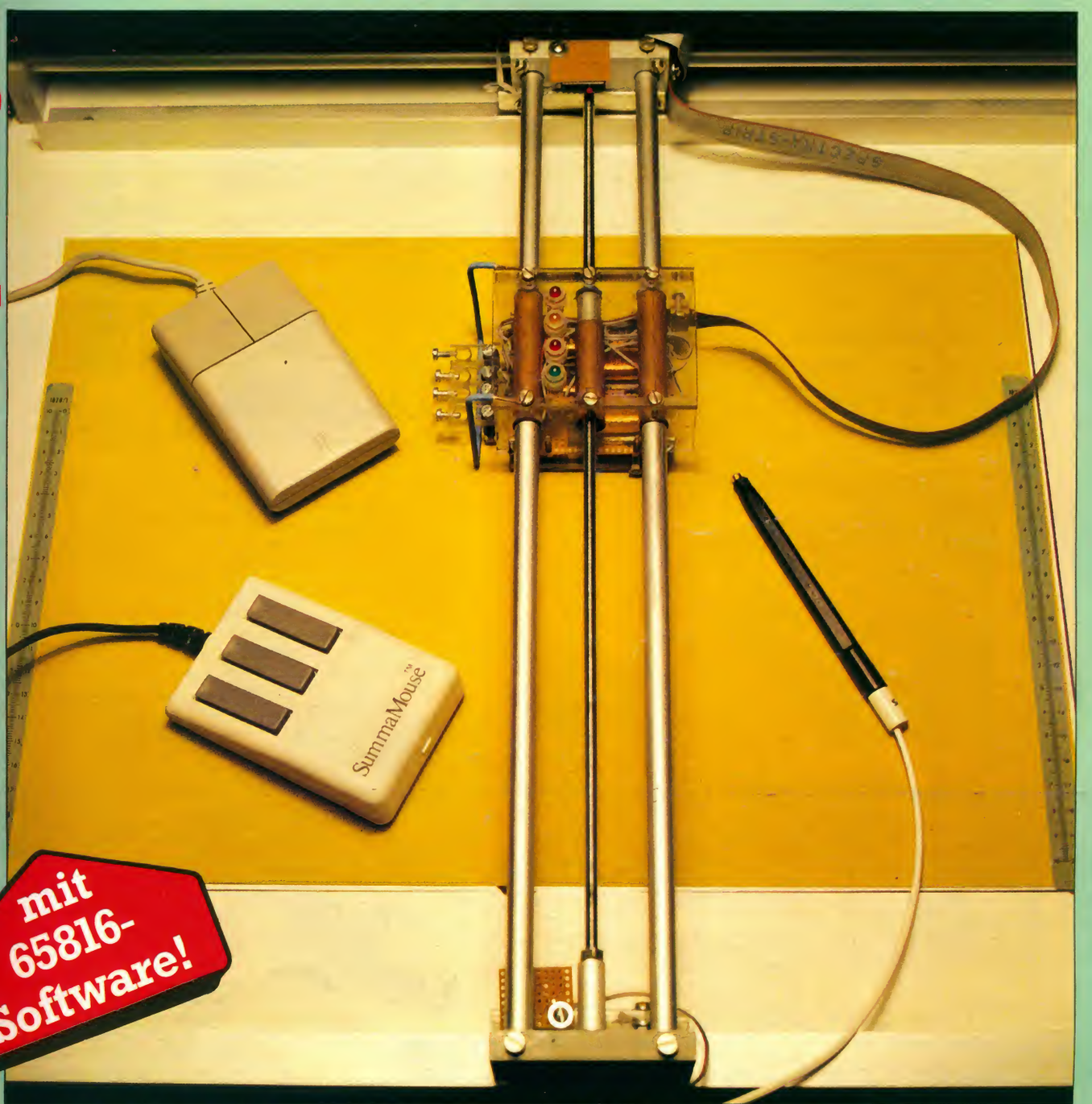
elektor 5. Computing-Sonderheft

elektor COMPUTING

DM 18,–
öS 140
sFr 18,–



mit 65816-Software!

INTERFACE-TECHNIKEN UND ERGONOMIE

UCSD-p-System • Selbstbau-Plotter

8 MHz-Z80-Karte • Maus-Interface

. . . ACS KEYBOARDS

universell
intelligent
kompatible

Modell AN 111 FXT
Das Keyboard für deutsche Textverarbeitung, einsteckfertig, passend für PC/XT-Kompatible (AT in Vorbereitung)

Zusätzliche Leistungsmerkmale:
- 2fach PROG-Modus für 10 Fixprogrammiertasten
- Abgesetzter 10er Block, große RETURN & ENTER Taste
- 10 LEDs für CAPS-LOCK, NUM-LOCK, PROG usw.
- 7 weitere sinnvolle Funktionstasten, 2 Paßwörter

ACS GmbH
steuerungssysteme
computerperipherie
datentechnik

D-4930 Detmold
Alter Mühlenweg 2–5
Telefon 05231–4176/4177
Telex 935660 acs d

Böhm Microcomputer EC68K

Vorstellung des Böhm EC68K in Elektor 11/85, anwendungsorientierte Beschreibung in Elektor-Computing ab Dezember '85

Die EC68K-Module im einzelnen:	Best.-Nr./ Bausatz-Preis
• CPU mit dem 16-Bit-Prozessor 68008, Taktfrequenz 8 MHz, Echtzeituhr mit eigenem Akku	37 601 499,— DM
• Speicher für 16 Speicherbausteine 8x8. Anzahl und Typ anwenderabhängig, EPROMs und RAMs beliebig mischbar. Bis zu sieben Speicher-Platinen sind im System einsetzbar.	37 602 159,— DM
8kx8 EPROM 8 St. zum Tagespreis	80 601 59,— DM
8kx8 RAM	80 602 119,— DM
• Video-/Tastatur-Interface Ausgang: Video-Signal für Monitore, Bildschirmauflösung per Software variabel, z.B. 80 Zeichen bei 25 oder 50 Zeilen. Eingang: Parallel-Tastatureingang für ACS II-Tastaturen	37 603 299,— DM
• Seriell-Interface mit vier unabhängigen Kanälen, einzeln zwischen TTL und RS 232 umschaltbar. Einstellbare Übertragungsgeschwindigkeit (Baudrate) bis zu 38 400 Zeichen pro Sekunde. Anschlußmöglichkeiten z.B. für Drucker, Modem, EPROM-Programmiergerät	37 604 236,— DM
• Floppy-Interface Mit Betriebssystem und Controller für zwei Laufwerke. (Mehrere Interfaces einsetzbar.)	37 605 347,— DM
• Laufwerk 3½" mit Anschlußmaterial	37 615 355,— DM
• EPROM-Simulator Dual-Port-RAM-Speicher, simuliert bis zu 256-kBit-EPROMS	37 608 291,— DM
• MIDI-Interface zwei getrennte Kanäle zur Verarbeitung unterschiedlicher Daten z.B. von MIDI-KEY, Expander, Orgel usw., auch zur Klangfarbenprogrammierung der MUSICA DIGITAL 1000/1030	37 607 186,— DM
• BUS mit Kontaktleisten für 13 Steckplätze	37 610 198,— DM
• Netzteil sekundär getaktet, mit Transformator, +5V/6 A sowie +/— 12V/0,5 A	37 611 298,— DM
• 19"Einschubgehäuse	37 620 365,— DM
• ASC-II-Tastatur	37 630 342,— DM
• Monitor	37 640
• Verschiedene Software-Systeme auf Anfrage, weitere Module in Vorbereitung	

Böhm
Kuhlenstr. 130-132 · 4950 Minden · Tel. (0571) 5045-0

der flexible 16-Bit-Computer im Bausatzsystem

ELEKTOR

Neu

6502-Maschinencode für schnelle Grafik

von K. Diedrich und G. Klein
136 Seiten
ISBN 3-921608-40-6
DM 23,— (+ DM 2,50 Porto/ Versand)

Maschinencode! Welcher Computerfreak träumt nicht davon, mit Hilfe der Maschinensprache alle Möglichkeiten seines Mikroprozessors voll auszuschöpfen. Das Buch sorgt dafür, daß es nicht nur ein schöner Traum bleibt. Es bringt die Maschinenbefehle "spielend" an die Frau/den Mann.
Die Autoren erläutern die wichtigsten 6502-Befehle anhand schneller, bewegter Grafik. Die einfachen Befehle stehen am Anfang und erleichtern den Übergang zum Verständnis der komplexeren Befehle. Wo es angebracht ist, ziehen die Autoren Parallelen zwischen den Maschinenbefehlen und Situationen des täglichen Lebens. Eine Vielzahl von Demonstationsprogrammen machen die Theorie recht abwechslungsreich. Das Maschinenprogramm "MASTERPAINT" einschließlich der Tabellen und Darstellungen erleichtert den Umgang mit der Blockgrafik erheblich.

