      0C
05C1      19      .BYT $19,8,0,$10,$20,3,2,$FF
05C2      08
05C3      00
05C4      10
05C5      20
05C6      03
05C7      02
05C8      FF
05C9      FF      .BYT $FF,1,1,0,$FF,$FE
05CA      01
05CB      01
05CC      00
05CD      FF
05CE      FE
05CF
05CF              *=$112
0112      4C0002  JMP ALMAR                              □□
```

Peter Engels, 5308 Rheinbach-Niederdrees

High Resolution Screen Dump

von CBM auf EPSON Typ 2/3

Dieses Programm erlaubt eine naturgetreue Schirmkopie auf einen Epson-Drucker mit Einzelnadelansteuerung (Epson Typ 2, Typ 3 oder mit Graftrax 80-ROMs). Es belegt mit Character-ROM den RAM-Bereich $7700-8000 im CBM und wird mit SYS 30464 aufgerufen.

Bei 1. Aufruf schützt sich das Programm zuerst selbst vor BASIC (Zeilen 330-430. Ab dem 2. Aufruf erfolgt eine Kopie des Bildschirmes. Dazu wird der Bit-Image-Mode der o.g. Drucker verwendet. Eine Änderung auf andere Drucker ist ohne weiteres durch Ändern der Steuerzeichen in Zeilen 620-670 bzw. 1400-1420 möglich. Das Programm übergibt die Zeichen an den IEC-Bus wie in /1/ beschrieben. Eine weitere Erklärung braucht daher nicht zu erfolgen. Damit der Rechner weiß, wie die Zeichen für den Drucker gebildet werden, liegt im Bereich von $7900-7FFF eine Kopie des jeweiligen Character-ROMs. Damit das Treiberprogramm von $7700-77F0 kurz bleibt, wird das Character ROM nicht direkt übernommen, sondern die Zeichen werden gedreht.

Original ROM-Darstellung des Buchstaben A:

Sp.Stelle	Hex-Byte	Binär
0008	18	00011000
0009	24	00100100
000A	42	01000010
000B	7E	01111110
000C	42	01000010
000D	42	01000010
000E	42	01000010
000F	00	00000000

Darstellung im RAM:

```
7708        00        00000000
7709        3E        00111110
770A        50        01010000
770B        90        10010000
770C        90        10010000
770D        50        01010000
770E        _ 3E       00111110
```

Durch Drehung der Zeichen ist es nun möglich, diese direkt aus der Tabelle Byte für Byte an den Drucker weiterzugeben. Somit ist im Treiberprogramm keine weitere Umrechnung mehr notwendig, und das Programm kann ziemlich kurz gehalten werden. Die entsprechenden Bytes werden folgendermaßen in der Tabelle ausfindig gemacht:

Beispiel Buchstabe D

Der Buchstabe hat den Bildschirmcode 4 (dez.). Die acht Byte für dieses Zeichen beginnen dann bei 4*8=32 (dez.) = 20 (hex). 20+7800 (hex) = 7820 (hex). - Da mittlerweile sehr viele verschiedene Charactersets in Gebrauch sind, wurde auf einen Memory-Dump verzichtet. Es wurde ein Assemblerprogramm geschrieben, welches im 2. Cassettenbuffer sitzt und ein Character-ROM von A000 nach 7800 kopiert und die Zeichen dabei wie oben beschrieben dreht. Es erhielt den ROM-DREH und ist weiter unten abgedruckt. Das Character-ROM wird dabei wie folgt kopiert:

1. ROM-DREH z.b. mittels TIM in den 2. Cassettenbuffer eingeben.
2. Programm auf Disk oder Cassette mit Namen 'DREH' abspeichern.
3. Rechner ausschalten.
4. Rechner öffnen, Char.-ROM aus Sockel UF10 nehmen und in Sockel UD4 einsetzen (Positionen gelten für CBM 3000).
5. Rechner einschalten. Der Bildschirm bleibt weiß, weil das Char.-ROM fehlt, alles normal!
6. Folgende Tasten drücken: LOAD "DREH",8 (Return), dann warten bis der Ladevorgang beendet ist.
   SYS 826 (Return). Der Cursor verschwindet nun für ca. 1 Sek.
   SYS 4 (Return). Der Rechner befindet sich nun im Monitor.
   S"0:CHR-ROM",08,7800,8000 (Return). Nun das Saving abwarten.
7. Rechner ausschalten und Char.-ROM wieder am alten Platz einstecken.

Nun kann das File "CHR-ROM" wieder eingeladen werden. Dann wird das Treiberprogramm von $7700-77f1 hinzugefügt. Achtung: Vor dem ersten Aufruf ist das gesamte Programm zuerst abzuspeichern!

Die im Listing genannten absoluten Adressen gelten für CBM 3032, in Klammern enthaltene gelten für CBM 4032. - Wem das Kopieren zu aufwendig ist, wende sich mit einem Datenträger an den Autor: Kreisstraße 29, 5308 Rheinbach-Niederdrees.

Literatur:

/1/ Kirchgäßner R.: CBM-Standardroutinen für Daten-I/O, MC 10/82
/2/ EPSON: MX 82/3 Control-Codes.

```
0010 ; ' HIRES SCREEN-DUMP '
0020 ;   ' FUER CBM-3032 '
0030 ; ' & EPSON TYP 2 + 3 '
0040
0050 ;   AUFRUF: SYS 30464
0060
0070 ZEIGER      .DE $21
0080 ZWSP1       .DE $BA              ; ZEILEN-ZAEHLER
0090 ZWSP2       .DE $BC              ; ZEICHEN-ZAEHLER
0100 ZWSP3       .DE $BD              ; BYTE-ZAEHLER
0110 ZWSP4       .DE $BE              ; ZEICHEN-SPEICHER
```

```
0120 ZWSP5       .DE $BF              ; RVS-FLAG
0130 STRADR      .DE $1F
0140 FNLEN       .DE $D1
0150 LA          .DE $D2
0160 SA          .DE $D3
0170 FA          .DE $D4
0180 CHRROM      .DE $7800
0190 PCR         .DE $E84C
0200 OPENFILE    .DE $F524            ; ( $F563 )
0210 CLRCH       .DE $FFCC
0220 CLOSEFILE   .DE $F2AC            ; ( $F2E0 )
0230 CRLFBAS     .DE $C9E2            ; ( $BADF )
0240 BSOUT       .DE $FFD2
0250 CKOUT       .DE $FFC9
0260 NEW         .DE $C55D            ; ( $B5D4 )
0270 WRTWO       .DE $E784            ; ( $D731 )
0280
0290             .BA $7700
0300             .MC $7700
0310             .OS
```

```
ADR. $7700 +     0320
00- A9 77        0330 SAVE       LDA #$77        ; HIMEM AUF
02- A2 00        0340            LDX #$00        ; $7700 SETZEN
04- 86 34        0350            STX *$34
06- 85 35        0360            STA *$35
08- A9 4C        0370            LDA #$4C        ; NACH 1. AUFRUF
0A- A0 1A        0380            LDY #L,START    ; JMP START
0C- A2 77        0390            LDX #H,START    ; NACH SAVE
0E- 8D 00 77     0400            STA SAVE        ; BRINGEN, DA SAVE
11- 8C 01 77     0410            STY SAVE+1      ; NICHT MEHR NOETIG
14- 8E 02 77     0420            STX SAVE+2
17- 4C 5D C5     0430            JMP NEW
1A- A9 7F        0440 START      LDA #127        ; OPEN 127,4,4
1C- 85 D2        0450            STA *LA
1E- A9 04        0460            LDA #4
20- 85 D4        0470            STA *FA
22- 85 D3        0480            STA *SA
24- A9 00        0490            LDA #0          ; FILENAME=""
26- 85 D1        0500            STA *FNLEN
28- 20 24 F5     0510            JSR OPENFILE
2B- A2 7F        0520            LDX #127        ; CMD 127
2D- 20 C9 FF     0530            JSR CKOUT
30- A2 1B        0540            LDX #$1B        ; ESC "A"+8
32- A9 41        0550            LDA #$41        ; LF = 8/72 INCH
34- 20 84 E7     0560            JSR WRTWO
37- A9 08        0570            LDA #$08
39- 20 D2 FF     0580            JSR BSOUT
3C- 20 E2 C9     0590            JSR CRLFBAS     ; CRLF
3F- A2 00        0600            LDX #$00
41- 86 BA        0610            STX *ZWSP1
43- A2 1B        0620 NEWLINE    LDX #$1B        ; ESC"K"+40+1
45- A9 4B        0630            LDA #$4B        ; GRAFICMODE
47- 20 84 E7     0640            JSR WRTWO       ; 320 PUNKTE
4A- A2 40        0650            LDX #$40        ; = 40 CBM ZEICHEN
4C- A9 01        0660            LDA #$01
4E- 20 84 E7     0670            JSR WRTWO
51- A2 00        0680            LDX #$00
```

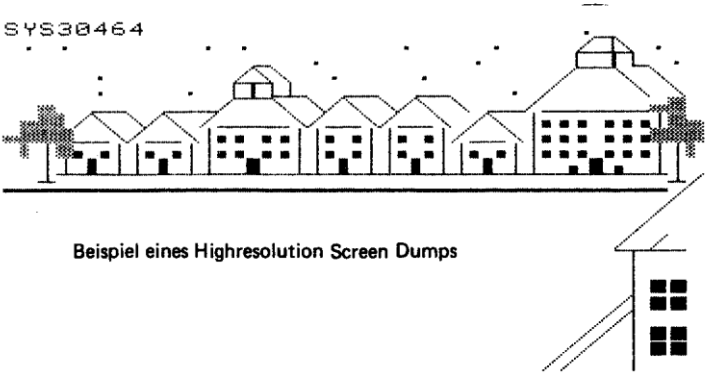
```
53- 86 BC      0690              STX *ZWSP2
55- A9 80      0700 NEXTCHR      LDA #$80          ; BILDSCHIRSTELLE
57- 85 22      0710              STA *ZEIGER+1     ; ERRECHNEN
59- A9 00      0720              LDA #$00          ; ( 32768+40*ZEILE
5B- 85 21      0730              STA *ZEIGER       ; +SPALTE )
5D- A6 BA      0740              LDX *ZWSP1
5F- F0 10      0750              BEQ SPALTE
61- 18         0760 ZEILE        CLC
62- A9 28      0770              LDA #$28
64- 65 21      0780              ADC *ZEIGER
66- 85 21      0790              STA *ZEIGER
68- A9 00      0800              LDA #$00
6A- 65 22      0810              ADC *ZEIGER+1
6C- 85 22      0820              STA *ZEIGER+1
6E- CA         0830              DEX
6F- D0 F0      0840              BNE ZEILE
71- 18         0850 SPALTE       CLC
72- A5 BC      0860              LDA *ZWSP2
74- 65 21      0870              ADC *ZEIGER
76- 85 21      0880              STA *ZEIGER
78- A9 00      0890              LDA #$00
7A- A8         0895              TAY               ; Y=0
7B- AA         0896              TAX               ; X=0
7C- 65 22      0900              ADC *ZEIGER+1
7E- 85 22      0910              STA *ZEIGER+1
80- B1 21      0930              LDA (ZEIGER),Y    ; ZEICHEN VON
82- 85 BE      0940              STA *ZWSP4        ; SCHIRMSTELLE LESEN
84- 86 BD      0960              STX *ZWSP3
86- A5 BE      0970 NEXTBYTE     LDA *ZWSP4
88- 10 04      0980              BPL NORVS         ; TEST OB RVS
8A- A2 FF      0990              LDX #$FF
8C- 29 7F      1000              AND #$7F
8E- 86 BF      1010 NORVS        STX *ZWSP5
90- 85 1F      1020              STA *STRADR
92- A9 00      1030              LDA #$00
94- 85 20      1040              STA *STRADR+1     ; ANFANGSADRESSE
96- 06 1F      1050              ASL *STRADR       ; DES ZEICHENS
98- 26 20      1060              ROL *STRADR+1     ; IM VIDEOROM
9A- 06 1F      1070              ASL *STRADR       ; BERECHNEN
9C- 26 20      1080              ROL *STRADR+1 ;
9E- 06 1F      1090              ASL *STRADR       ;(= BILDSCHIRMCODE * 8
A0- 26 20      1100              ROL *STRADR+1     ; + ANFANG ROMKOPIE)
A2- A5 20      1110              LDA *STRADR+1
A4- 09 78      1120              ORA #H,CHRROM
A6- 85 20      1130              STA *STRADR+1
A8- AD 4C E8   1140              LDA PCR           ; WENN TEXTMODE
AB- 29 02      1150              AND #$02          ; OBERES KB
AD- F0 07      1160              BEQ GRAFICMODE
AF- A5 20      1170              LDA *STRADR+1
B1- 18         1180              CLC
B2- 69 04      1190              ADC #$04
B4- 85 20      1200              STA *STRADR+1
B6- A4 BD      1210 GRAFICMODE   LDY *ZWSP3
B8- B1 1F      1220              LDA (STRADR),Y    ; INVERTIEREN
BA- 45 BF      1230              EOR *ZWSP5        ; WENN ZEICHEN RVS
BC- 20 D2 FF   1240              JSR BSOUT
BF- E6 BD      1250              INC *ZWSP3
C1- A5 BD      1260              LDA *ZWSP3
```

```
C3- C9 08      1270              CMP #$08       ; ZEICHEN FERTIG ?