FORTH

Programmiersprache der vierten Generation

von Rüdiger Birkemeyer
126 Seiten
ISBN 3-921608-38-4/
DM 28,50 (+ DM 2,50 Porto/Versandkosten)

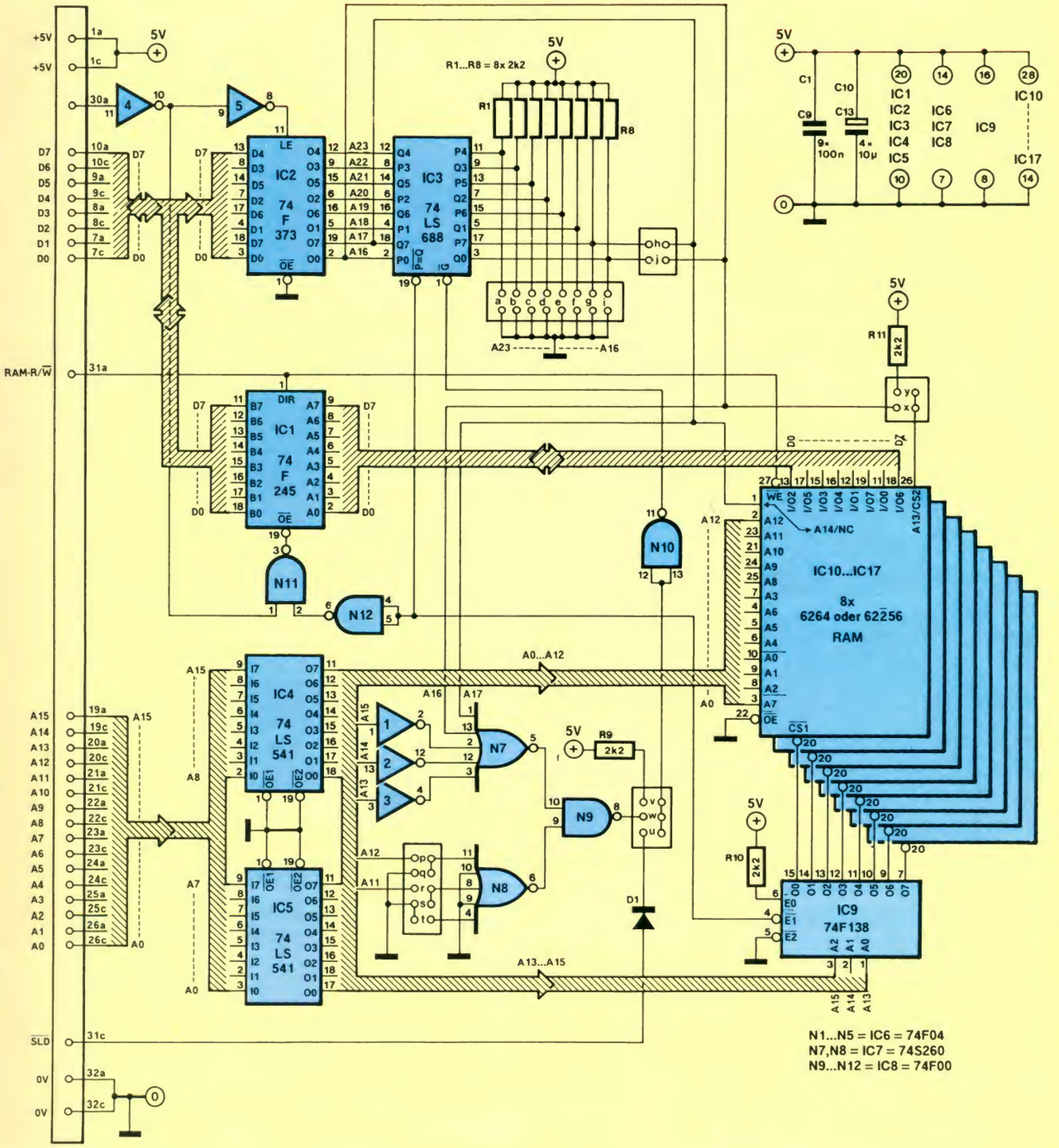
Wer gerne programmiert und seinen Computer optimal nutzen will, hat mit FORTH dazu jede Gelegenheit. Denn die Besonderheit von FORTH, der Programmiersprache der vierten Generation, ist: Der Anwende kann FORTH fast beliebig erweitern und eigene (Befehls-)Worte hinzufügen.
Der Autor baut in seinem Buch die Sprache systematisch auf und beginnt mit der Stapelorganisation, einem sehr wichtigen Detail bei FORTH. Es folgen Kontrollstrukturen, Schleifen und Verzweigungen, die für die Programmierung unerläßlich sind. Auch der Editor erleichtert das Schreiben eines Programms erheblich. Mit dem FORTH-Assembler ist es schließlich noch möglich, neue Worte direkt in Maschinensprache zu formulieren. Das macht die Kommunikation zwischen Anwender und Computer noch schneller.

Am besten bestellen Sie mit der Bestellkarte am Heftende.

Elektor Verlag GmbH
Süsterfeldstr. 25 5100 Aachen

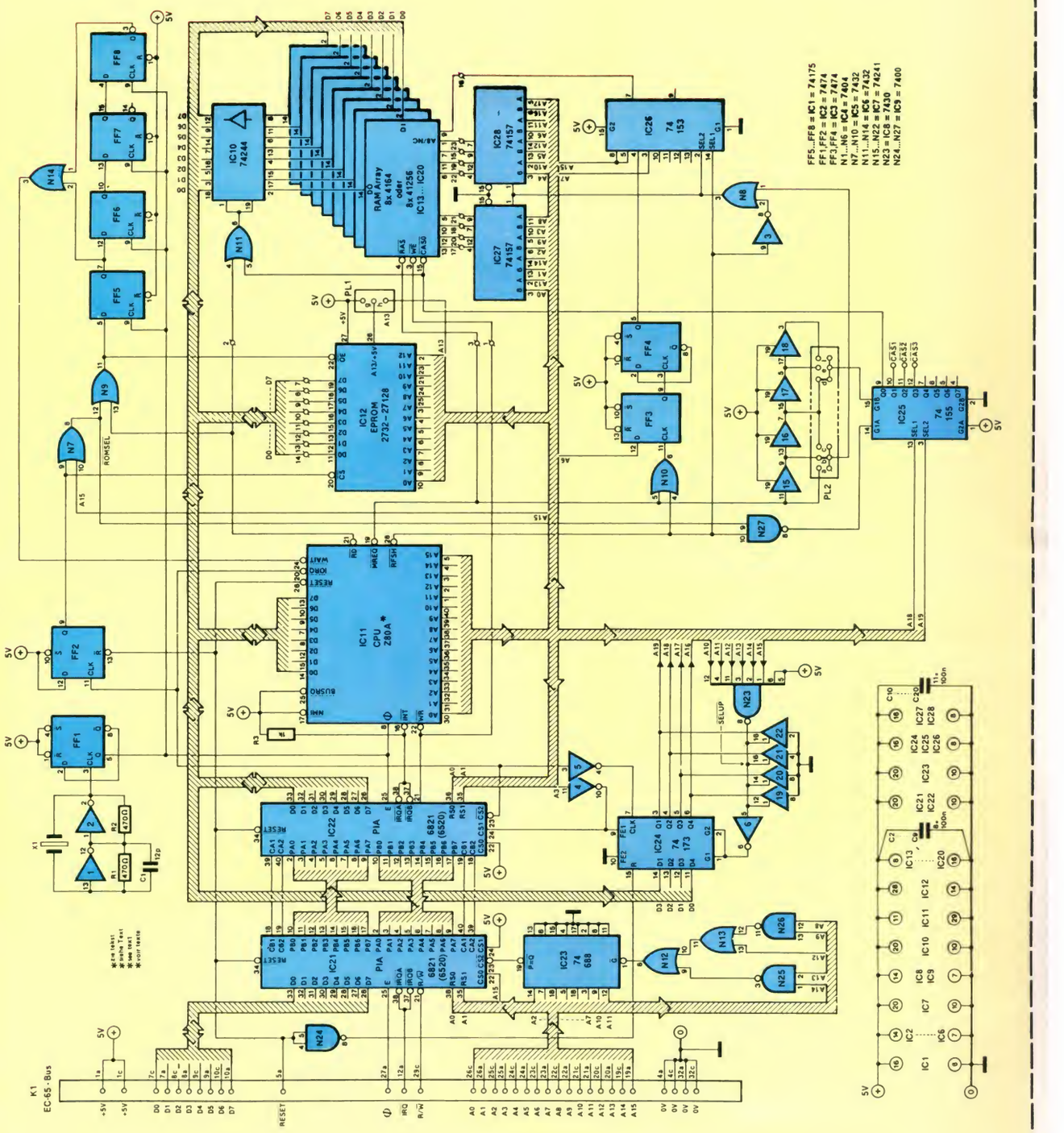
Zum Ausschneiden:

EC-65K: 64/256 KByte SRAM-Karte



Zum Ausschneiden:

EC-65 (K): Z80-Karte



MSX

Das Sonderheft für MSX 2 + 1

. . . informiert auf 100 (!) Seiten MSX-Interessenten und MSX-User über den neuen, erweiterten MSX-2-Standard, ohne die MSX1-User zu vernachlässigen.

Soft- und Hardwarebeiträge sind so gestaltet, daß sie für beide MSX-Versionen gelten. Informative Artikel machen das Heft für jeden MSX'1er und Computerfreak interessant:

Nach Eins kommt Zwei
MSX1 und MSX2 im Vergleich

Der Videoprozessor V9938

Basicode
Die gemeinsame Computersprache auch für MSX-Rechner

* **Multitext**
  Ein starkes Textverarbeitungsprogramm
* **Der Speicher für MSX2-Rechner**
  Ein Gedächtnis wird vorgestellt
* **Neue BIOS-Routinen bei MSX2**
* **Test**
  Thermo-Transfer-Printer Brother HR5
  Yamaha-Musikcomputer
* **Der MSX-Computer als Meßgerät**

* **Listings**
  Tontaubenschießen
  Menütechnik
  Zeichensatz-Konverter
  Directory
  Memory
  und neun weitere interessante Programme

Das ist nur ein Auszug aus dem Sonderheft "MSX +". Informieren Sie sich direkt und schauen Sie ins Heft.

Ab sofort im Zeitschriftenhandel, oder direkt beim Verlag.
DM/sFr 15,— öS 115,—

Elektor Verlag GmbH,
Süsterfeldstr. 25, 5100 Aachen

Bestellen Sie mit der Karte am Heftende!

COMMODORE WELT
20/64/128
Das unabhängige Commodore-Magazin

JETZT AN IHREM KIOSK

AGRESSIV: Commodore greift an

UMFASSEND: Neue Basic-Befehle für den VC 20

GETESTET: Echte Software für 128 PC

PROFIHAFT: 60 Zeichen am 64 er-Bildschirm

INTENSIV: CP/M-Das unbekannte Wesen

Editorial

HILFE...

...das ist Zuviel: Zuviel des Guten nämlich!
Seit ec 4 erschienen ist, mußte ich oft den Hörer nebens Telefon legen, um an ec 5 arbeiten zu können. Und ein Stapel Briefe wartet auch noch auf Beantwortung...

Liebe Leser, Eure Botschaft hat die Redaktion erreicht:
"ec 4 war das bisher beste Heft dieser Sonderheft-Reihe. Weiter so! Ich wünsche mir... Wann kommt die XY-Karte, wann kommt die XZ-Software...?" Sammelantwort: Wenn ich telefonisch nicht erreichbar war oder Briefe (noch) nicht beantwortet habe, so lag dies an meinem Versuch, ec 5 noch besser zu machen. Ob dies gelungen ist, darüber müssen Sie urteilen.

Eines ist jedenfalls klar: Ohne die hervorragende Mitarbeit der Autoren wäre diese Heftreihe nicht möglich, und deshalb geht an dieser Stelle unser ausdrücklicher Dank an alle, die die ec-Sonderheftreihe mittragen.

Und jetzt noch einige Details:

Ausgerechnet beim EC-65K ist unser Zeitplan durcheinander gekommen — auch hier größtenteils nach dem Motto: Zuviel des Guten. Aber es gab auch zwei Dinge, auf die die Redaktion schlichtweg keinen Einfluß hat. Nähere Informationen hierzu finden Sie in dem Artikel: "EC-65K / 65816: Software" im Innenteil des Heftes.

Die in ec 4 erwähnte 6809-Karte für den EC-65 und EC-65K haben wir vorläufig zurückgestellt — zugunsten einer V30-Karte, mit der unsere beiden "Spitzenreiter" (was die Anzahl der Nachbauten angeht) MS-DOS-fähig werden sollen. Deshalb enthält dieses Heft auch den ersten Teil einer Serie zum V30. Es kann passieren, daß hierdurch nicht genug Platz für den vierten Teil der Serie "Die Nachfolger" (zum 65802/65816) bleibt, oder diese in einen vierten und fünften Teil "zerlegt" werden muß. Dies hätte allerdings auch einen positiven Aspekt: Ich könnte die Erfahrungen, die ich mit dem 65816 im Moment sammle, in die Überarbeitung einfließen lassen.

Wir hoffen, die Nachbauer des Big-Board II, des EC-68 und des EC-68K werden durch Projekte wie den Spooler, das Mini-Modem und Auto-Dial für diesen (siehe nebenan im Inhaltverzeichnis) und natürlich auch das Mausinterface und den Selbstbau-Plotter ausreichend dafür entschädigt, daß die genannten Projekte im Moment — also in ec 4 und diesem Heft — etwas unterrepräsentiert sind. Es liegt so manches in der Schublade, und ein klein wenig schlechtes Gewissen haben wir deshalb auch: Das wird die Planung von ec 6 entsprechend beeinflussen.

Kein schlechtes Gewissen haben wir, weil Sie im Moment manchmal länger auf eine Platine warten müssen: Wir freuen uns, daß Sie so fleißig unsere Projekte nachbauen. Die Erst-Serie mancher Platinen ist schon ausverkauft, bevor Sie überhaupt gefertigt ist. Und weil die Fabrik, die diese Platinen für uns fertigt, das allgemeine Anziehen der Konjunktur sehr deutlich spürt, dauert es leider im Moment etwas länger, bis uns die zweite und teilweise dritte Serie erreicht. Aus diesem Grunde haben wir die Platinen für das Maus-Interface und die Steuer-Elektronik des Plotters schon vorsichtshalber gleich auf ec 6 verschoben.

Trösten wir uns gemeinsam damit, daß einige Meteorologen einen besonders schönen Sommer prophezeien (andere behaupten natürlich das Gegenteil...). Im Herbst sehen wir uns dann mit ec 6 wieder.

Ihr

(Peter von der Linden)

PS. Uns ist da eine Idee gekommen...

Obwohl ursprünglich garnicht als solches gedacht, entwickelt sich der EC-65K zu einem Rechnersystem, das aufgrund seines modularen, transparenten Aufbaus und der Möglichkeit, mehrere Betriebssysteme zu "fahren", für den Einsatz in Schule und Ausbildung hervorragend geeignet ist. Eine Firma, die den EC-65K als Fertiggerät oder in Form von geprüften Baugruppen anbietet, ließe sich sicher finden. Wenn Sie als Leser im Bereich Schule und Ausbildung tätig sind und dies ebenfalls für eine gute Idee halten, nehmen Sie doch einfach mit der Redaktion Verbindung auf.

INHALT

Interface-Techniken und Ergonomie 9
Mensch und Computer - da stoßen zwei Welten aufeinander. Wie aus diesem Zusammenstoß ein optimales Miteinander werden kann, ist Thema dieses Artikels

V30: Eine Einführung 15
Der High-Speed-CMOS-Prozessor V30 ist nicht nur abwärtskompatibel zum 8086, sondern bringt auch einiges mehr durch seine ausgefeilte innere Struktur. Was wir damit im Sinne haben? Siehe Editorial

EC-65K: SRAM-Karte 19
Maximal 256 KByte bringt die neue Speicherkarte für unseren "Nachfolger". Und sie kann in jede beliebigen Teil des 16-MByte-Speicherbereichs des 65816 gelegt werden!

Auto-Dial-Interface 24
Automatisches Wählen per Computer ist sicher für jeden Modem-Besitzer interessant. Hier ist die Lösung - nicht nur fürs Mini-Modem aus dem letzten ec-Heft.

CP/M auf dem EC-65 (Teil 1) 27
Die Hardware der Z-80-Karte ist noch schöner geworden als erhofft. Geflucht hat nur unser Platinen-Layouter: Es geht eng zu!

Mäuse 34
Ein hochauflösendes Mausinterface zum Selbstbau - das paßt gut zum Grundlagenthema dieses Heftes

Ein 64-K-DRAM-Zusatzspeicher 37
Die "Wiederverwendung" von Platinen hat Tradition bei uns. Diesmal wird aus unserer guten alten DRAM-Karte ein 64 KByte-Zusatzspeicher nicht nur für die Grafikkarte

Selbstbau-Plotter (Teil 1) 45
Hoffentlich überfordern wir nicht Ihr mechanisches Geschick. Das Ergebnis kann sich sehen lassen: Ein DIN-A3-Vierfarbplotter zu einem sensationell niedrigen Preis

FORTH - komfortabel 49
Das FORTH für den EC-65 hat eine Menge dazugelernt - jetzt ist es erst richtig interessant

Das UCSD-p-System 55
Die Vorteile dieses modernen Betriebssystems werden in dieser umfassenden Einführung dargestellt

Programmieren unter CP/M 67
Die Assembler-Programmierung unter CP/M ist für viele Leser ein interessantes Thema - unter anderem für die Big-Board-II-Nachbauer und ab diesem Heft auch die EC-65-Nachbauer

CP/M auf dem EC-65 (Teil 2) 74
Hier geht es um die Systemsoftware unserer neuen Coprozessor-Karte

CP/M auf dem EC-65 (Teil 3) 79
Geheimnisse gibt es nicht in EC-Computern: Das Software-Know-How zur Z80-Karte mit vielen Seiten Source-Listing

EC-65: CP/M-Software 90
Etliche Free-Soft-Disketten gibts zur Z80-Karte als Draufgabe, damit Sie die Karte auch gleich richtig ausnützen können

Centronics-Adapter 94
So kann der EC-65 Daten von einem beliebigen Computer empfangen

FLEX: Datenbanksysteme 96
Die Nützlichkeit eines Computers steht und fällt mit der verfügbaren Anwender-Softbare - und da braucht sich unser EC-68 wahrlich nicht zu verstecken

EC-65K / 65816: Software 100
Ein Top-Betriebssystem für den EC-65K wird es in Kürze geben. Auch ansonsten haben wir zum Thema "Software für den EC-65K" einiges mitzuteilen

Listing-Stopper 104
Eine praktische Arbeitshilfe für den EC-65

C64: Software-Maus 105
Neben der Hardware-Lösung auch ein "weiches" Maus-Interface. C64-Besitzer brauchen nur das Listing eintippen - für die Benutzer anderer Computer gibt es Flußdiagramme

Rubriken

Editorial 6

Korrekturen und Ergänzungen 104

Markt 109

Büchermarkt 110

Kontakte / Clubs / Infos 111

EPS-Service 113

Inserentenverzeichnis 114

Impressum 116

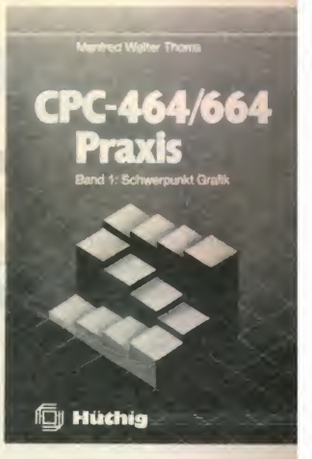
Die letzte Seite 116

Computerbücher
die gehen, für Computer die kommen.

Manfred Walter Thoma
Brücke zum Commodore 64
Ein erweitertes Handbuch
1985, 277 S., kart., DM 46,—
ISBN 3-7785-1095-9

Hajo Lemcke
Assembler mit dem C-64 (C-128)
1985, 227 S., 22 Abb., 6 Tab., kart., DM 38,—
ISBN 3-7785-1091-6

Thomas Tai
Computer-Abenteuer
Das Adventurebuch für den C-64
1985, ca. 120 S., kart., DM 29,—
ISBN 3-7785-1132-7



Manfred Thoma
CPC 464 /664-Praxis
Band 1: Schwerpunkt Grafik
1985, 188 S., zahlr. Abb., kart., DM 34,—
ISBN 3-7785-1149-1

Jochen Reuther, Jürgen Kehrel
Simulator 6510
Debugger und Simulator für den Commodore 64
1985, Diskette und Manual, DM 78,—
ISBN 3-7785-1165-3

Frank Bühler
Applesoft Basic
Tips und Tricks
1985, 241 S., 40 Abb., kart., DM 38,—
ISBN 3-7785-1094-0

Boris Allan
Sinclair QL-Begleiter
1985, 133 S., 30 Abb., 13 Tab., kart., DM 35,—
ISBN 3-7785-1101-1

Wolfgang Hagenmüller
Programmierkurs mit Microsoft-BASIC
Band 1: Grundlagen
1985, 188 S., kart., DM 38,—
ISBN 3-7785-1038-X

Weitere Titel und Informationen finden Sie in unserem Computerbuch-Katalog:
Dr. Alfred Hüthig Verlag, Postfach 10 28 69, 6900 Heidelberg 1

Hüthig

KNOW-HOW

Interface-Techniken und Ergonomie

Mensch und Computer - zwei Welten treffen aufeinander!