C5- D0 BF      1280              BNE NEXTBYTE   ; NEIN
C7- E6 BC      1290              INC *ZWSP2
C9- A5 BC      1300              LDA *ZWSP2     ; 40 ZEICHEN FERTIG ?
CB- C9 28      1310              CMP #$28       ; CMP #$50  FUER CBM 8000
CD- F0 03      1320              BEQ NEXTLINE
CF- 4C 55 77   1330              JMP NEXTCHR    ; NEIN !
D2- 20 E2 C9   1340 NEXTLINE     JSR CRLFBAS
D5- E6 BA      1350              INC *ZWSP1
D7- A5 BA      1360              LDA *ZWSP1
D9- C9 19      1370              CMP #$19       ; 25. ZEILE FERTIG ?
DB- F0 03      1380              BEQ EXIT       ; JA -> EXIT
DD- 4C 43 77   1390              JMP NEWLINE    ; NEIN ->
E0- A2 1B      1400 EXIT         LDX #$1B       ; ESC 2
E2- A9 32      1410              LDA #$32       ; LF = 1/6 INCH
E4- 20 84 E7   1420              JSR WRTWO
               1430
E7- 20 CC FF   1440              JSR CLRCH      ; STANDART I/O SETZEN
EA- A0 7F      1450              LDY #127       ; CLOSE 127
EC- 84 D2      1460              STY *LA
EE- 4C AC F2   1470              JMP CLOSEFILE  ; -> BASIC
               1480              .EN
```

```
BSOUT =FFD2              CHRROM =7800            CKOUT =FFC9
CLOSEFILE =F2AC          CLRCH =FFCC             CRLFBAS =C9E2
EXIT =77E0               FA =00D4                FNLEN =00D1
GRAFICMODE =77B6         LA =00D2                NEW =C55D
NEWLINE =7743            NEXTBYTE =7786          NEXTCHR =7755
NEXTLINE =77D2           NORVS =778E             OPENFILE =F524
PCR =E84C                SA =00D3                SAVE =7700
SPALTE =7771             START =771A             STRADR =001F
WRTWO =E784              ZEIGER =0021            ZEILE =7761
ZWSP1 =00BA              ZWSP2 =00BC             ZWSP3 =00BD
ZWSP4 =00BE              ZWSP5 =00BF
//0000,77F1,77F1
```

```
               0010 ;      ROM-DREH
               0020 ;
               0030 ; VIDEO-ROM MUSS SICH
               0040 ; IM SOCKEL UD4 BEFINDEN
               0050 ; = BEREICH $A000 - $A7FF
               0060 ; 'GEDREHTES' ROM WIRD
               0070 ; VON $7800 - $7FFF
               0080 ; ANGELEGT
               0090 ;
               0100              .BA $033A
               0110              .MC $033A
               0120              .OS
               0130 ;
               0140 ROM          .DE $A000
               0150 COPY         .DE $7800
               0160 COPYZ        .DE $BA
               0170 ROMZ         .DE COPYZ+2
               0180 BUFF         .DE $03F0
               0190 ;
033A- A9 A0    0200 START        LDA #H,ROM     ; ZEIGER IN ZERO-
033C- 85 BD    0210              STA *ROMZ+1    ; PAGE INITIALISIEREN
033E- A9 78    0220              LDA #H,COPY
0340- 85 BB    0230              STA *COPYZ+1
```

```
0342- A9 00     0240            LDA #L,ROM
0344- 85 BC     0250            STA *ROMZ
0346- 85 BA     0260            STA *COPYZ
                0270 ;
0348- A0 00     0280 LOOP       LDY #$00
034A- B1 BC     0290 LOOP2      LDA (ROMZ),Y    ; ORGINAL-ZEICH
034C- A2 00     0300            LDX #$00        ; LESEN (8 BYTE)
034E- 6A        0310 LOOP1      ROR A           ; DREHEN
034F- 3E F0 03  0320            ROL BUFF,X      ; IN BUFFER-SPEICHER
0352- E8        0330            INX             ; BRINGEN
0353- E0 08     0340            CPX #$08
0355- D0 F7     0350            BNE LOOP1       ; ALLE BITS ?
0357- C8        0360            INY
0358- C0 08     0370            CPY #$08
035A- D0 EE     0380            BNE LOOP2       ; ALLE BYTES ?
                0390 ;
035C- A2 07     0400            LDX #$07
035E- A0 00     0410            LDY #$00
0360- BD F0 03  0420 STORE      LDA BUFF,X      ; BUFFER IN
0363- 91 BA     0430            STA (COPYZ),Y   ; COPY-BEREICH SCHIEBEN
0365- CA        0440            DEX             ; (8 SP.-STELLEN)
0366- C8        0450            INY
0367- C0 08     0460            CPY #$08
0369- D0 F5     0470            BNE STORE
                0480 ;
036B- 18        0490 MODZEIG    CLC
036C- A5 BC     0500            LDA *ROMZ       ; ROMZEIGER
036E- 69 08     0510            ADC #$08        ; & COPYZEIGER
0370- 85 BC     0520            STA *ROMZ       ; JEWEILS 8
0372- A5 BD     0530            LDA *ROMZ+1     ; ADDIEREN
0374- 69 00     0540            ADC #00
0376- 85 BD     0550            STA *ROMZ+1
0378- 18        0560            CLC
0379- A5 BA     0570            LDA *COPYZ
037B- 69 08     0580            ADC #$08
037D- 85 BA     0590            STA *COPYZ
037F- A5 BB     0600            LDA *COPYZ+1
0381- 69 00     0610            ADC #00
0383- 85 BB     0620            STA *COPYZ+1
0385- C9 80     0630            CMP #$80        ; LETZTES ZEICHEN ?
0387- D0 BF     0640            BNE LOOP
0389- 60        0650            RTS
```



Beispiel eines Highresolution Screen Dumps

FORCE SYS68K/CPU-1

16 Bit VME-Bus-System mit MC 68000

Beim Herausgeber wurde vor einiger Zeit der o.a. Einplatinen-Computer in Betrieb genommen. Hier ein erster Erfahrungsbericht. Vorweg aber der Hinweis auf weitere Artikel in diesem Heft, die dem Verständnis dienen können: Wichtige Merkmale der CPU 68000 und des Interfacebausteins PI/T.

Hardware

Die Platine hat doppeltes Europaformat mit 96-poliger Messerleiste P1 gemäß VME-Bus Spezifikation, auf die die mit LS645 gepufferten CPU-Signale herausgeführt sind. Eine zweite 64-polige Messerleiste P2 führt die Signale des Interfacebausteines 68230 (Standardausrüstung) oder der wahlweisen PIA 6821 heraus. Die Frontplatte enthält Taster für Reset und Abort, eine HALT-LED sowie drei serielle Ports (RS232) mit jeweils 25-poliger Buchse. An diese Schnittstellen kann ein Terminal, ein Host-Rechner und ggfs. ein zweites Terminal (Remote) angeschlossen werden, das der Anwender aber selber einprogrammieren muß.

Jede dieser Schnittstellen wird von einer eigenen ACIA 6850 betrieben. Die Bauraten sind bei Lieferung auf 9600 eingestellt. Durch anderes Jumpern können aber individuell Baudraten zwischen 110 und 38400 eingestellt werden. Am Terminal-Port wird hier das in der Zeitschrift MC 1/ und 2/83 beschriebene Terminal nach Regge et alia betrieben. Zu Terminals später noch mehr. Der Verbund mit einem Host-Rechner wurde hier noch nicht erprobt. Das Datenübertragungsformat (S-Records) ist jedoch beschrieben, so daß es in der Anfangsphase empfehlenswert sein mag, z.B. die Dienste des AIM 65 für eine Massenspeicherung in Anspruch zu nehmen, denn im Gegensatz zum Profi Kit 2 der Lieferfirma hat diese Platine von Haus aus kein Interface für Cassetten.

Die CPU ist eine 8 MHz-Version vom Zweitlieferanten Hitachi. On board sind vier Steckplätze für Festwertspeicher (EPROMs), die je nach Typ bis zu 64 KB System- oder Anwenderprogramm ansprechbar machen. Die Grundversion hat ferner 128 KB DRAM, nachrüstbar mit anderen Typen (256 k*1) und Decoder-Prom auf 512 K on board. Der Adreßraum ab $100000 ist für den VME-Bus, also für externe Einheiten vorgesehen.

In der Grundversion hat man zusammenhängendes RAM ab Adresse hex 8 (darunter liegt ein Prom mit dem System-Stackpointer für den Reset sowie der Reset-Vektor) bis 1FFFF. Das System belegt den Teil bis FFF. Der Bereich bis 3FF wird bei der Inbetriebnahme mit exception vectors gefüllt, die das Monitorprogramm verwertet und auch mit Auffangvektoren für die noch nicht benutzten Interrups. Ab 80008 liegt das Monitorprogramm und ab A0000 der optionale Editor/Assembler oder das auch verfügbare BASIC.

Zur Hardware gehört eine batteriegepufferte Echtzeituhr (1/1000 Sek. bis Monate), die von Anwenderprogrammen in das System einbezogen werden kann.

Die Platine ist in einer robusten industriellen Qualität gefertigt. Die hiesige hat auf der Unterseite (nur) fünf Jumperdrähte und einen Ableichkondensator. Für den Integrationsgrad sind das nur kleine Korrekturen. Die erforderlichen TTLs sind fest eingelötet, CPU, Speicher und Interfacebausteine sind gesockelt. Das Anwenderhandbuch (User's Manual) enthält für alle Teile einen verständlichen Schaltplan.

Einen Computer im Werte von rund DM 4000 sollte man von Anfang an in einen Kartenträger stecken, um alle Risiken, die mit einem fliegenden Aufbau verbunden sind, auszuschließen. Immerhin wird das System mit 3 Spannungen betrieben (+5V/ca. 1,6 A, +12V und -12 V, je 0,2A). Der Autor wählte von BICC-VERO den Kartenträger KM6 mit Bestellnummer 173-26518K, der für den VME-Bus geschaffen wurde und der einfache und doppelte Europakarten aufnimmt. Die Aufnahmemöglichkeit für Einfach-Europakarten ist zu empfehlen, wenn man den Interfacebus des PI/T in verschiedener Form herausführen will. Natürlich kann man das auch mit einem Flachkabel-

verbinder machen. Man benötigt anfangs jedoch nicht unbedingt eine VME-Bus Mutterplatine, die wegen der hochgesteckten Spezifikationen dieses Busses oberhalb DM 600 kostet. Es kommt zunächst nur auf eine 96-polige VG-Buchse an, die man fest im Kartenträger verankert und auf die man in einer betriebssicheren Form die Spannungen drahtet.