Bernd Fasel

Die rasante Entwicklung der Computertechnologie hat dazu geführt, daß viele Menschen gewollt oder ungewollt an ihrem täglichen Arbeitsplatz mit Computersystemen konfrontiert werden. Die daraus resultierende Problematik beschäftigt schon seit vielen Jahren ein Heer von Psychologen und soll nicht näherer Gegenstand dieses Artikels sein. Parallel dazu haben die Mikroprozessoren so manches Wohnzimmer erobert, was genau wie im Beruf zu einer Arbeitsplatzproblematik im privaten Bereich führt. Das Zauberwort in diesem Zusammenhang heißt Ergonomie.

Die Ergonomie ist die Wissenschaft der menschengerechten Arbeitsgestaltung — so ihre einfachste Definition. Hier fangen die Probleme auch schon an: Menschen sind nun einmal Individuen, die sich nur höchst unwillig in Schächtelchen in Form von Statistiken zwängen lassen. Daher sind alle Versuche, einen "Europa-Einheits-Menschen" zu definieren, gescheitert. Jeder, der schon einmal versucht hat, sich an einem Samstagvormittag neu einzukleiden, wird dies bestätigen. Warum ist es aber überhaupt wichtig, sich mit der Ergonomie seines Personalcomputers zu beschäftigen?

Viele menschlische Eigenschaften wie Charackter, Gewissen oder das Selbstverständnis werden entscheidend geprägt durch den Kontakt mit der Umwelt. Wenn diese Umwelt zu einem mehr oder weniger großen Anteil aus Computern besteht, eben auch durch Computer. Ferner wird die geistige Leistungsfähigkeit des Menschen durch die Kommunikation mit der Umwelt direkt geschult, das heißt, es kann durch einseitige Belastung bestimmter physischer Fähigkeiten zu einem Verlust auf anderen Gebieten kommen. Zum Beispiel haben Studien ergeben, daß an vielen Computerarbeitsplätzen vorwiegend das Kurzzeitgedächtnis beansprucht, dagegen das Langzeitgedächtnis kaum geschult wird. Dadurch entsteht die Gefahr, daß im Laufe der Zeit das Langzeitgedächtnis zurückbleibt, also ein Verlust an geistiger Leistungsfähigkeit entsteht.

1

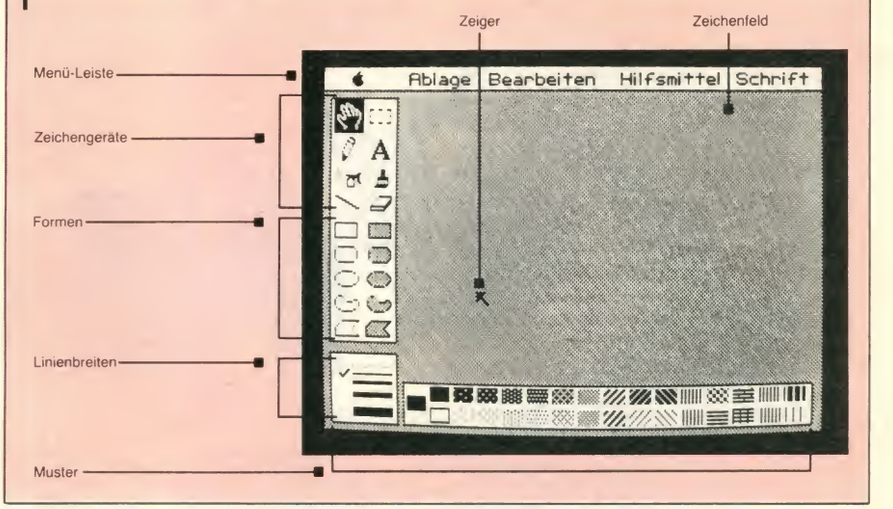


Bild 1. Ein gutes Beispiel für einen ergonomisch aufgeteilten Bildschirm liefert hier der Apple IIc. In Verbindung mit anderen, im Verlauf dieses Artikels geschilderten Interface-Techniken läßt sich eine komfortable Bedienung des Computers realisieren.

Die Frage lautet daher: Worauf sollte man beim Umgang mit Computern grundsätzlich achten? Dabei ist zu unterscheiden zwischen der Softwareergonomie und der Hardwareergonomie.

Die Softwareergonomie

Die Softwareergonomie beschäftigt sich mit dem Dialog des Menschen mit Programmen. Hier gibt es eine ganze Reihe von kritischen Fragen:

- Wieviel Information wird auf dem Bildschirm angeboten?
- Wie wird diese Information gruppiert (Einsatz von Grafik und Farbe)?
- In welcher Sprache erfolgt der Informationsaustausch?
- Ist der logische Ablauf eines Programmes für den Benutzer nachvollziehbar?

Wird zuviel Information auf dem Bildschirm dargestellt, frustriert das lange Suchen nach dem eigentlich Wichtigen. Solche Suchphasen können entscheidend verkürzt werden, wenn die Informationsdarstellung geschickt strukturiert wird. Zu diesem Zweck kann es sinnvoll sein, mit diversen grafischen Tricks den Bildschirm in Teilbereiche zu unterteilen und/oder bestimmte Informationen farbig hervorzuheben. Solche Techniken sind auf beeindruckende Art und Weise bei Apples "Mac" und "artverwandten" realisiert worden. Bild 1 zeigt am Beispiel des Apple IIc, wie ein ergonomisch aufgeteilter Bildschirm aussehen kann.

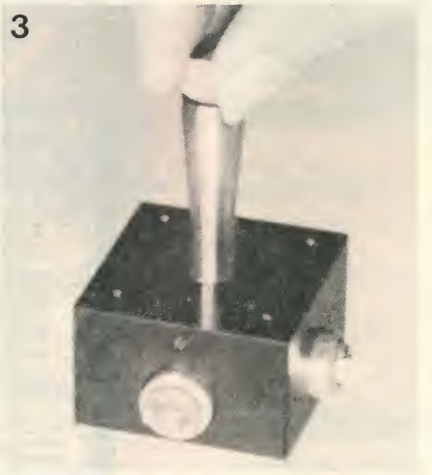
Der Unterpunkt "Sprache der Information" mag sich zunächst etwas merkwürdig anhören. Jeder wird sagen: "Bei einem gut angepaßten Anwenderprogramm muß der Benutzerdialog natürlich in Deutsch sein. Natürlich, zum Beispiel

bei einem Assembler...". Doch leider hat sich nicht nur bei den "Computer-Freaks", sondern auch bei vielen Firmen im Computerbereich ein Englisch-Deutsch-Esperanto eingebürgert, das für den "Normalverbraucher" absolut unverständlich ist. Da wird der RAM-floppy-driver debuggt, indem der Hexdump gepatch wird und dergleichen mehr (Anmerkung: Autor und Redaktion wissen, daß sich "debug" nur mit einem "g" schreibt!). Oft wäre eine gute englische Anleitung besser...

Aber das ist nicht das einzige Sprachproblem. Nach wie vor haben viele Personalcomputer Schwierigkeiten, deutsche Umlaute darzustellen. Es ist nun einmal umständlicher, "Buerogeraet" zu lesen als "Bürogerät". Dann sind da noch die Zahlen: Aus guten Grund wird "maschinennah" in Computersystemen mit binären oder hexadezimalen Zahlen gearbeitet. Aber muß der Dialog mit Programmen auch mit hexadezimalen Zahlen erfolgen? Hand aufs Herz: Wem macht es schon Spaß, $A772 von $FFC758 zu subtrahieren?

Ein weiterer wichtiger Punkt: Ist der logische Aufbau des Programmes für den Benutzer einsichtig und nachvollziehbar? Treten hier Schwächen auf, löst das meist spontane Frust-Reaktionen aus wie "Was ist denn jetzt schon wieder los" oder einfach "Hä?". Daher wurden Modelle für die Strukturierung von Programmen entwikkelt, die sich an die gewohnte Denksystematik des Menschen anlehnen. Ein solches Modell ist die sogenannte Kommando-Baumstruktur. Das funktioniert wie folgt: Dem Benutzer werden erst einmal ganz grob die verschiedenen Funktionen des Programmes angeboten. Nachdem eine Funktion ausgewählt wurde, wird in weiteren Schritten immer detaillierter ermittelt, was die genauen Wünsche des Benutzers sind - bis hin zur Ausführung. (Anm. der Redaktion: Dies wurde beim UCSD-p-System schon vor Jahren realisiert, siehe auch den Artikel im Softwareteil dieses Heftes).

Zu guter Letzt: Bis jetzt wurde vorwiegend von der Überforderung des Benutzers gesprochen. Die Unterforderung stellt aber ein genauso wichtiges Problem dar! Viele Programme (indirekt also deren Programmierer) trauen dem Benutzer gerade eben noch das Lesen von ein paar Zeilen Text auf dem Bildschrim zu. Nach dem Motto: "Legen Sie jetzt die Diskette ein. Disketten erkennen Sie daran, daß sie schwarz, flach und viereckig sind". Dabei wird vergessen, daß sich jeder Mensch durch solch eine pseudo-ergonomische Benutzerführung verschaukelt fühlt und mit Ablehnung reagiert. Dieser Effekt wird im Psychologen-Chinesisch "Dequalifikationsproblem" genannt. Treten in Verbindung mit solch einer naiven Benutzerführung auch noch längere Wartezeiten in der Programmausführung auf, kann jeder Programmierer sicher sein, daß seine Arbeit auf Ablehnung stoßen wird. Überhaupt spielen die Wartezeiten, in denen der Computer mit dem Programmlauf beschäftigt ist und sonst weiter nichts passiert, eine entscheidende Rolle. Eine genaue Frustrationsgrenze läßt sich nicht angeben, da diese stark von der gesamten Art der Benutzerführung abhängt. Jedoch sollte die Abarbeitung auch komplexerer Operationen im Sekundenbereich erfolgen.



3

4

Bild 2. Hier erkennt man eine moderne, flache Tastatur mit ergonomisch geformten Tastenkappen und farbiger Kennzeichnung wichtiger Tasten.

Bild 3. Ein einfacher Joystick mit Schaltern. Nachteil: Da ein Schalter nur offen oder geschlossen sein kann, ist der Bedienungskomfort arg eingeschränkt.

Bild 4. Komfortabler als ein Joystick mit Schaltern ist ein Potentiometer-Joystick. Dies schlägt sich aufgrund des komplizierteren Aufbaus allerdings im Preis nieder.

Hardwareergonomie

Hardware ist bekanntlich grob gesagt alles, was man anfassen kann. Im Zusammenhang mit dem Thema Ergonomie meint Hardware allerdings die gesamte räumliche Umgebung, in der mit einem Computer gearbeitet wird, und die Werkzeuge, die der Mensch zur Benutzung des Computers einsetzt.

Die Umgebung des Arbeitsplatzes Computer besteht im wesentlichen aus einem geeigneten Bürostuhl, einer Arbeitsplatte und vielleicht einem Regal für die verschiedenen Geräte. "Das haben wir natürlich alles zuhause", werden die meisten sagen. Aber Hand aufs Herz: Haben Sie nicht auch schon mal nach längerer Arbeit am Computer Kreuzschmerzen gehabt, weil der Stuhl, auf dem Sie sitzen, unbequem ist und zu einer ungünstigen Sitzhaltung verführt? Eine Arbeitshaltung mit gekrümmtem Rücken führt über kurz oder lang zu Rückenschmerzen - diese alte Erfahrung ganzer Generationen von Sekretärinnen macht heute eine Vielzahl von Computeranwendern. Oder haben Sie nicht öfters schon einmal Genickschmerzen und Verspannungen gehabt, weil Sie dauernd den Kopf bewegen müssen, immer hin und her zwischen Tastatur und Bildschirm? Steht Ihr Monitor vielleicht doch zu hoch, auch wenn er so besser ins Regal paßt?

Haben Ihnen nach stundenlanger Arbeit am Computer vielleicht schon einmal die Augen wehgetan? Vielleicht ist der Kontrast des Bildes auf dem Monitor zu stark, oder die Farben sind zu grell. Oder ist der Abstand zum Monitor zu klein? Und was ist mit der Geräuschbelästigung durch den Computer? Schon mal insgeheim ein "Gott sei Dank" gen Himmel geschickt, wenn nach fünf Stunden Arbeit endlich sämtliche Lüfter Ihres Personalcomputers zum Stillstand kommen und sich Ruhe einstellt? Sollte sich da nicht für den einen oder anderen Hersteller die Frage stellen, ob sich durch einen temperaturgesteuerten Lüfter nicht eventuell ein Marktvorteil erzielen läßt? Und zum Standard wird so etwas dann schnell. Und bei Selbstbausystemen läßt sich ein Lüfter häufig genug entbehren. Man muß sich bloß einen entsprechenden mechanischen Aufbau einfallen lassen, zum Beispiel alle Karten senkrecht stehend in einem größeren Abstand montieren, damit ein "Kamin"-Effekt entsteht. Dadurch werden die Gehäuse meist größer und unansehnlicher, aber was soll's? Ein Computer ist schließlich kein normaler Einrichtungsgegenstand, und vielleicht hat die "Kiste" noch irgendwie unterm Tisch Platz.

Dies sind natürlich alles relativ vage Vorschläge. Aber wie zu Anfang schon gesagt: Menschen haben ihre individuellen Eigenschaften, und jeder muß an sich erkunden, wie sein Arbeitsplatz am Computer aussehen sollte. Man muß den genannten Punkten erst einmal Aufmerksamkeit schenken, anfangen zu experimentieren und sich selbst dabei beobachten.

Werkzeuge

Der andere Aspekt der Hardwareergonomie sind die Werkzeuge zur Bedienung des Computers, weniger verständlich auch "Interaktionselemente" genannt. Und damit kommen wir zu den in der Überschrift genannten Interface-Techniken.

Hier muß natürlich zuerst die Tastatur genannt werden. Gerade im Punkt Computer-Tastaturen hat die Ergonomie-Diskussion die ersten wichtigen Erfolge gehabt. Die ersten Tastaturen ähnelten nämlich eher Zigarrenkisten mit ein paar Tastern darauf. Das führte zwangsweise zu einer unnatürlichen und anstrengenden Haltung beim Tippen. Die Folge waren häufige Erkrankungen im Bereich der Handgelenke, wie zum Beispiel Sehnenscheiden-Entzündungen. Wie entscheidend sich die Ergonomie-Forschung hier ausgewirkt hat, sieht man an den heutigen Tastaturen: Bild 2 zeigt ein typisches Beispiel.
Die Tastaturfläche ist nach vorne geneigt, die Tastenkappen sind ergonomisch geformt und die Übersichtlichkeit wurde durch eine spezielle Anordnung der Tasten und geeignete Farbgebung verbessert.

Es gibt natürlich eine ganze Reihe von Ein/Ausgabegeräten, die alle auf irgendeine Art und Weise dazu dienen, Informationen mit dem Computer auszutauschen. Diese Informationen können als Befehle zur Ausführung von Benutzerprogrammen führen oder Symbole auf dem Bildschirm bewegen. Die heute gebräuchlichen Bedienungselemente nutzen so ungefähr alle denkbaren Möglichkeiten, um die mechanischen Größen wie Wegstrecke, Geschindigkeit, Druck etc. in Form von elektrischen Größen an den Computer weiterzugeben.

Joysticks haben meist einen sehr einfachen Aufbau. Bild 3 zeigt einen solchen mechanisch einfachen Steuerknüppel. Für jede der vier Bewegungsrichtungen ist je ein Schalter eingebaut. Wird der Joystick in eine bestimmte Richtung bewegt, schließen ein bzw. zwei dieser Schalter,

5

6

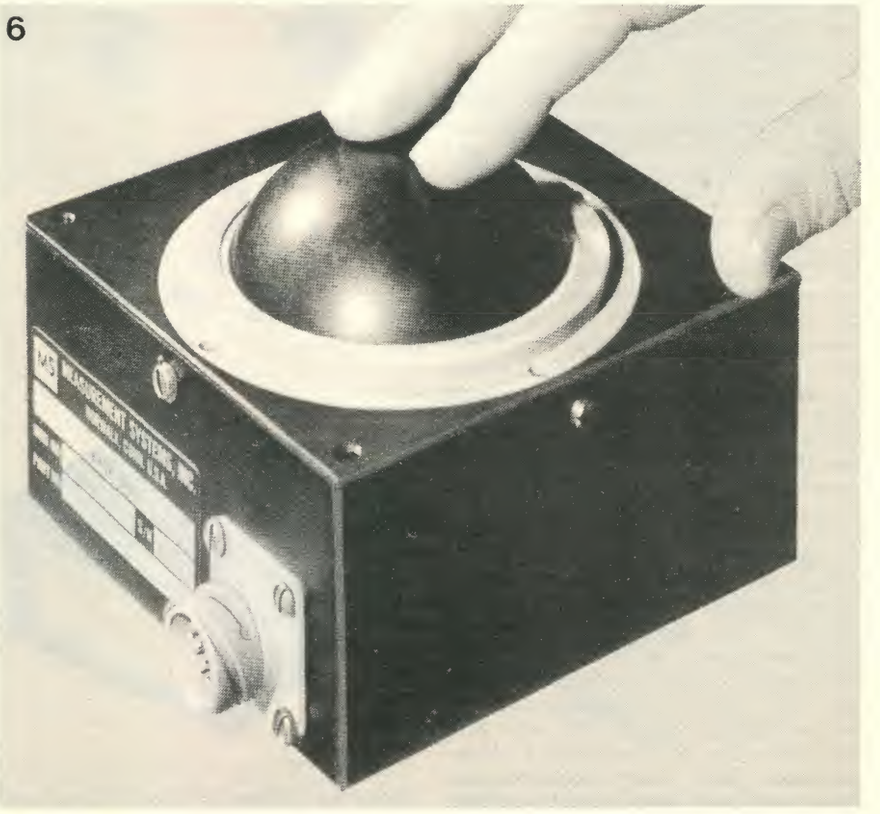


Bild 5. Eine recht komfortable und inzwischen weit verbreitete Möglichkeit zum Steuern von Computerfunktionen bietet die Maus, die bei manchen modernen Computern gleich zu Grundausstattung gehört.

Bild 6. Das Maus-Prinzip auf den Kopf gedreht: Der Trackball. Sein wesentlicher Vorteil: Zu seiner Verwendung benötigt man keine größere Arbeitsfläche.

was zum Beispiel dazu führt, daß auf dem Bildschirm ein Cursorsymbol in die entsprechende Richtung wandert. Dieses einfache Prinzip hat natürlich entscheidende Nachteile: Rein gefühlsmäßig wird meist ein Zusammenhang zwischen dem Druck auf den Steuerknüppel und der Bewegungsgeschindigkeit erwartet. Also fester Druck gleich hohe Bewegungsgeschwindigkeit. Genau das leistet ein normaler Joystick nicht. Wer sich schon einmal mit Computerspielen beschäftigt hat, wird diese Eigenschaft bestimmt schon verwünscht haben. Man kann diesen Effekt durch einen einfachen Trick etwas entschärfen: Indem sich nämlich der Cursor immer schneller bewegt, je länger man auf den Joystick drückt. Prinzipiell kann diese Art von Joysticks aber nur Bewegungsinformationen von 45° weitergeben, was einen weiteren entscheidenden Nachteil darstellt. Modelle mit eingebauten Potentiometern anstelle der Schalter kennen diese Probleme nicht. Die aufwendigere Konstruktion mit den Potis schlägt sich allerdings auch direkt in einem höheren Preis nieder. Auf dem Foto Bild 4 erkennt man einen solchen Joystick mit Potis.