Zu den auf der Platine enthaltenen Interfacebausteinen ist zu bemerken, daß keiner von ihnen zum Interrupt zugelassen ist. Zwar sind die Ausgangssignale des PI/T, nämlich PIRQ und TOUT, gemeinsam (logisch ODER) auf Level 5 des CPU-Interrupts geführt, da aber kein dekodiertes IACK auf PIACK oder TIACK zugrückgeführt ist, landet die CPU nach dem TIMEOUT auf dem unechten Interrupt (spurious interrupt). Bei Bedarf kann man die Beschaltung natürlich entsprechend abändern.

Der System-Monitor

Das Monitorprogramm sitzt in zwei 28-poligen EPROMs 2764 (für Lower und Upper Data Strobe). Zu ihm gehört ein etwa 70-seitiges Anwenderhandbuch, das für alle Kommandos Beispiele enthält. Wer von einem kleineren Computer her kommt, für den er manche Monitordienstleistung selber zusätzlich hat schreiben müssen, wird angenehm überrascht sein, daß er hier fast alles Wünschenswerte von Anfang an findet. Mit dem FORCE-Monitor freundet man sich schnell an und hat seine Möglichkeiten in mehrstündiger Sitzung spielend erprobt. Dabei macht man dann auch Versuche, die CPU vorsätzlich zum Abstürzen zu bringen. Wegen der vom Monitor initialisierten Interrupt-Vektoren (exceptions) ist das gar nicht so leicht. Meistens meldet sich die CPU hantierbar mit einer Fehlermeldung zurück. Nun ein Überblick über die Dienstleistungen des Monitors:

Block Move - Bereich verschieben
Block Fill - Bereich mit Zeichen füllen
Block Search - Bereich auf Hexwerte oder ASCII-String absuchen
Block Test of Memory - Speicherbereich physikalisch prüfen, RAM-Test
Breakpoints anzeigen, setzen, löschen
Zahlenumrechnung hex-dez-hex
Registersatz anzeigen, Register ändern, auch die im RAM geführten Anwenderregister
Memory anzeigen, hex, zugleich mit ASCII-Ausdeutung, Zeilen mit 16 Bytes
Option: Memory disassemblieren für Bereich
Memory verändern. Option: Mit Eingabe neuer Befehle in Assembler-Schreibweise
(Line by line Assembler ohne Abspeicherung des Textes, keine Label)
Memory Set - Data oder ASCII-Strings einschreiben
GO - Programmausführung mit oder ohne Berücksichtigung von Breakpoints.
Es kann zu jedem Breakpoint auch eine Zahl angegeben werden, die angibt, wie oft ein Breakpoint zunächst ohne Anhalten überlaufen werden soll
HELP - Anzeige der zur Verfügung stehenden Monitor-Kommandos
Lade/Speichere/Verifiziere Object Files
Printer EIN/AUS - Ein Printer mit Centronix-Interface ist unterstützt (am PI/T)
OFFSET - bewirkt die Ausführung von Kommandos an spezifizierter Adresse+Offset
Änderung des Übertragungsformates an den seriellen Schnittstellen
TRACE, TRACE bis Breakpoint - Einzelschrittbearbeitung
Transparent Mode - Direkte Durchleitung des Terminals an den Host-Rechner

Die vorgenannten leistungsfähigen Kommandos können oft mit verschiedenen Optionen ausgeführt werden.

Der Screen Editor

Als Option zum besprochenen Computer und auch zum Profi Kit 2 kann man den Editor/Assembler FORCE IDEAL 2.0 in Form von zwei EPROMs beziehen. Für jeden Betreiber, der häufig Programmentwicklungen betreibt oder der viel mit anderen Textenarbeiten will, ist diese preiswerte Anschaffung (ca. DM 300) sehr zu empfehlen.

Der Editor wird 'Screen Oriented Editor' genannt. Und das ist er wirklich: Wie bei Commodore, wo das Bildschirmrefresh-RAM Teil des allgemeinen RAM ist, gilt auch bei FORCE 'what you see

ist what you get'. Nur daß hier alles über die serielle Verbindung läuft. Das Editor-RAM des Computers folgt per Programm den Cursorbewegungen des unabhängigen Terminal-Bildspeichers.

Ein Bildschirminhalt von im allgemeinen 24 Zeilen zu 80 Zeichen stellt einen 'screen' dar, ein Bildfenster, das mit Befehlen über den gesamten vorhandenen Text geschoben werden kann, auch abschnittsweise. Das jeweilige Bildfenster kann mit dem Edit-Befehl der Nachbearbeitung unterworfen werden. Dabei wird nicht zeilenweise mit CR übernommen, sondern man kann mit dem Cursor überall in das Bild fahren und Veränderungen vornehmen, auch mit Insert/Delete, Clear to End of Line, Insert New Line. Der Text im Computer wird zugleich für den ganzen screen verändert. Mit CR verläßt man das Bildfenster.

Neben diesen rangierenden und korrigierenden Befehlen finden wir eine ein- und ausschaltbare Numerierung, Copy (Zeile oder Block), Delete (ebenso), Move (umordnen, ebenso), Print auf den Drucker, List (verbunden auch mit dem Find-Befehl), Replace (String gegen String), Save, Assembler und GO (assembliertes Programm ausführen). Die Übergänge zwischen Editor und Assembler sind arbeitssparend, zumal der Editor fast immer sinnvolle Anschlußkommandos vorschlägt.

Es muß noch ein Wort zu Terminals gesagt werden: Der Editor ist auf die Steuerzeichen eines VT52/FACIT 4420 -kompatiblen Terminals programmiert, das für die Steuerung z.T. Sequenzen von mehreren Bytes abgibt. Das ist jedoch keine Begrenzung (hier läuft das preiswerte MC-Terminal), denn die Dokumentation enthält genaue Hinweise, welche Parameter benutzt werden und wo sie gespeichert sind. Es ist dem Anwender damit möglich, das voll verschiebliche Editorprogramm per Block Move ins RAM zu übertragen und dort die Steuerzeichen seines Terminals abzugleichen und zu erproben, ehe er die Korrekturen in ein EPROM schießt.

Der Assembler

Angesichts des großen RAM-Speichers lag es nahe, für eine schnelle und einfache Hantierung sowohl den Quelltext, als auch den erzeugten Maschinencode auf der Maschine resident zu halten, ohne daß externe Medien herangezogen werden müssen. Nach den Angaben verarbeitet der Assembler 7-12000 Statements per Minute.

Die Syntax und die Mnemonics weichen nur in wenigen Punkten von Motorola ab. Bemerkenswert ist allerdings die formatfreie Eingabe des Quelltextes: Eine Zeile kann mehrere Statements gemischt auch mit Kommentaren enthalten, sofern die damit notwendigen Begrenzerzeichen dieses Assemblers beachtet werden. Hinzu kommt die Möglichkeit, nicht geschachtelte Blockstrukturen zu benutzen, die mit lokalen Symbolen arbeiten, die also außerhalb des Blockes nicht zu Beanstandungen führen, wenn die Namen dort wieder benutzt werden. Symbole können beliebig lang sein, signifikant sind allerdings nur die ersten 14 Zeichen. Der Assembler ist in der Lage, mit Symbolen und Ausdrücken in den vier Grundrechenarten zu rechnen.

Für die Adressierungsarten sind alle Möglichkeiten der 68000 berücksichtigt, auch diejenigen relativ zum Programmzähler, die einen voll verschieblichen Code möglich machen.

Bei den Assembleranweisungen finden wir nicht nur ORG, DC (Konstante), DS (Speicherraum), EQU (Gleisetzung) und Paging, sondern auch verschiedene Listoptionen sowie SIZE (Festelegung der Operandengröße on default, die wirksam wird, wenn man in einem Befehl keine ausdrückliche Größenangabe formuliert). Das BASE gestattet die Angabe einer Berechnungsbasis, die ggfs. erst zur Assemblierungszeit berechnet wird.

Das Handbuch zum Editor/Assembler enthält zahlreiche Beispiele, wie man in einer gut dokumentierenden symbolischen Art formulieren kann. Bemerkenswert ist, daß auch die Maschinenregister sogar mit Angabe der auf sie anzuwendenden Adressierungsart in die Symbolsprache einbezogen werden können, so daß der Programmierer sich nach den anfänglichen Definitionen voll auf den Lösungsweg konzentrieren kann, ohne immer die Adressierungsarten und Register wiederholen zu müssen. Beispiel: ADD COLUMN,INDEX wird vom Assembler aufgelöst zu ADD.W (A4),D6, wenn ihm die Symbole zuvor so erklärt waren.

Dokumentation

Mit der Lieferung erhält man das User's Manual, ein Hardware-Handbuch von ca. 120 Seiten Text nebst Schaltplänen plus den Datenblättern für 68000 (CPU), 68230 (PI/T), 6850 (ACIA) und 58167A (RTC). Die hierin enthaltene umfangreiche Beschreibung der Signale und ihrer Ablauflogik, der Jumpermöglichkeiten, des VME-Bus dürfte ausreichen, um das System zu erweitern, zu verändern oder bei Störungen zu diagnostizieren.

Für die Hantierung des Monitorprogramms enthält ein weiteres 67-seitiges Handbuch zahlreiche Beispiele. Es enthält ferner die Einsprungspunkte von 15 wichtigen Systemroutinen zusammen mit der Angabe ihrer Leistung, besonders für die E/A. Wenn auch ein Monitor-Listing noch wünschenswerter wäre: Mit den hervorragenden Diagnosemöglichkeiten kann man auch selber (mühsamer) den Sitz von Systemparametern und Abläufen durchleuchten. Für den Anfang scheint wesentlich, daß alle wichtigen Dienstleistungen zur Verfügung stehen, die Monitor, Editor und Assembler bieten.

Zusammenfassung

Das besprochene FORCE-System ist erst seit der Jahreswende am Markt und wird auch hier erst kurz betrieben. Seine wesentlichen Merkmale sind: Professioneller Aufbau, großer Speicher, leicht hantierbare durchdachte Systemprogramme und viel Dokumentation für die Hardware und Hantierung. Es ist erweiterbar mit VME-Bus Modulen. Es macht viel Spaß, sich seiner als Entwicklungssystem und für die Einarbeitung in die Möglichkeiten des 68000 zu bedienen.

R.L. □□

Dipl.-Ing. Wolfgang Seer, 1000 Berlin 26

PETAL

Ein Precompiler für die CBM-Serie

Allgemeines

Precompiler (Vorübersetzer) sind in der Datenverarbeitung nichts neues. Der bekannteste unter ihnen ist der FORTRAN-Precompiler RATFOR. Wenn auch einige wenige nur auf 'kosmetische Punkte' ausgerichtet sind, stellten die meisten von ihnen verbesserte Kontrollfluß-Strukturen bereit, in dem sie so die Unzulänglichkeiten der Programmiersprache hinsichtlich ihrer Strukturierungsmöglichkeiten ausgleichen.

Der BASIC-Precompiler PETAL verbessert beides. Die zusätzlichen Konstrukte werden als Pseudobefehle wie 'do until - enduntil' usw. dargestellt. Im Gegensatz zum normalen Compiler werden die Konstruktbefehle hier auf der Grundlage der in der Programmiersprache vorhandenen Schlüsselwörter (Keywords) gebildet. Für PETAL bedeutet dies, daß der Precompiler CBM-BASIC erzeugt!. Er ist selbst ebenfalls vollständig in BASIC geschrieben (1-Pass-Übersetzer), liest das zu übersetzende Programm zeilenweise von der Diskette (GET-Befehl), inspiziert dieses und schreibt die Zeile (u.U. verändert) auf die angegebene Diskette zurück.

Das hat den Vorteil, daß der Precompiler relativ einfach den individuellen Bedürfnissen angepaßt werden kann. Im Handbuch sind eine Reihe von Beispielen angegeben. Man denke da nur an die Namenskonvention zwischen Sourcefile und RUN-File. Andererseits ist die Übersetzungsgeschwindigkeit wegen des GET-Befehles nicht sehr groß. Es bleibt dem Anwender aber unbenommen, mit einem geeigneten Compiler den Precompiler zu übersetzen, um so die Verarbeitungsgeschwindigkeit zu erhöhen.

PETAL-Konstrukte

Petal besitzt alle Konstrukte, die auch in PASCAL üblich sind und geht mit seinem EXIT noch darüber hinaus:

DO UNTIL	- ENDUNTIL
DO WHILE	- ENDWHILE
CASE	- ENDCASE
IF-THEN-ELSE	- ENDIF
DO EXIT	- ENDEXIT

Syntax-Format	Beispiel	Uebersetzung
DO UNTIL Bedingung Block ENDUNTIL	100 do until x=n 110 : x=x+1 120 enduntil	100 x=x+1 110 if(x=n)=0goto100 120 ...
DO WHILE Bedingung Block ENDWHILE	100 do while x>n 110 : x=x-1 120 endwhile	100 if(x>n)=0goto130 110 x=x-1 120 goto100 130 ...

CASE
  <<Bedingung 1>> Block 1
  . .
  . .
  <<Bedingung n>> Block n
ENDCASE
	100 case
	110 : <<a=3>> print d
	120 : <<x>c>> print f
	130 endcase
		100 if(a=3)=0goto130
		110 printd
		120 goto150
		130 if(x>c)=0goto150
		140 printf
		150 ...

IF Bedingung THEN
  Block 1
ELSE
  Block 2
ENDIF
	100 if a>7 then
	110 : x=1
	120 else
	130 : x=0
	140 endif
		100 if(a>7)=0goto130
		110 x=1
		120 goto140
		130 x=0
		140 ...

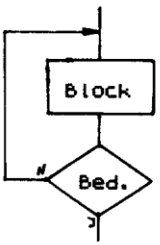
DO EXIT
  Block 1
IF Bedingung THEN EXIT
  Block 2
ENDEXIT
	100 do exit
	110 : a=1
	120 if k=n then exit
	130 : k=k+1
	140 endexit
		100 a=1
		110 if(k=n)goto140
		120 k=k+1
		130 goto100
		140 ...

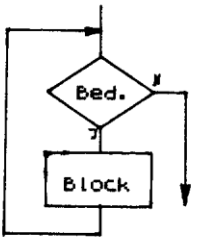
Die in BASIC vorhandene FOR-NEXT-Schleife steht natürlich weiterhin zu Verfügung. Tabelle 1 zeigt eine Gegenüberstellung der PETAL-Konstrukte in verschiedenen Darstellungsarten, während Bild 1 die Konstrukte in der Form eines Programmablaufes darstellt. Die in der Übersetzung enthaltene IF (Bedingung)=0 THEN ... sieht etwas fremdartig aus, hat aber ihren Sinn darin, daß im CBM-BASIC die boolschen Werte für TRUE und FALSE intern als –1 bzw. 0 dargestellt werden. In BASIC wird üblicherweise die IF–THEN-Anweisung immer dann ausgeführt, wenn die Bedingung <>0 ist, also nicht nur bei TRUE= –1!. Das bedeutet, daß man nicht ohne weiteres IF NOT (Bedingung) THEN ... bilden kann, wenn man eine Bedingung negieren möchte, da in diesem Fall das Zweier-Komplement einer 16-Bit-Dualzahl gebildet wird. Die Werte –1=TRUE und 0=FALSE stellen einen Sonderfall dar. Die in PETAL angewandte Art der Negation ist jedoch allgemein anwendbar.

Allein die bloße Anwendung der Konstruktbefehle gibt dem Programm eine Blockstruktur. Der BASIC-Befehl GOTO wird überflüssig und seine Anwendung wird wie bei PASCAL in den Hintergrund gedrängt (ein Verbot von GOTO ist in PETAL möglich). - Andererseits begünstigen die angebotenen Strukturierungsmöglichkeiten einen Programmentwurf vor der Codierungsphase. 'Rucksackprogrammierung' nach der Methode der intuitiven Eingebung beim Codieren kann auch durch PETAL nicht ausgeschlossen werden, wird aber erschwert und im Listing für jedermann offenkundig.

PETAL ist von der in BASIC üblichen Zeilennumerierung unabhängig, denn Blöcke (Unterprogramme) werden mit ihrem Namen (Label) 'aufgerufen' wie z.B. GOTO Ende, GOSUB Dialog usw.. Die Zeilennummer hat somit nur noch eine ordnende Funktion. Es besteht i.a. auch keine Übereinstimmung zwischen Quell- und übersetzter Programmzeilen-Nummer.

Nach E. W. Dijkstra kann jedes Programmierproblem auf jeder Zerlegungsphase mit einem der drei grundlegenden Strukturelemente gelöst werden, nämlich





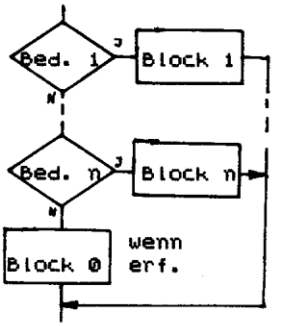
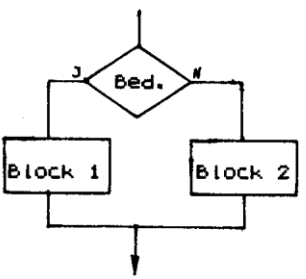


Bild 1a.DO UNTIL 1b.DO WHILE 1c.CASE



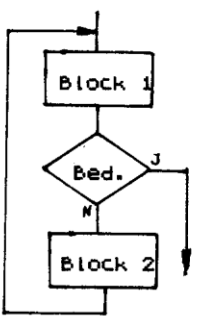


Bild 1d. IF THEN ELSE 1e. DO EXIT

Bild 1 Konstrukte als Programmablauf

Aneinanderreihung	(Sequenz)
Wiederholung	(Repetition)
Auswahl	(Selektion).

Dadurch kann die Problemlösung beispielsweise mit einem Baumdiagramm nach der Top-down-Methode sprachunabhängig, problemadäquat, einfach und schnell und bis zum gewünschten Detaillierungsgrad beschrieben werden. In Bild 2 sind die drei grundlegenden Strukturelemente als Blöcke dargestellt.

Dokumentation im Quellprogramm

PETAL unterstützt die Dokumentation innerhalb des Quellprogrammes. Bei der Übersetzung werden alle überflüssigen Leerzeichen (Spaces) im Coding entfernt. Zu dem REMARK-Befehl in BASIC kommen noch besondere Kommentarklammern ([ ]), die nicht nur am Ende, sondern innerhalb einer Zeile, ja sogar innerhalb eines Befehls (Variablennamen) eingesetzt werden können. Rückt man eine Zeile mit einem Doppelpunt (:) und Leerzeichen ein, so wird dieser mit den Leerzeichen bei der Übersetzung entfernt. Wenn nötig, können auf diese Art Leerzeilen erzeugt werden, die nicht in das übersetzte Programm eingehen. - Während normalerweise Maßnahmen zur Verbesserung der Lesbarkeit eines Programmes in BASIC allgemein eine Verschlechterung der Effektivität in der Programmausführung ergeben, d.h. der Speicherplatz nimmt zu und die Geschwindigkeit sinkt, ist es bei PETAL umgekehrt: Das Coding des übersetzten Programmes ist sehr kom-

pakt. Programme, die vom Autor nachträglich übersetzt wurden, ergaben Speicherplatzeinsparungen bis zu 40% und die Bildschirmmasken wurden merklich schneller ausgegeben.

Nutzt der Programmierer die Möglichkeiten hinsichtlich einer problemadäquaten Codierung gut aus, so ergeben sich Konstrukte mit Variablen und Konstanten, die in hohem Maße selbstdokumentierend sind, wie

1. DO UNTIL DATEI=EOF ... ENDUNTIL
2. DO EXIT ... IF EOF THEN EXIT ... ENDEXIT
3. DO WHILE DATEI=NOT EOF ... ENDWHILE

Unterprogramme - Allgemeines

Nicht ganz so trivial, wie es vielleicht den ersten Anschein hat, ist das Umwandeln der Labelnamen in Zeilennummern, wie BASIC es verlangt, sofern man die bei BASIC vorhandenen Freiheiten bezüglich der Lage von Unterprogrammen und ihren Aufruf innerhalb eines Programmes nicht einschränken will. So muß z.B. bei der Übersetzung für den Labelnamen eines GOSUB-Befehls sofort eine Zeilennummer eingesetzt werden (1-Pass-Übersetzer), obwohl die Subroutine erst weiter hinten im Coding auftritt. Dies führt zwangsläufig dazu, daß sich der Anwender zum Zeitpunkt der Übersetzung über die Numerierungsstruktur des übersetzten Programms (RUN-File) Gedanken machen sollte. Der Precompiler erwartet den oder die Namen der in die Übersetzung einzubeziehenden Programme, das Laufwerk und die Numerierungsparameter Startnummer, Schrittweite sowie Segmentierung. Die Default-Parameter 100,10,1000 erleichtern die Eingabe und decken viele Anwendungsfälle ab. Ergeben sich Kollisionen in den Zeilennummern der Blöcke, so wird der Segmentabstand vom Precompiler neu gesetzt, und die Übersetzung wird von Beginn an wiederholt. Von dieser Philosophie her erscheint es nur logisch, Unterprogramme nicht nur innerhalb des Quellprogrammes zu haben, sondern sie auch in Form einer Unterprogramm-Bilbliothek auf Diskette zuzulassen, die während der Übersetzung eingefügt werden.

Implizite Unterprogramme

Als implizite Unterprogramme werden alle 'normalen' Unterprogramme (Subroutinen) innerhalb eines Programmsegments (z.B. Hauptprogramm) bezeichnet. Sie sind durch das PETAL-Schlüsselwort SUB gekennzeichnet (z.B. SUB lies-datensatz). Im Gegensatz zur expliziten Deklarierung wird während des Übersetzens Labelname und aktuelle Zeilennummer in eine interne Tabelle eingetragen. Bei einem GOSUB labelname wird dann in der Tabelle 'nachgesehen' und die eingetragene Zeilennummer anstelle des Labelnamen gesetzt. Hieraus folgt, daß implizite Unterprogramme im Quellprogramm physikalisch immer vor ihrem ersten Aufruf stehen müssen, weil ihre Zeilennummern sonst unbekannt sind. Siehe auch Bild 3.

Explizite Unterprogramme

Auf Diskette vorhandene (explizite) Unterprogramme werden zu Beginn des Übersetzens dem Precompiler mitgeteilt und von diesem in eine Liste (Tabelle) eingetragen. Aus dem vorher angegebenen Segmentparameter wird die Zeilenummer des Blockanfangs berechnet und der Tabelle zugefügt (Bild 4). Somit ist es möglich, für jeden Aufruf GOSUB label eine Zeilennummer einzusetzen, auch wenn das Programmsegment (Unterprogramm) physikalisch hinter dem Aufruf liegt. Dadurch ist es auch möglich, daß sich Unterprogramme gegenseitig aufrufen können.