Eine bessere Lösung bietet hier die Maus (Bild 5). Sie besteht aus einem kleinen Kästchen mit einem Gummiball an der Unterseite. Die Bewegungen des Gummiballs werden über zwei mitlaufende Achsen, an denen Lochscheiben und Lichtschranken befestigt sind, in elektrische Impulse umgewandelt. Dadurch lassen sich alle zweidimensionalen Bewegungen erfassen, also auch das beliebige Herumfahren mit der Maus auf der Tischplatte. Um die Bewegungsrichtung erkennen zu können, wird hier ein geschickter Trick angewandt: An jedem Lochscheibenrad sind zwei Lichtschranken angebracht, die gegenüber dem Lochmuster verschoben positioniert sind. Dies ist einfacher zu verstehen, wenn man sich vorstellt, die Maus würde mit konstanter Geschwindigkeit bewegt. Dann würde entsprechend an den Ausgängen beider Lichtschranken eine konstante Frequenz anstehen. Die Anordnung der Lichtschranken ist so gewählt, daß die beiden Frequenzen je nach Bewegungsrichtung um +90° oder -90° gegeneinander phasenverschoben sind. So läßt sich jede Bewegung der Maus eindeutig rekonstruieren. Wie die Drehrichtungs-Erkennung besonders einfach realisiert werden kann, wird in dem Artikel "Mäuse. . ." an anderer Stelle in diesem Heft beschrieben.
Es gibt aber auch Computermäuse, die intern keine bewegte Mechanik haben. Sie funktionieren nur auf einer speziellen Unterlage, auf der ein Streifenmuster verschiedenfarbiger Linien aufgedruckt ist. Diese Linien werden durch Reflexion auch mittels Lichtschranken erkannt.

Der entscheidende Vorteil der Mäuse ist eindeutig die feste Beziehung zwischen der Bewegung der Mäuse auf dem Tisch und der Bewegung des zugeordneten Symbols auf dem Bildschirm. Ein solches Symbol läßt sich dadurch sehr schnell und gezielt positionieren. Es wird allerdings oft als Nachteil empfunden, daß man eine größere Bewegungsfreiheit auf dem Arbeitstisch braucht.

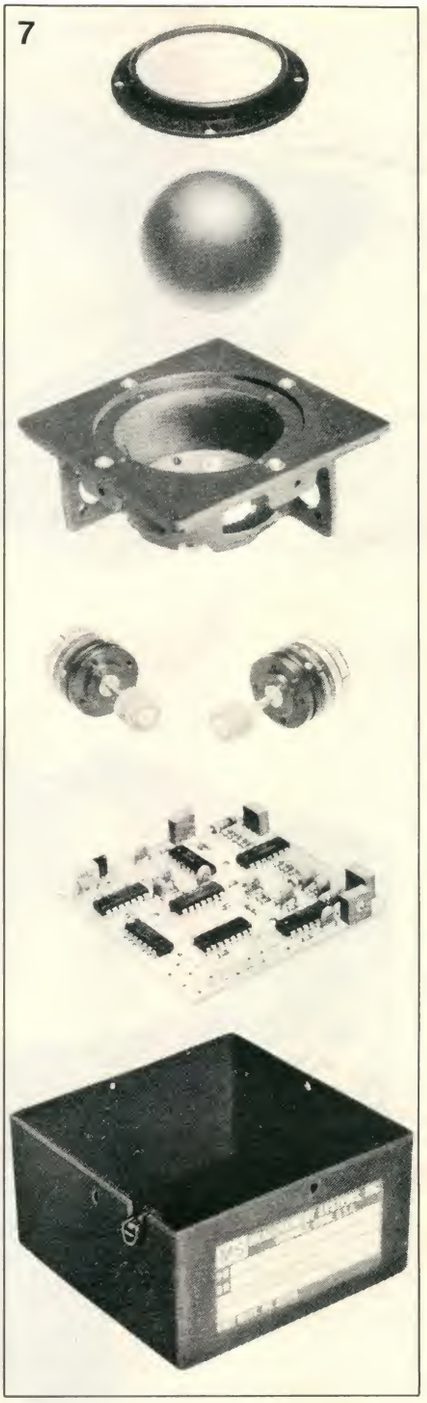


Bild 7. Hier ist die innere Konstruktion eines Trackballs deutlich zu erkennen. Auch die meisten Mäuse sind sehr ähnlich aufgebaut.

Bild 8. Noch einfacher geht es kaum noch: Beim "Touch screen" reicht die Berührung des Schirmes beziehungsweise das Zeigen auf eine Stelle des Schirmes aus, um bestimmte Funktionen abzurufen.

Diesen Nachteil hat ein Trackball nicht. Er läßt sich am einfachsten mit einer umgedrehten Maus vergleichen, wie auf dem Foto Bild 6 unschwer zu erkennen ist. Die Gummikugel ist hierbei meist wesentlich größer als bei einer Maus ausgeführt, um die Handhabung zu erleichtern. Im Prinzip funktioniert ein Trackball aber genauso wie eine Maus. Bild 7 veranschaulicht die Konstruktion, deutlich sind die beiden mitlaufenden Achsen für die Bewegungsrichtung zu erkennen.

Es gibt aber auch eine Reihe von Bedienungselementen, die weniger für das Bewegen von Symbolen geeignet sind, sondern in erster Linie für die Auswahl von Feldern oder Funktionen auf dem Bildschirm. Sie werden daher vor allem in Verbindung mit menügesteuerten Programmen eingesetzt. Am bekanntesten dürfte hier der "light pen" oder Lichtgriffel sein, ein Kugelschreiber-ähnliches Gerät, in dessen Spitze eine Fotodiode eingebaut ist. Die Auswahl von Funk-

tionen erfolgt einfach, indem mit dem Lichtgriffel auf die gewünschte Stelle auf dem Bildschirm getippt wird. Da das Bild kontinuierlich vom Elektronenstrahl einer Kathodenstrahlröhre neu geschrieben werden muß, läuft der Strahl bei jedem Bildaufbau auch an der lichtempfindlichen Fläche der Fotodiode vorbei. Dies bewirkt einen entsprechenden Ausgangstrom, der in Verbindung mit einigen unterstützenden Daten des Videokontrollers zu einer eindeutigen Identifikation der berührten Stelle auf dem Bildschirm führt. Es ist klar, das dieses Verfahren um so besser funktioniert, je kürzer die Nachleuchtdauer der Bildröhre ist. Bei besonders lang nachleuchtenden Bildschirmen läßt sich die momentane Position des Elektronenstrahls nicht mehr mit der geforderten Genauigkeit bestimmen. Einige der auf dem Markt befindlichen Video-Kontroller unterstützen von sich aus schon den Betrieb von Lichtgriffeln, z. B. der μPD 7220, der 6845 und der 9365/66/67.

Die verblüffenste Art, einen Computer zu bedienen, ist aber sicherlich, einfach mit dem Finger ein Menüfeld auf dem Bildschirm zu berühren und damit die gewünschte Funktion auszulösen. So etwas gibt es auch schon, wie Bild 8 zeigt, und zwar unter der schönen Bezeichnung "Touch screen" (berühre den Bildschirm). Da gibt es zum Einen das Verfahren des "Lichtvorhangs". Das heißt, rings um den Bildschirm wird eine große Zahl von Lichtschranken angeordnet. Wenn der Finger oder ein Gegenstand in diesen Lichtvorhang eintaucht, wird ein Teil der Lichtschranken abgedunkelt, wodurch auf den Ort der Berührung zurückgeschlossen wird. Die direkte Berührung der Glasoberfläche des Schirms ist hierbei nicht notwendig.

Bei einem anderen Verfahren wird vor dem Bildschirm eine Glasplatte angebracht, die auf sog. Kraftmeßdosen montiert ist. Ein Druck auf eine bestimmte Stelle der Glasplatte bewirkt eine entsprechende Verteilung der auf die Meßdosen wirkenden Kräfte, wodurch wiederum der Berührungspunkt ermittelt werden kann. Sicherlich eines der anspruchsvollsten Verfahren!

Zu guter Letzt: Das Grafiktablett. Wie in Bild 9 zu erkennen, besteht es aus einer Platte und einem mausähnlichen Gebilde. Die Position der Maus auf dem Grafiktablett wird kontinuierlich bestimmt und an den Computer weitergeleitet. Es ist also möglich, mit der Maus beliebige Kurven auf einer Vorlage nachzufahren und damit in den Computer einzulesen. Das würde der Funktion eines Digitalisierers entsprechen. Für diese Anwendung wird die Maus auch oft durch einen Stift ersetzt. Eine andere verbreitete Einsatzmöglichkeit zeigt Bild 10. Auf einer Vorlage wird eine große Anzahl von Menüfeldern eingezeichnet. Die zugehörigen Funktionen werden direkt aktiviert, indem man mit der Maus in das gewünschte Feld fährt und einen Taster drückt.

Die Grafiktabletts sind intern meist relativ kompliziert aufgabaut. Es werden häufig elektrostatische und elektromagnetische Verfahren benutzt, teilweise mit einer periodischen Vormagnetisierung und dergleichen physikalischen Tricks mehr. Andere Fabrikate ermitteln die Position der Grafik-Maus mit Ultraschallfeldern, in die Maus ist dann ein piezoelektrischer Sensor eingebaut. Es sind auch Low-Cost-Versionen auf dem Markt, die mit Widerstandsfolien arbeiten.

Wie man sieht, kann man mit etwas Phantasie und natürlich der entsprechenden Hard- und Software die Bedienung eines Computers sehr komfortabel gestalten. Die für den Menschen komfortabelste Methode wäre sicherlich die direkte Spracheingabe. Doch auf diesem Gebiet ist noch mancher (Forschungs-) Schritt zu tun...