Unterprogramme in Maschinensprache

Kleinere Maschinenprogramme werden in BASIC-Programme oft als DATA-Zeilen (Bytes als Dezimalwert abgelegt und mit dem POKE-Befehl an die Obergrenze des RAM gebracht. Eine solche

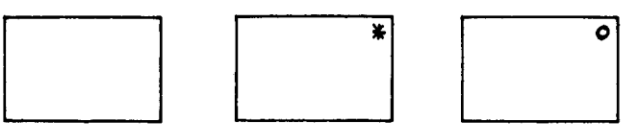


Bild 2 Die 3 grundlegenden Strukturelemente als Bloecke dargestellt

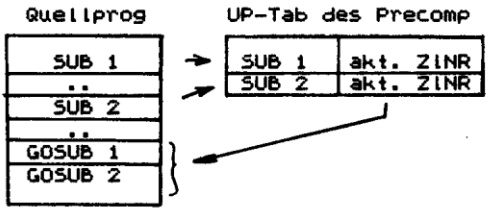


Bild 3 Implizite Unterprogramme stehen immer vor dem Aufruf

explizit deklariert:

*eingabe
*ausgabe
verarbeitung
*input

⇒

Tabelle	
Segment-Name	ZlNR
eingabe	100
ausgabe	1000
verarbeitung	2000
input	3000

*: BASIC-UP

Bild 4 Als Disk-Files explizit deklarierte Programmsegmente (Hauptprog ab ZlNR 2000)

Routine wird dann durch Ändern der BASIC-Pointer gegen Überschreiben geschützt. Speichert man aber die DATA-Zeilen als separates BASIC-Programmfile ab, so bietet PETAL die Möglichkeit, das oben geschilderte Handling automatisch durchzuführen. Der Programmierer braucht sich um die Ablage- und Aufrufadresse nicht zu kümmern. Der Aufruf wird mit 'SYS Name des Datafiles' angegeben. Voraussetzung ist aber, daß die Routine für sich verschieblich ist, damit sie von PETAL frei abgelegt werden kann, und daß der Beginn der Routine auch die Aufrufadresse darstellt. Das Ablegen der Routine an vorgegebener Stelle ist als Option ebenfalls möglich. Im RUN-File wird von PETAL außer dem zu verschiebenden Maschinenprogramm noch eine Lade- und Löschroutine zugefügt. Zu Beginn des ersten RUNs, unmittelbar nach dem Verschieben der eigentlichen Routine, werden alle drei (nicht mehr benötigten) Teile des BASIC-RUN-Files dynamisch gelöscht.

Das Binden von Programmen

Ist im Laufe der Zeit die PETAL-Programmbibliothek mit nützlichen BASIC-Routinen angewachsen, wird man immer häufiger darauf zurückgreifen, wenn neue PETAL-Programme erstellt werden. Um nun nicht bei jedem Übersetzungsversuch die Routinen explizit angeben zu müssen, bietet PETAL die Möglichkeit, alle Teilprogramme zu einem separaten Programm zusammenzufassen oder mit dem Hauptprogramm zu einem einzigen zu binden. Die Zeilenummern der einzelnen Programmteile sind - wie bisher - unerheblich. Binden heißt also, mehrere PETAL-Quellprogramme zu einem neuen PETAL-Quellprogramm zu vereinigen, um es danach zu übersetzen. Zu beachten ist dabei, daß alle in dieser Form gebundenen Unterprogramme für das Hauptprogramm implizit deklariert sind und damit vor diesem stehen müssen.

PETAL-Demoprogramm

Um nicht den Rahmen dieses Heftes durch eine direkte Gegenüberstellung eines BASIC- und PETAL-Programmes zu sprengen, wurde auf das BASIC-Programm PROGMENUE von R. Kirchgäßner (65xx MICRO MAG, Heft 19: Drei Disk Utilities) zurückgegriffen. Obgleich hierfür nicht alle PETAL-Konstrukte demonstriert werden konnten, zeigt sich PETAL in beeindruckender Weise. Wegen der Vergleichbarkeit beider Programme wurde die Maschinen-Routine INPUT (alle eingerahmten Zeilen) im PETAL-Programm belassen und auch so übersetzt. In der Praxis würden die DATA-Zeilen als BASIC-Programm gehalten und dem Precompiler als Maschinen-Unterprogramm (MUP) deklariert. Nach der Übersetzung hatte das (nicht abgebildete) RUN-File eine Länge von 1379 Bytes.

Schlußbemerkungen

Solange die Konstrukte, die PETAL bietet, nicht von Haus aus in BASIC integriert sind, muß PETAL als ausgezeichnetes Werkzeug der Programmentwicklung betrachtet werden. Mit PETAL erstellte Programme brauchen in der Regel keine oder nur wenig Dokumentation außerhalb des Codings. Sie werden schneller erstellt und beinhalten nach den Erfahrungen des Autors weniger logische Fehler. PETAL-Programme sind auch pflegeleichter als andere.

Ob allerdings all' diese Tugenden für den Hobbyprogrammierer relevant sind? Der Autor jedenfalls ist skeptisch. Wie gesagt, PETAL ist nur Werkzeug, und auch das PETAL-Programm kann nur so gut sein, wie der Entwurf, der ihm zugrundeliegt.

Literatur

Precompiler PETAL, v. Blücher, Hamburg 1982
Grundsätze des Programmentwurfs, M. Jackson, stmv, Darmstadt 1979.

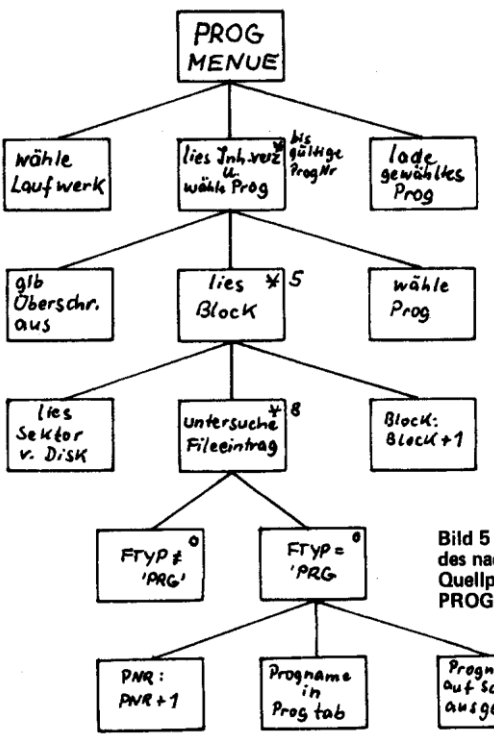


Bild 5: Blockstruktur des nachfolgenden Quellprogrammes PROG-MENUE

```
30 rem
40 rem " PROG-MENUE    13.11.82
50 rem " R. Kirchgaessner
60 rem " Nibelungenstr. 4
70 rem "  D 6831 Bruehl
80 rem
90 rem "    als PETAL-Programm
100 rem
110 rem "    von W. Seer
120 rem "   1000 Berlin 26
130 rem "     13.02.83
```

```
160 :
170 poke 53,126: clr
180 open 15,8,15
190 dim progtab$(144)
200 inp  =32786
210 prg  =130
220 pnr  =0
230 :
240 :
250 for x=inp to inp+61                    +" Lies DATA-Statements &
260 :   read s: poke x,s                   +" lege sie als Maschinen-Prog ab
270 next x
280 :
290 +   " ..........................
300 +   "      Waehle Laufwerk
310 +   " ..........................
320 :
330 do until d=8 or d=1
340 :  input"■■■Drive   8■■■■";d
350 enduntil
360 :
370 print#15,"i"d
380 open 1,8,2,"#"
390 :
400 +  " ..........................
410 +  "  Lies Inhaltsverzeichnis &
420 +  "      waehle Programm aus
430 +  " ..........................
440 :
450 do until wahl>0 and wahl<=pnr
460 :  print "■"tab(12)"■PROG-MENUE■■"
470 :  block=0
480 :
490 :+    " ..........................
500 :+    "       Lies Block aus
510 :+    "     Inhaltsverzeichnis
520 :+    " ..........................
530 :
540 :  do until block=5 or t$=""                        + " 8032 => block=9
550 :
560 :     if t$="" then t$=chr$(18): s$=chr$(1):pnr=0    + " 8050 => t$=chr$(39)
570 :     t=asc(t$): s=asc(s$)                          +" naechster Sektor
580 :     print#15,"u1:"j2;dj;t;s                       + "auf Sektor positionieren
590 :     get #1,t$: sys (inp)1,s$,1                    + "Link-Bytes

620 :+       " ..........................
630 :+       "    Untersuche File-Eintrag
640 :+       "         wenn PRG-File,
650 :+       "    dann Prog-Namen in Tabelle
660 :+       " ..........................
670 :
680 :     for f=1 to 8                                  + "8 Eintraege lesen
690 :         sys (inp)1,ftyp$,1
700 :         if asc(ftyp$)=prg then
710 :                         pnr=pnr+1
720 :                         sys (inp)1,x$,2
730 :                         sys (inp)1,progtab$(pnr),16
740 :                         sys (inp)1,x$,13
750 :                         if p=2 then p=0:      + "8032: if p=4...
760 :                         print tab(20*p);
770 :                         print "■"right$(str$(pnr),2)"■ "progtab$(pnr)" ";
780 :                         p=p+1
790 :         else
800 :                         sys (inp)1,x$,31      +  " kein PRG-File
810 :         endif
820 :     next f
830 :     block=block+1
840 :  enduntil   + block ...
850 :  print
860 :  print tab(20)"■Ihre Wahl >=0 ■";: input wahl
870 enduntil   + wahl ...
880 :
890 close 1
900 :
910 +  "  ..........................
920 +  "   Lade ausgewaehltes Programm
930 +  "  ..........................
940 :
950 print"■load"chr$(34)str$(d)":"progtab$(wahl)chr$(34)",8■■■■■run"
960 poke 623,19:poke 624,13:poke 625,13:poke 158,3
970 end
980 :
990 +  "  ..........................
1000 +  "        INPUT-Routine
1010 +  "  ..........................
1020 :
1030 data 32,120,214,32,198,255,32,248,205,32,109,207
1040 data 32,144,204,133,70,132,71,32,204,214,138,32
1050 data 79,211,160,0,32,228,255,166,150,208,18,145
1060 data 95,200,196,94,208,242,32,164,211,32,226,200
1070 data 76,204,255,169,0,145,95,200,196,94,208,247,240,236
ready.
```

Wolfgang Zweygart, 7030 Böblingen

Floppy-Controller für den AIM 65

Ein erster Test von Hard- und Software: Getestet wurden der Floppydisk-Controller mit der Bezeichnung RM 65-5101 NE und die dazugehörige Software, die sich AIM 65 DOS Version 1.0 nennt.

Schon lange besteht der Wunsch, den AIM mit einer Floppy zu betreiben, so daß auch schon einige Möglichkeiten dazu gefunden wurden. Durch die Benutzung eines schnellen Massenspeichers, wie dieses Floppydisk-Systems, wird der AIM in eine völlig neue Leistungskategorie angehoben. Beim Verfasser wurden alle ROMs, außer Monitor und DOS, durch RAM ersetzt, wodurch eine weitere Erhöhung der Flexibilität erreicht wurde. Die Initialisierung des DOS wurde in die Resetroutine des Monitors übernommen, so daß der AIM beim Einschalten sofort bereit ist. - Dies ist nun die Lösung von Rockwell:

Die Hardware

Die Platine macht einen mechanisch außerordentlich soliden Eindruck und wird in einer vierschichtigen Multilayertechnik gefertigt, wobei die beiden mittleren Schichten die Stromversorgung übernehmen. Als Controller findet ein FD 1793 Verwendung. Auch doppelseitige Laufwerke können verwendet werden. Die Hardware ist von 5,25 Zoll, einfache Dichte, bis 8 Zoll doppelte Dichte eingerichtet. Dieses wird durch steckbare Jumper erreicht, die man nur umdrehen muß. Als Datenseparator wird eine aufwendige PLL-Schaltung verwendet. Auf der Platine befindet sich auch eine Fassung für einen (EP)ROM. In der normalen Version ist das Modul für den Bereich 8000-8FFF dekodiert, aber durch ein anderes Adreßdekodier-PROM kann auch ein anderer Bereich gewählt werden. Auch die Software ist für diesen Bereich geschrieben. - Beim Verfasser läuft die Karte als einziges Modul aus der MIKROFLEX-Serie. Dies wurde nur dadurch möglich, daß ein Adapter eingebaut wurde, der den Adreß- und Datenbus invertiert.

Die Software

Genau wie die Hardware läßt auch das DOS alle Möglichkeiten offen. So ließ sich das Systen leicht an ein Laufwerk mit 80 Spuren anpassen, nämlich durch das Ändern einer einzigen Variablen im RAM-Bereich. Das DOS benutzt die Zellen D7-DD in der Zeropage, sowie ab 04A0 bis 07FF im Hauptadreßbereich. Dieser Bereich wird als Variablenbereich und als I/O-Puffer verwendet. Dieser Puffer läßt sich natürlich auch in einen anderen Bereich verschieben. - Die Kommunikation zwischen dem DOS und dem Benutzer geschieht durch die Verwendung der F1-Taste zum Ausdruck der Directory und der F2-Taste für ein Utility-Menue mit Dienstleistungen, die ein File löschen, auflisten, wiederherstellen oder die ganze Diskette kopieren. Der Datenaustausch zwischen AIM und Floppy wird über den User-Vektor abgewickelt. Jede Ein-/Ausgabe, die über die Unterprogramme WHEREI und INALL bzw. WHEREO und OUTALL läuft, kann auch zur Floppy gehen.

Das DOS besteht aus zwei Teilen: (1) dem Kern aus primitiven Diskroutinen, die eine Spur suchen, einzelne Sektoren beschreiben oder lesen, sowie ein Laufwerk initialisieren und Motorbefehlen und (2) aus einem spezifischen Überbau, der die Eröffnung von Files und alle anderen Dienstleistungen des DOS übernimmt und die Zusammenarbeit mit dem AIM steuert.

Wie bei Rockwell üblich, bekommt man auch ein umfassendes Handbuch dazu, in dessen Anhang sich ein kommentiertes Assemblerlisting für die primitiven Routinen des DOS befindet, so daß man hierauf auch eigene Software aufbauen kann. Außerdem wird im Manual die Zusammenarbeit mit den verschiedenen Sprachen beschrieben, die inzwischen für den AIM 65 zugänglich sind.

Bei einfacher Dichte wird im Format IBM 3740 und bei doppelter Dichte in dem des IBM System 34 gearbeitet.

Beim Verfasser wird zur Zeit daran gearbeitet, den Monitor des AIM abzuändern, damit auch ohne Benutzung des User-Vektors mit der Floppy gearbeitet werden kann.

Das Diskettensystem in der Praxis

Erprobt wurde die Zusammenarbeit mit BASIC, FORTH, PL/65 und PASCAL mit dem DOS. Bei

allen Sprachen muß der Beginn des Memory nach oben verlegt werden. Dies wird im mitgelieferten Handbuch ausführlich beschrieben. Es hat sich aber bewährt, in die Sprache einzugreifen, d.h. in den Initialisierungsroutinen und in den dazugehörigen Tabellen einige Bytes abzuändern, um einen Start ab hex 800 zu erreichen. Für die Arbeit mit dem Assembler ist ein Hilfsprogramm im Handbuch abgedruckt. Das Fehlen von Befehlen zum Behandeln von Files fiel hier besonders ins Auge. Als kleine Hilfe sind einige Hilfsroutinen als Anregung dort abgedruckt. Es hat sich hierbei gezeigt, daß man möglichst immer in einem dem DOS zugänglichen Format arbeiten sollte, um ein Diskettenchaos zu vermeiden. Einige Zeit wurde in FORTH mit einem Editor, der das Screenformat benutzt, gearbeitet. Dies wurde aber wieder fallengelassen, weil mit dem AIM-Editor und Compilierung über SOURCE komfortabler gearbeitet werden kann.

Zusammenfassung

Die Hardware ist ohne Fehl und Tadel (außer dem Preis), aber die Software ist mit fas 4 KB zu schmalbrüstig, so daß mindestens noch eine Systemdiskette wünschenswert wäre. Doch Software ist ja nur eine Frage der eigenen Entwicklung. Das Modul ist mit anderer Software auch für den AIM 65/40 vorgesehen. Mit anderen Computern müßte es zusammenarbeiten können. - Ein Vorteil dieses Systems ist die volle Transparenz des Floppy-Systems für den Benutzer. - Insgesamt eine gelungene Entwicklung und eine wirksame Verbesserung für den AIM 65. □□

Das MC-Terminal

Die Zeitschrift mc enthielt zu Anfang diesen Jahres das Platinen-Layout sowie weitere Aufbauhinweise für ein Video-Terminal 1/. Es wurde als Bausatz bezogen und ohne Schwierigkeiten montiert und in Betrieb genommen. Man muß aber bemerken, daß die abgedruckte Beschreibung kleine Druck- und Zeichenfehler enthielt, auf die man beim Bezug des Kernbausatzes bei 2/ aufmerksam gemacht wurde. Ferner: Beim großen Ansturm auf Bauteile gab es besonders bei der CPU eine Verknappung, die inzwischen überwunden ist.

Als Kernbauteile enthält die Terminalkarte einen TMS 9995 von TI sowie den vieltausendfach bewährten CRTC 6845 von Motorola. Ein EPROM sorgt für das Betriebsprogramm, ein weiteres enthält den Zeichensatz. Dabei ist dokumentiert, wie die Dekodierung der anschließbaren Tastatur ggfs. anders umzusetzen ist und wie die Zeichenmuster verändert werden können. Davon wurde kein Gebrauch gemacht. Es wurde die dort auch beschriebene und doch recht preiswerte Cherry-Tastatur angeschlossen, die auch Cursor-Kontrolltasten hat. Beide Teile sind jetzt Terminal an einem FORCE-System (betrieben mit 9600 Baud).

Neben dem Tastaturanschluß hat die Terminalkarte eine RS232-Schnittstellen, die bekanntlich mit +12 und -12 Volt betrieben werden, eine als Sender zum Computer, die andere als Empfänger. Sofern man auf der Empfangsseite die Möglichkeit benutzt, im TTY-Betrieb (optisch getrennt, 20 mA Stromschleife) zu arbeiten, kommt man ohne die beiden Hilfsspannungen aus. In dieser Art wurde die Karte für einige Wochen mit 9600 Baud als Video-Interface des AIM 65 betrieben. DIL-Schalter erlauben die Einstellung des Übertragungsformates (Stop bits, Parity, 7 oder 8 Datenbit, Baudrate zwischen 75 und 19200).

Die in industrieller Qualität gefertigte Karte kann bei einem Format von 100x195 mm durch einen Schnitt auf Europaformat gebracht werden. Sie ist dann mit einer 64-poligen Messerleiste zu bestücken, um die Signale herein- und herauszuführen. Andererseits kann man auf den 35 mm Überbreite auch einen Spannungswandler für die Hilfsspannungen montieren, die an vielen Laborarbeitsplätzen sowieso vorhanden sind. Die Gewissensfrage bleibt dann, ob man die auf der Überbreite einlötbaren Eckbuchsen für 25-pol. RS232 und 10-pol. Tastatur montieren will, um bequem bestecken zu können.

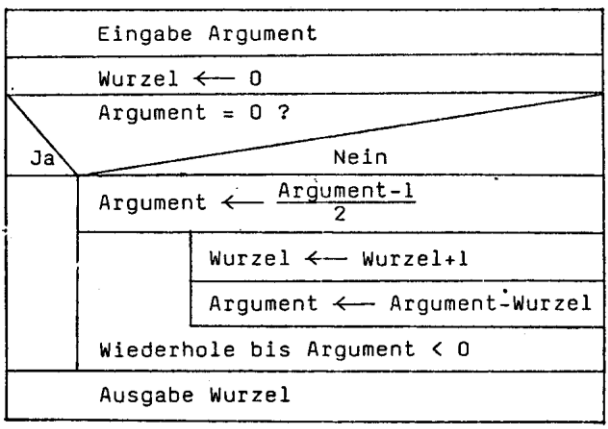
Beim Format von 80x24 Zeichen erzeugt die Karte ein deutlich lesbares Bild. Die bei den ersten Versionen noch aufgetretenen Streifen sind jetzt beseitigt. - Serielle Video-Schnittstellen haben den Vorteil, daß sie dem Computer keinen Speicherraum aus der Memory Map fortnehmen, der vielleicht schon von anderen Erweiterungen beansprucht wird und daß sie entfernt vom Computer betrieben werden können. Bei 9600 Baud erreicht man auch eine annehmbare Arbeitsgeschwindigkeit. Ob man sich demgegenüber für ein Video-Interface entscheiden soll, das ein mit den Prozessorbussen direkt verbundenes Video-RAM hat, sollte man im Einzelfall von den Umständen abhängig machen, insbesondere auch davon, wie es durch das Betriebssystem unterstützt ist, um bequeme Editiermöglichkeiten zu finden.

1/ Jäger,R., Regge, H-J. und Tobergte,G.: Das mc-Terminal. mc 1/83, S. 30 und 2/83, S. 57.
2/ Regge, H.-J., Fesenfeld 57, 2800 Bremen 1.

St.Dir. Peter Rix, 2350 Neumünster

Gerundete Integer-Quadratwurzel ISQR

Nachfolgend sind Struktorgramm und 16-Bit-FORTH Implementierung eines einfachen Algorithmus zur Berechnung der korrekt gerundeten Integer-Quadratwurzel angegeben:




```
: ISQR ( U --- ROOT )
       0 SWAP -DUP
       IF 1- 0 2 U/ SWAP DROP
          BEGIN
            SWAP 1+ SWAP OVER - DUP 0<
          UNTIL DROP
       THEN ;
```

Der FORTH-Algorithmus entnimmt einen 16-Bit-Betrag vom Stapel und ersetzt ihn durch seine Wurzel. Die Ausführungszeit wächst mit dem Argument und liegt bei AIM-FORTH zwischen 1 und 287 ms. - Der auch in anderen Hochsprachen mit Integer-Arithmetik leicht zu implementierende Algorithmus ist ursprünglich als Assembler-Routine entwickelt worden. Theorie und Dokumentation dazu sind in der ELEKTRONIK, Heft 23/1982 veröffentlicht worden.

Grundlage des Algorithmus ist die Tatsache, daß die Reihensumme

$$S_n = 1+2+4+6+8+ \ldots +2(n-1)$$

gerade die kleinsten ganzzahligen Argumente liefert, deren Integerwurzel auf n aufzurunden ist.

$$\frac{S_n-1}{2} = 1+2+3+4+ \ldots +(n-1)$$

ist dann eine durch einfaches Abzählen zu bildende Reihensumme. Wegen

$$\frac{S_n-1}{2} -1-2-3-4- \ldots -(n-1) = 0$$

ist der Wurzelwert n erreicht, wenn beim fortlaufenden Subtrahieren eine negative Differenz auftritt.

Spätlese

AIM 65: Verschiedentlich wurde schon erwähnt, daß BASIC Steuer- und Sonderzeichen nicht übernimmt. Herr Wolfgang Zweygart aus Böblingen schreibt dazu: Mich ärgerte, daß BASIC die Buchstaben 'ß' und 'ü. nicht vom Editor übernimmt. Es wurde festgestellt, daß die Routine ab hex B356 die Filtrierung aller ASCII-Zeichen vornimmt. Wenn man an den Stellen B37E, B37F, B386 und B387 NOP's ($EA) einfügt, werden alle ASCII-Zeichen übernommen. Damit es es auch möglich, Steuerzeichen für das Terminal oder einen externen Drucker in der Form von PRINT " .. " zu übergeben. Bei Bedarf sind also entsprechende EPROMs zu brennen oder RAM-residentes BASIC ist entsprechend abzuändern.

6502 steuert Arithmetikprozessor. Heft 24 enthielt einen entsprechenden Artikel von Herrn Thomas Kröger. Der Autor weist darauf hin, daß es auf Seite 33, 4. Absatz richtig heißen muß: Die vier Daten*ausgänge* D01 bis D04 ....

EPROM-Programmiereinheit für AIM 65 in Heft 20 und Heft 21: Nach einem Hinweis aus der Leserschaft sind in den Schaltplänen offensichtlich die Funktionen der Pins PB2 und PB3 zeichnerisch vertauscht worden.

Zweidimensionale Felder sortieren (CBM). Heft 20 enthielt auf Seite 12 einen entsprechenden Beitrag von Herrn Sellschop. Hierzu folgender Änderungsvorschlag von Herrn Michael Bauer, München 21: Die Routine wurde in ein Programm eingebaut und getestet. Es wurde schnell und korrekt sortiert - leider zu korrekt. Elemente der Länge 0 wurden als kleiner betrachtet und nach vorne gebracht, was für den Anwendungsfall nicht wünschenswert war. Es stellte sich heraus, daß die Abfrage nach der Länge des zweiten Strings ab $0587 steht. Sie lautet BEQ $05B3. Sie war nun so abzuändern, daß bei L=0 nicht vertauscht wird. Das geschieht durch Veränderung zu BEQ $05D6 (F0 4B).