9

10

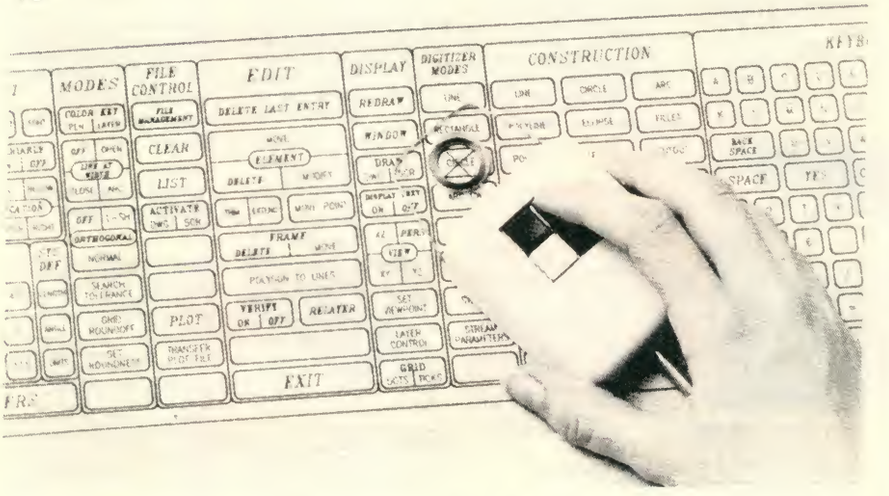


Bild 9. Zum Grafiktablett gehört meist eine spezielle Maus oder ein spezieller Stift. Hochwertige Grafiktabletts sind intern sehr kompliziert aufgebaut, einfachere Ausführungen begnügen sich mit einer Widerstandsfolie.

Bild 10. Eine spezielle Anwendung des Grafiktabletts: Statt eines Symbols auf dem Bildschirm wird ein Symbol auf dem Grafiktablett angefahren und dann ein Taster betätigt. Vorteil: Der Bildschirm bleibt für die eigentliche Anwendung (zum Beispiel zur Abbildung einer technischen Zeichnung) frei.

ELEKTOR

BÜCHER

ELEKTRONIK-BAUBUCH

Populäre Schaltungen für die Praxis

Der Schriftzug "Elektronische Schaltungen für die Praxis" auf der Titelseite verspricht nicht zu viel. Es sind populäre Schaltungen, veröffentlicht in der Zeitschrift Elektor, die in (Hobby-)Elektronikerkreisen auf große Resonanz gestoßen sind. Die Schaltungsauswahl geht querbeet durch alle Bereiche der Elektronik: Audio und Hifi, Auto, Haus und Hof, Empfänger, Hobby und Spiele, Messen und Testen, Musikelektronik sowie Stromversorgung und Netzteile.

Was gibt es im einzelnen:

* Zwei Equalizer-Entwürfe
* Leuchtstoffröhren-Dimmer
* Elektronische Orgel
* Endverstärker hoher Qualität
* Modelleisenbahn-Elektronik
* Kleinstradio
* Auto-Alarmanlage
* u.v.a.m.

ISBN 3-921608-39-2
256 Seiten - DM 29,50

300 Schaltungen

300 Schaltungsideen sind für den Elektronikinteressierten eine wahre Fundgrube. In ihr findet der kreative Leser Anregung zum Experimentieren und Kombinieren. Das Buch enthält interessante Anwendungsfälle aus nahezu allen Bereichen der Elektronik: Audio und Musik; Elektronik für Haus, Hof und Garten; Elektronik zum Testen und Messen, für Hobbys und fürs Spielen; Niederfrequentes, Digitales und Analoges.

ISBN 3-921608-09-0
270 Seiten DM 26,50

ne Quelle wichtiger Fakten und wertvoller Anregungen. Elektronik – gar nicht schwer, ein Buch von bleibendem Wert.

ISBN 3-921608-32-5
174 Seiten, 201 Abbildungen
DM 17,75

6502-MASCHINENCODE FÜR SCHNELLE GRAFIK

Maschinencode! Welcher Computerfreak träumt nicht davon, mit Hilfe der Maschinensprache alle Möglichkeiten seines Mikroprozessors voll auszuschöpfen. Das Buch sorgt dafür, daß es nicht nur ein schöner Traum bleibt. Es bringt die Maschinenbefehle "spielend" an die Frau/den Mann.

Die Autoren erläutern die wichtigsten 6502-Befehle anhand schneller, bewegter Grafik. Die einfachen Befehle stehen am Anfang und erleichtern den Übergang zum Verständnis der komplexeren Befehle. Wo es angebracht ist, ziehen die Autoren Parallein zwischen den Maschinenbefehlen und Situationen des täglichen Lebens. Eine Vielzahl von Demonstrationsprogrammen machen die Theorie recht abwechslungsreich. Das Maschinenprogramm "MASTERPAINT" einschließlich der Tabellen und Darstellungen erleichtert den Umgang mit der Blockgrafik erheblich.

Die abgedruckten Programme sind zum SAMSON-Computer von elektor kompatibel, laufen aber, leicht abgeändert, auch auf jeder anderen 6502-Maschine.

Von K. Diedrich und G. Klein

136 Seiten ISBN 3-921608-40-6
DM 23,-

302 SCHALTUNGEN

Die begehrte "Dreihunderter-Reihe" bietet Ihnen mit **302 Schaltungen** 302 neue und praktische Anwendungsbeispiele aus dem gesamten Bereich der Elektronik. Auf 368 Seiten finden Sie Schaltungen zu den Themen Audio, Video und Musik, Auto, Fahrrad und Motorrad, Haus und Hof, HF, Empfänger und Antenne, Hobbys und Spiele, interessante Bauelemente, Messen und Testen, Mikrocomputer, Musikelektronik, Oszillatoren und Generatoren, Stromversorgung und Netzgeräte und zu Themen, die sich nicht katalogisieren lassen. Schaltungen, die Sie schon lange gesucht haben und die Sie für Ihre Problemlösung direkt so übernehmen können; aber auch Schaltungen, die Sie auf ganz neue Ideen bringen.

ISBN 3-921608-37-6
368 Seiten - DM 31,–

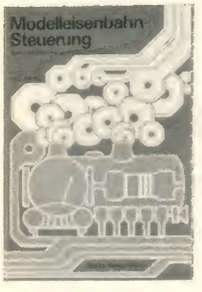
BASICODE

Basicode ist im Prinzip ein verkürztes Basic, das mit Befehlssatz arbeitet, den diese Rechner verstehen. Um den Computer für Basicode zu formatieren, enthält die beigelegte Kassette alle Übersetzungsroutinen für die unten aufgeführten Rechner. Alle 14 Programme sind ebenfalls auf der Kassette abgespeichert, womit das muhsame Eintippen entfällt. Zur Sicherheit sind aber alle Listings nochmals im Buch abgedruckt. Basicode wird in Deutschland vom WDR abgestrahlt, in Kürze auch vom SWF und ORF.

Die Programme laufen auf folgende Rechner:

Apple 2, BBC A + B, Colour Genie, Commodore VC 20/C 64, 3008, 3016, 3032, 4016, 4032, 8032, 8096, DAI, Dragon, Exedy, Sourcere Junior Computer (elektor), P 2000, PET 2001, Sharp MZ 80A und MZ 80 k, TRS 80, Modell I und III, Videogenie I und II, ZX 81, ZX Spectrum.

ISBN 3-473-44010-8
DM 58,– (einschl. Kassette)

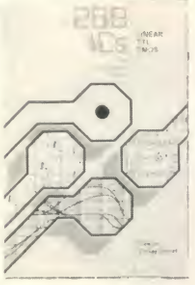


MODELLEISENBAHN-STEUERUNG

Automatisieren mit und ohne Computer
Paul De Bra

"Fahrdienstenleiter Computer" – das ist auf einen kurzen Nenner gebracht das Ziel dieses Buches. Es beginnt mit dem elektronischen Gegenstück zum bekannten Stelltransformator, einer digitalen Steuerung der Zuggeschwindigkeit. Weiter geht's mit der elektronischen Weichen- und Signalsteuerung, gefolgt von einem Blocksicherungssystem. Mit diesem Buch schaffen Sie die Koordinaten des gesamten Fahrbetriebs.

ISBN 3-921608-36-8
ca. 150 Seiten
DM 25,-



269 ICs

LINEAR
TTL
CMOS
Die wichtigsten Daten.

Jetzt ist Schluß mit dem zeitraubenden Wälzen in diversen Datenbuchern. Das Elektor-Datenbuch bietet übersichtliche Informationen über 62 lineare ICs, 131 TTL-ICs und 76 CMOS-ICs. Zu jedem IC gibt es eine Kurzbeschreibung; Grenzdaten und Eigenschaften sowie elektrische Kenndaten sind in Tabellenform dargestellt. Anschlußbilder und Anwendungsbeispiele machen das Buch für alle Profi-, Hobby-, Berufs- und Amateur-Elektroniker unentbehrlich.

ISBN 3-921608-31-7
240 Seiten
DM 33,50

WEGE IN DIE ELEKTRONIK

Glagla/Lindner
288 Seiten

"Wege in die Elektronik" – eine zuverlässige und verständliche Einführung in die grundlegenden Zusammenhänge der Elektronik – auch für denjenigen, der sich bisher noch nicht mit dieser Materie beschäftigt hat.

Das Buch zeigt nicht nur, wie sich Schaltungen rechnerisch und experimentell dimensionieren lassen, es stellt auch die handwerkliche Seite des Umgangs mit Elektronik ausführlich dar und unterbreitet über 60 reizvolle und brauchbare Anwendungsvorschläge, die reich illustriert sind.

ISBN 3-473426-26-1
DM 34,–

GRUNDWISSEN INFORMATIONS-VERARBEITUNG

Von Helmut Schiro, 311 S., zahlreiche Bilder und Grafiken, durchgehend 2farbig, kart., DM 48,-

Ausgehend von der Entstehung der EDV bis hin zu den modernsten Kommunikationstechniken, wie Bildschirmtext, Datennetzstrukturen und Telekommunikation, behandelt der Autor auf sehr verständliche und anschauliche Weise alle wesentlichen Aspekte: Zweck und Anwendung der Datenverarbeitung, Struktur und Aufbau von Datenverarbeitungssystemen, Bedeutung von Betriebssystemen, Betriebsarten der Datenverarbeitung, Informationssysteme, Bürokommunikation, Bildschirmtext, Sprachspeicher und Vermittlungssysteme, Datenkommunikation, Expertensysteme.