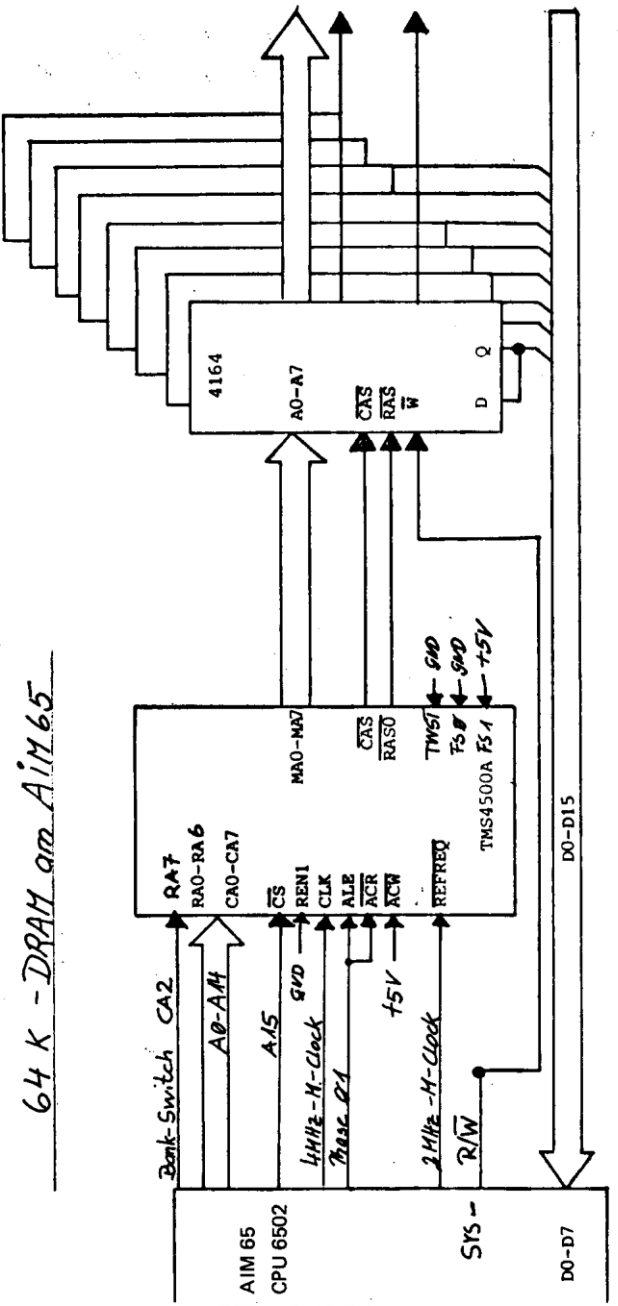
COLD RESET beim AIM 65: Heft 13 enthielt auf Seite 53 bereits einen Vorschlag, den AIM zurückrufbar zu machen, wenn er sich verlaufen hat und wenn außerdem der DILINK-Vektor keine Anzeige mehr erzeugt. Das Betätigen der RESET-Taste allein führt zu keiner Neu-Initialisierung dieses Vektors und damit auch zu keiner Anzeige. Bei einem RESET wird nämlich geprüft (ab Adresse E0DD), ob der NMIV2-Vektor verändert wurde, wenn nein, dann nur ein Warm Reset. Es kommt also nur darauf an, daß man nach A402/03 irgendeinen anderen Wert bringt, um eine volle Re-Initialisierung zu erhalten. □□

Rolf-Hubert Pobloth, 2207 Kiebitzreihe

64 K DRAM am AIM 65

Es wurde eine Schaltung mit 2x32 K dynamischen RAM für den AIM 65 aufgebaut. Dabei traten anfangs enorme Schwierigkeiten mit dem Timing auf, zumal es von Texas Instruments keine Erfahrungen und Applikationsblätter für den 6502 mit seinen überlappenden Phasen vorlagen.

Als Speicher werden die TI 4164 mit 64 Kx1 Bit benutzt. Der Speicher wird durch den Dynamic RAM Controller TMX4500A gem. folgendem Schaltbild angesteuert. Es wurde eine Anschaltung von 2x32 K gewählt, dabei ist der Adreßraum von hex 0000-7FFF durch den Bank-Switch zweimal zu benutzen. Die Bankumschaltung erfolgt dabei über den Pin CA2 eines Interfacebausteines, der auf RA7 geschaltet wird. A15 übernimmt am Controller das Signal CS (aktiv low). Die benötigten Frequenzen von 2 und 4 MHz werden vom Mutteroszillator abgegriffen. 4 MHz vom letzten Inverter und 2 MHz vom 7474 am Q-Ausgang. Die Phase 01 ist vom Expansionsstecker abgegriffen worden. R/W wird vom SYS-R/W und die Adressen und Daten werden vom Expansionsbus abgenommen. Der Refresh wird vom TMX4500A gesteuert und erfolgt in den ersten 500 ns des CPU-Taktes.


64 K - DRAM am AIM65
4164
A0-A7
D
Q
CAS
RAS
W
MA0-MA7
RA7
RA0-RA6
CA0-CA7
CS
REN1
CLK
ALE
ACR
ACW
REFREQ
TMS4500A FS1
TWST
FS0
FS1
GND
GND
+5V
D0-D15
Bank-Switch CA2
A0-A14
A15
GND
4MHz-H-Clock
Phase 02
+5V
2MHz-H-Clock
R/W
AIM 65
CPU 6502
SYS -
D0-D7

Aus der Branche - Produkte

FORTH: Das nächste Treffen des norddeutschen FIG-Chapters findet am Sonnabend, den 23.4.83, um 16 Uhr im Lokal BLIMP statt, Hamburg 19, Müggenkampstr. 63. Kontakt: FIG, c/o Klaus Schleisiek, Postfach 20 22 64 in 2000 Hamburg 20, Tel.: 040-480 81 54.

MICRO-Treff 83 am 14. und 15.5.83 in Ludwigshafen. Die Arbeitsgemeinschaft Micro Computer in Deutschen Amateur Radio Club bittet zu ihrem fünften (beliebten) Meeting mit Ausstellung, Kurzvorträgen und Flohmarkt nach Ludwigshafen, Willi-Graf-Haus am Ruthenplatz, Leuschnerstraße 151. Sa. von 10-18, So. von 10-16 Uhr. (Der Herausgeber wird dort Sa./nachm. und So./vorm. anwesend sein). Kontaktadresse für Fachfragen: Kuno Schöllhorn, T. 0621 - 56 83 70.

Die CMOS-CPU R65C02 von Rockwell wird erst ab etwa Juni/Juli 1983 in Stückzahlen geliefert werden. Eine Bemusterung wurde allerdings schon bei verschiedenen Abnehmern vorgenommen. Von hier aus ist zu berichten, daß sie auf dem AIM 65 einwandfrei läuft, auch zusammen mit dynamischen RAMs, wenn man nach dem Vorschlag des Herstellers einen Serienwiderstand von 220 Ohm vor Pin 8 (+5 V) und etwa 100 pF zwischen SYNC und Masse legt. - Der erweiterte Befehlssatz der CPU wird nach weiterer Erprobung Anlaß geben, zusätzliche Programmiertechniken hierfür darzulegen.

Rockwell International sandte verschiedene Pressemitteilungen und Datenblätter: PROM-Programmer für den 65/RME-Bus. Das Modul RM65-2901E kann EPROMs und E^{2}PROMs von 1, 2, 4 oder 8 K mit 24 und mit 28 Polen programmieren. Preis. $295. **R6501Q Mikroprozessor für Zweichip-Mikrocomputersysteme:** Dieser interessante Chip mit 64 Pin, separaten Adreß- und Datenbusleitungen hat 4 Ein- und Ausgabeports, zwei Zähler sowie 192 Byte RAM, eine voll duplex Seriellschnittstelle mit programmierbaren Baudraten sowie 10 Interrupts. Der Befehlssatz enthält vier neue Bit-Manipulationsinstruktionen, auch Branch on Bit Set oder Clear. In der Summe eignet dieser Chip (1 oder 2 MHz) für den Aufbau sehr kompakter Systeme, gleichwohl kann der volle Adreßraum ausgenutzt werden. Document No. 29651N48, Bestell-Nr. 2145, Oct. 1982. **Für die 68000-Familie ist der R68561 MPCC (Multi Protocol Communications Controller)** in Musterstückzahlen lieferbar. Er ermöglicht Datenübertragungsgeschwindigkeiten von 4 Mbit/sek und unterstützt voll duplex synchrone und asynchrone Empfänger- und Senderoperationen und Übertragungsformate. Dem 68000 kann er drei programmierbare Interruptvektoren mitteilen. Die Version 68560 ist für 8 Bit-Prozessoren bestimmt und im 40-poligen Gehäuse lieferbar.

Die Schulungsabteilung der Fa. Commodore Büromaschinen führt auch in diesem Jahre wieder zalreiche Kurse für Programmiersprachen und die Bedienung der Softwarepakete durch. Info über Lyoner Str. 38, 6000 Frankfurt 71, Tel.: 0611 - 66 38 159.

Streamer zur Datensicherung mit einer Kapazität von 5 und 10 MByte liefert jetzt die Fa. CTT GmbH, Kreillerstr. 21, 8000 München 80. Sie arbeiten mit den bewährten Seagate-Laufwerken.

Weitere Platinen der FORCE Computers. Nach einem Bericht der Firma soll das Lieferprogramm in den nächsten Monaten wie folgt abgerundet werden: Motherboards mit 9 oder 20 Steckplätzen (im März), dyn. 512 K RAM-Platine, stat. Platine für ROM/CMOS-RAM mit wählbaren Basis-Adressen und Batteriepufferung (April), serielle SIO-Platine (I/O mit mindestens 6 RS232-Schnittstellen) Floppy-Interface SASI für mindestens 4 Floppys und zwei Winchester (Juni), SYS68K/CPU-2 mit extern (global) zugreifbarem RAM (Juli), Grafik-Platine mit 1024x1024 Bildpunkten und mindestens 16 Farben, SYS68K/CPU-3 mit 68010 Virtual Memory Processor, 68451 MMU und 68450 DMA-Controller, die als Master-CPU für die Boards /CPU-2 und -3 eingesetzt werden kann. Die /CPU-3 soll voll unter Betriebssystem UNIX laufen. Ab Juni 83 soll FORTH ROM-resident angeboten werden. FORCE stellt in Hannover bei der SE-Elektronik, Halle 2 OG 1207 aus.

Speicheradapter für CBM-Rechner. Der in CBM-Rechnern nicht genutzte Adreßraum von $E900-EFFF kann mit einem EPROM 2716 oder einem RAM 6116 nutzbar gemacht werden. Der Adapter nimmt dieses sowie das normale ROM E000-E7FF auf und enthält die Dekodierung. Gebohrte Platinen und fertige Adapter sind zu beziehen bei Fa. Peter Engels, Kreisstr. 29, 5308 Rheinbach.

Bücher

Heß, K.-H.: Programme für CBM/VC 20-Computer. Verlag Markt & Technik, Haar 1983, 150 S., ISBN 3-922120-28-8, DM 32,--. Mit Problembeschreibung, -lösung und Programmbeschreibung stellt der Autor folgende Themen vor: Etikettenbeschriftung, Lotto, Hardcopy, Kopfrechentraining, Bildschirmumrahmung, Römische Zahlen, Sonderzeichen,Code-Wandlung, Balkendiagramme, Laufbandanzeige, Banküberweisungen ausfüllen.Damit ist es vor allem dem Computerneuling gewidmet, der Profi wird hier und da wohl einen Kniff absehen können.

Falkner R.: Mikrocomputer-Lexikon. DEV Verlags GmbH, München 1982, 181 S., ISBN 3-923 858-00-0, DM 29,50. Der Untertitel lautet: 1500 Fachbegriffe exakt definiert mit Register Englisch/Deutsch. Das Lexikon erhebt den Anspruch auf die bessere Definitionsqualität und die größere Anzahl der Begriffe. Es ist tatsächlich ein nützliches Nachschlagewerk für Benutzer, die sich in die Fachsprache einarbeiten und für solche mit wenig englischen Sprachkenntnissen. Wer lange schon im Fach tätig ist, würde hier und da auch anders definieren und sicher auch 500 und mehr seltener auftauchende Begriffe in einer nächsten Auflage finden wollen.

Koch, J., Hrsg. Valvo Unternehmensbereich Bauelemente der Philips GmbH: Der 16bit-Mikroprozessor SC 68000 - Eigenschaften. Verlag Boysen + Maasch, Hamburg 1982, 105 S., ISBN 3-870 95-255-5, ca. DM 19,50. In mancher Weise darf das Buch als deutsches Datenblatt für den Prozessor bezeichnet werden, denn es enthält alle wesentlichen Beschreibungen des amerikanischen Vorbildes. Für die Hardware treten jedoch viele eigene Verweise und Erklärungen hinzu, so daß dieses Buch besonders wegen seiner deutlichen Formulierungen dem deutschen Leser für die Einarbeitung empfohlen werden darf. Im April soll ein zweites Buch aus gleicher Feder zu Softwarethemen folgen.

Schumny, H. Hrsg.: BASIC und PASCAL im Vergleich. Vieweg Programmbibliothek Mikrocomputer 3, Braunschweig 1983, 82 S., ISBN 3-528-04224-9, DM 24,80. Gleiche Aufgabestellungen aus dem mathematischen und dem spielerischen Bereich werden in den genannten Sprachen zusammen auch mit Programmablaufplänen gegenübergestellt und die Argumente für und wider die Formulierungsmöglichkeiten erörtert. Klar zeigt sich eigentlich, daß man mit PASCAL saubere Strukturen erzielt, die weitgehend selbstdokumentierend auch optisch gegliedert sind.

Editorial

Die beiden in diesem Heft enthaltenen Text Editor-Programme werden wohl nicht die letzten Editoren sein, denn der Herausgeber meint, daß ein leistungsfähiges Editiersystem nicht nur die Programmentwicklung erleichtert, sondern daß die allgemeine Textverarbeitung für jeden schreibenden Bürger von Bedeutung ist. Wir finden bereits heute täglich Briefe in der Post, die am Personal Computer editiert und über Matrixdrucker oder Typenradschreibwerk (elektronische Schreibmaschine) ausgegeben wurden. Diese Beobachtung erhellt die Bedeutung, die man dem schreibenden Computer zumißt.

Der eine ist ein Screen-Editor, der ein Fenster für die textliche Umgebung offenhält, in der man arbeitet. Der andere nimmt Steuerzeichen in den Text auf, die bei Bedarf sichtbar gemacht und editiert werden können. Diese Steuerzeichen dienen vor allem als Drucker-Steuerzeichen. Interessant ist hier das Prinzip 'mapped': sie können aktiviert und passiviert sein. Ein dritter Editor ist auch in diesem Heft besprochen worden, der FORCE IDEAL 2.0. Er hat die Fähigkeit, die von einem seriellen Terminal übermittelten Cursorbewegungen in Neu-Arrangements des Textspeichers umzusetzen. Auch dieses Prinzip ist für andere Rechner verfolgenswert.

Zu 16 Bit: Es ist vorgesehen, von nun ab ständig für den 68000, seine Interfacebausteine und den VMEbus zu berichten, und zwar für Hardware und ihre geeignete Programmierung. In diesem Heft wird zunächst die CPU in einer knappen Übersicht vorgestellt, ferner ihr Interfacebaustein 68230 PI/T. Aus diesen Berichten möge der Leser den zutreffenden Eindruck gewinnen, daß die Einarbeitung in diesen verwandten Prozessor gar nicht so schwer ist, wenn man vor dem Lesen der umfangreichen Datenblätter und Bücher bereits die Leitideen hat erfassen können.

Die Sprache FORTH wird auch in Zukunft weiter intensiv betreut. In diesem Heft mußte sie platzbedingt einmal pausieren. Weiter Artikel liegen schon vor. - Für den 68000 gibt es von der FIG in den USA ein Source-Listing für die Implementierung. Es hat etliche Mängel: Es ist erstens nicht verschieblich geschrieben, es kann nur die ersten 64 K Adreßraum überstreichen und seine Assemblerformulierungen unter FORTH sind in besonderem Maße gewöhnungsbedürftig. Besser als abzutippen scheint der Versuch zu sein, mit einem leistungsfähigen Assembler einen grundsätzlich neuen Aufbau zu wagen.

Heft 29 hätte richtig den Erscheinungsmonat 'Februar 1983' tragen müssen. Der Fehler entstand beim Titelsatz in der Dunkelkammer. Das vorliegende Heft Nr. 30 ist mit 'April' damit genau in der Schedule. Für hinzukommende Leser wird der Nachbezug früherer Hefte jetzt verbilligt (siehe Bezugsbedingungen auf der Rückseite). Im Gegenzug ist die Herausgabe weiterer zusammenfassender Bücher für die früheren Ausgaben *nicht* geplant. Kapital sollte besser für die stetige Innovation investiert werden. Gleichwohl bleibt alle früher veröffentlichte Information einstweilen nachlieferbar.

Kleinanzeigen

AIM 65, ASM, BASIC, 8080 XASM, PROM-Programmer, 32 KRAM, Dig.-Cassette, Gehäuse, Netzteil, Videointerface 64x16, Juniorbus, VB 2500 DM. Tel. 030 - 803 78 58.

REMON 3.2: Verbessertes Monitorprogramm für AIM 65, z.B. Erweiterung von Editor- und Monitorbefehlen, Tastenrepeat, Kleinschreibung, deutscher Zeichensatz u.v.a.m.. Software in zwei 2532-EPROMs für DM 88,-- einfach in die Steckplätze Exxx und Fxxx auf dem AIM einsetzen. Bestellung b. Dipl.-Ing. R. Wollenberg, Stockumer Str. 234, 4600 Dortmund 50, T. 0231-75 34 77.

AIM 65 4 K RAM, Gehäuse, Assembler, Math-Package, FORTH, BASIC, PL/65, PASCAL. VHB DM 1700. H.-G. Nülens, Hustadtring 53, 4630 Bochum, Tel. 0234 - 70 36 05.

XITEX-Video-Terminalplatine SCT-100, wahlweise TTY- oder RS232-Anschluß, Tastaturanschluß, viele Steuerzeichen, mit Doku. VHB DM 180. Roland Löhr, Tel. 04102 - 55 816.

32K-Bytes CMOS Speicherplatine

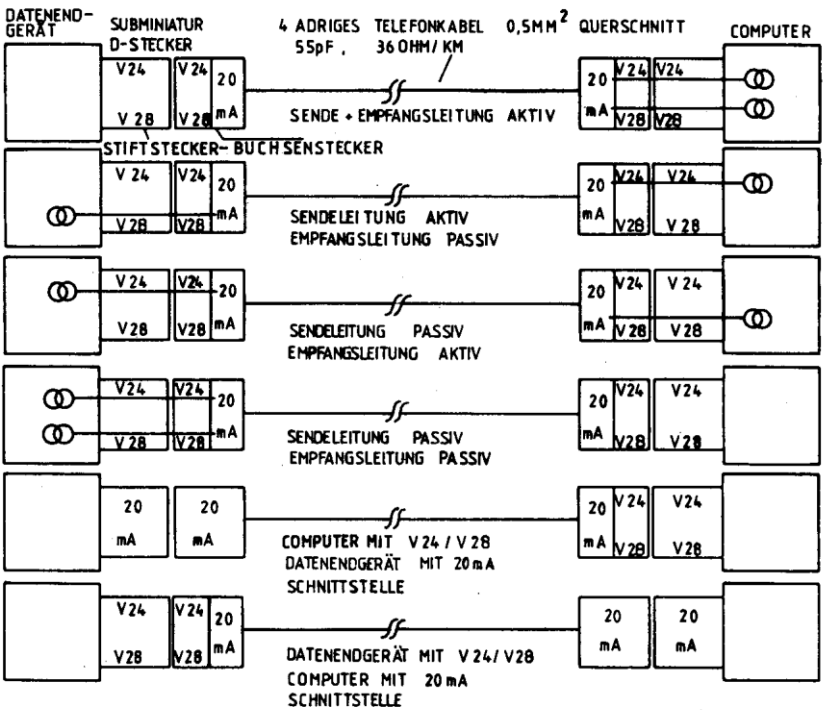
mit Pufferbatterie, Adressen 1000-9FFF für AIM 65 und PC 100, wird auf dem Rechnerboard angeschlossen, so daß der Expansionsstecker frei bleibt und die Platine im jeweiligen Rechnergehäuse untergebracht ist. Die Platine kann auch mit EPROMs vom Typ 2716 bestückt werden. **Preis DM 945,--,** komplett bestückt mit TMM 5517 AP, Anschlußstecker und Kabel. Info gegen Rückporto. Jessica Stecker, Gustav-Cords-Str. 7, 5000 Köln 60.

MC-Terminalkarten 80x24 Zeichen, max. 19200 Baud. Leerkarte DM 78,--. CPU, UART, Video-Controller, Quartz DM 138,--, EPROM-Satz DM 48,--. **ASCII-Tastaturen Cherry G80-246,** deutsche T.–Anordnung mit Auto-Repeat DM 225,--. Gehäuse orig. Cherry DM 47,50. **Netzteil auf Europakarte** +5V/6A, +12V/2A, –12V/1A, –5V/1A DM 164,--. **6502 Universalplatine** s. Titelbild MC 2/82, bes. für AIM 65 geeignet. Leerkarte DM 73,--. (Alles inkl. MWSt/plus Porto). Regge, Fesenfeld 57, 2800 Bremen 1, Tel.: 0421 - 7 11 14.

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.



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.
Der Preis: 339,00/Stück incl. Umsatzsteuer, 1 Jahr Garantie.

STECKER
INGENIEURBÜRO

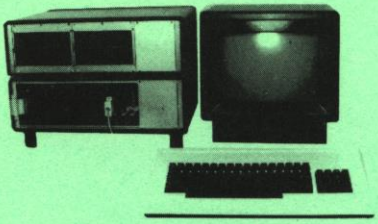
Postfach 60 07 66 5000 Köln 60 Tel.: (0221) 7 12 40 18
Neue Telefon-Nummer!!

GWK

GESELLSCHAFT FÜR TECHNISCHE ELEKTRONIK mbH.
HARDWARE SOFTWARE SYSTEMENTWICKLUNG

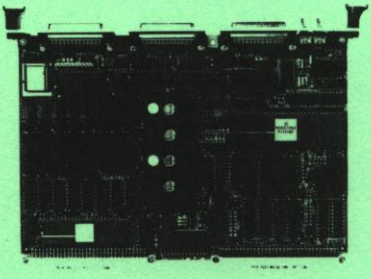
MICROCOMPUTER

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

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.

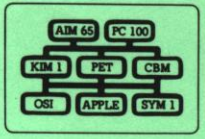
Bezugsbedingungen: Abonnement ab laufender Ausgabe für 6 Hefte DM 54,-- (Inlandsendpreis). Ausland/foreign via surface mail DM 59,--, USA air 26 Dollar. Abonnements laufen bis auf Widerruf mit Kündigungsmöglichkeit bis zu zwei Wochen vor deren Ablauf.

Nachliefermöglichkeiten: Hefte 1-6 und 7-13 sind als Buch nachlieferbar (s. Anzeige unten). Hefte 14-29 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.

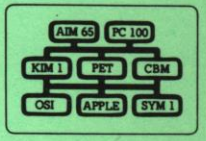
Leser-Service des Herausgebers

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
mit Anleitung für den leichten Einbau. Auffrischung des Druckes 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.

2-Pass-Assembler für R65C02 (erw. Befehlssatz) noch in Vorbereitung.