ISBN 3-8068-4314-7

SO MACHEN SIE MEHR AUS IHRER STEREO-ANLAGE

Was ist Hi-Fi, was heißt Sinusleistung/Musikleistung, was dB, warum klang die Anlage beim Kauf so phantastisch und zu Hause nur noch halb so gut? Was kann man mit einfachen Mitteln tun, um mehr "Sound" aus seiner Anlage herauszuholen? Diesen Fragen geht der Autor in übersichtlich gegliederten Kapiteln nach und zeigt in allgemeinverständlicher Weise, wie man vielen kleinen Problemen abhelfen kann. Neben den notwendigen Grundinformationen empfiehlt er eine ganze Reihe kleiner praktischer Schaltungen, vom Kabelverstärker bis hin zu einem Mischpult, die alle ohne allzu großen Aufwand auch von einem Anfänger nachgebaut werden können. Alle Schaltungen wurden im Elektor-Labor geprüft, und sind teilweise auf elex-Standardplatinen aufgebaut.

ISBN 3-473-44009-4
DM 24,–

UMWELT ELEKTRONISCH MESSEN

Michael Wiegand

Die in diesem Buch vorgestellten Meßgeräte sind für den Hobbyelektroniker eine kleine Sensation: der Spareffekt schlägt voll zu Buche. Z. Teil sind diese Geräte nur als teure Industriemeßgeräte bzw. nur im Selbstbau realisierbar. Dieses Buch ermöglicht Ihnen den preiswerten Nachbau von sechs hochwertigen präzisen Meßgeräten, womit Sie Wasserqualität, Sauerstoff, Chlor, Radioaktivität und meteorologische Daten messen können:

- Ph-Meßgerät mit Temperaturmessung
- Leitwert-Meßgerat mit Redox-Meßbereich
- Sauerstoff-Meßgerät
- Geiger-Müller-Zähler (Radioaktivitat)
- Eine komplette Wetterstation
- Schallpegelmesser

Mit den einleitenden Kapiteln über Werkzeug, Material und Arbeitsweise, den Platinenfolien und den getesteten Schaltungen ist der Nachbau schon für den Anfänger ohne weiteres möglich. Die Schaltungen werden ausführlich in Wort und Bild beschrieben, Fotos der Platinen mit den Bauelementen zeigen deutlich die Zuordnung, und die übersichtlichen Bauteilelisten informieren schnell über das, was man braucht. Checklisten für Aufbau und Fehlersuche helfen, damit nichts "schiefgeht".

Der Clou ist, daß die meisten Geräte noch nicht im elektor erschienen sind.

Sowohl der Elektronik-Fortgeschrittene als auch der Anfänger werden nun der Umwelt "auf den Zahn fühlen" können! Machen Sie mit!

ISBN 3-473-44004-3
Otto-Maier-Verlag
DM 24,–

ELEKTRONIK GAR NICHT SCHWER

Buch 1: Experimente mit Gleichstrom
A. Schommers

Elektronik – gar nicht schwer ist ein Freizeitbuch für Technikinteressierte jeden Alters. Vorkenntnisse sind nicht nötig, denn es geht gleich los mit kleinen Experimenten. Einfach und schnell aufgebaut, machen sie Elektronik greifbar und somit begreifbar. Der Autor führt seine Leser Schritt für Schritt durch die Welt der elektronischen Bauelemente und der Gleichstromtechnik. Die Datensammlung im Anhang ist auch für die weitere Beschäftigung mit der Elektronik ei-

Elektor Verlag GmbH
Süsterfeldstraße 25, 5100 Aachen

Bestellen Sie mit der Karte am Heftende!

V30: "Kompatible" CPU (Teil 1)

Ein 16-Bitter, der nicht nur IBM-Programme, sondern auch noch CP/M-80 Software "schluckt"

Frank Schmidt

Der V30 ist ein CMOS-Mikroprozessor von NEC, der einerseits aufwärtskompatibel zum 8086 von Intel ist, andererseits aber auch den 8080 emulieren kann. Und ein Geschwindigkeitsvorteil gegenüber dem 8086 ist auch noch drin! Der V30 erlaubt es damit, echte 16-Bit-Power zu nutzen und gleichzeitig nicht auf das Riesenangebot an "alter" CP/M-Software verzichten zu müssen. Und eine 8088-kompatible 8-Bit Version gibt's auch noch!

Gleich zu Anfang eine kleine Bemerkung, die helfen soll, Mißverständnissen vorzubeugen: Dieser Beitrag soll die NEC-CPU V30 möglichst detailliert beschreiben. Andererseits gilt aber ein großer Teil dessen, was hier gesagt wird, auch für den 8086. Da nun aber NEC für seine CPU andere Bezeichnungen gewählt hat, als Intel (z.B. für Anschlußbelegungen, Register, Mnemonics), andererseits aber die Intel-Bezeichnungen schon recht weit verbreitet sind, haben wir folgende Konvention beschlossen: Soweit möglich, werden für Register und Mnemonics die Intel-Bezeichnungen übernommen. Bei Mnemonics, die nur vom V30 erkannt werden, benutzen wir natürlich die NEC-Bezeichnungen. Und weil uns die Pin-Bezeichnungen von NEC einleuchtender erschienen, ziehen wir diese wiederum denen von Intel vor...

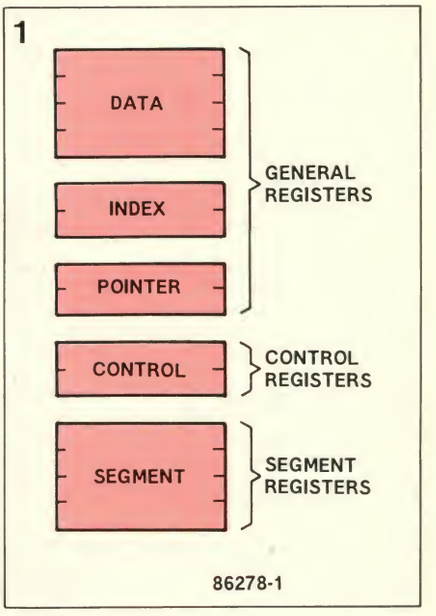
Familieangelegenheiten

Eine der ersten CPUs mit echtem 16-Bit Datenbus war der 8086. Er zeichnet sich durch einen großen Vorrat an leistungsfähigen Befehlen und Adressierungsarten sowie durch einen hohen Datendurchsatz aus, der nicht nur auf dem Rücken von Speicherzugriffszeiten getragen wird. Grund dafür ist das sogenannte Pipelining, das später noch eingehender besprochen wird. Ein weiterer Pluspunkt für den Intel-Prozessor ist es auch, daß er einen Speicherbereich von 1 MByte (1024 KByte) ohne weitere Hardware adressieren kann — für damalige Verhältnisse (und Speicherpreise) eine enorme Menge. Und auch heute für die meisten Anwendungen mehr als genug.
Dennoch hat auch Intel erkannt, daß sich ein System mit 8-Bit Datenbus doch einfacher und kostengünstiger aufbauen läßt, als eines mit 16-Bit Datenbus. Aus dieser Überlegung ist der 8088 entstanden: Er ist vollkommen Software-kompatibel zum 8086 und unterscheidet sich von ihm im Wesentlichen nur durch die Busstruktur. Aber dennoch: Der 8-Bitter ist wesentlich schneller als vergleichbare Prozessoren wie z.B. Z80 oder 6809. Auch lassen sich Programme für ihn leichter und in kürzerer Zeit schreiben: Schließlich besitzt er ja einen wesentlich umfangreicheren Befehlssatz.
Das was auch der Grund für IBM, seinen PC mit diesem Processor zu bestücken.
Weitere Entwicklungen von Intel waren die Prozessoren 80188/86 und 80286.

Sie unterscheiden sich von den beiden erstgenannten im Wesentlichen durch die On-Board-Peripherie. Da sind dann z.B. Interrupt- und DMA-Controller gleich integriert. Aber auch die Abarbeitung der verschiedenen Befehle geschieht bei den "Hundertern" und "Zweihundertern" schneller.
Zusätzlich können alle genannten Prozessoren mit intelligenten Arithmetik-Chips zusammenarbeiten, die zur Zeit zwar noch ein Heidengeld kosten, dafür aber rechenintensive Jobs in sehr kurzer Zeit erledigen. Der 80286 in der 12,5 MHz-Version ist übrigens eine der schnellsten CPUs, die zur Zeit zu haben sind.
Der V30 von NEC (seine offizielle Bezeichnung ist mittlerweile μPD 70116) ist als direkter Ersatz für den 8086 entworfen worden. Die beiden Chips sind so kompatibel, daß man den Intel-Prozessor in einem Rechner normalerweise bedenkenlos gegen seinen Konkurrenten austauschen kann. Das Gleiche gilt für das Paar 8088/V20 (μPD 70108). V25, V40 und V50 verkörpern den NEC-Angriff auf 80188/80186, sind aber nicht mehr direkt mit ihnen austauschbar. Alle Prozessoren — V20 bis V50 — haben den selben Befehlssatz. Der umschließt alle Befehle von 8088/86, einige neue Befehle und eine vollständige 8080-Emulation.

Programmiermodell

Der V30 besitzt insgesamt 14 16-Bit-Register, auf die der Programmierer Zugriff hat. Sie unterteilen sich in vier Daten-, zwei Index-, zwei Zeiger-, zwei Steuer- und vier Segmentregister.
Die vier 16-Bit-Datenregister werden AX, BX, CX und DX genannt. Sie können auch als jeweils zwei 8-Bit-Register angesprochen werden. Ihre Bezeichnung lautet dann z.B. AH und AL (siehe Bild 2) für das höher- und das niederwertige Byte von AX.
Alle Datenregister können benutzt werden, um darin einfache arithmetische und logische Funktionen auszuführen. Im Gegensatz zu anderen Prozessoren muß man sich hierbei also nicht nur auf den Akkumulator beschränken. Darüberhinaus haben einige der Register noch Spezialfunktionen. So wird beispielsweise das Register BX für einige leistungsfähige Adressierungsarten benutzt und CX stellt den Zähler bei Wiederholungsbefehlen — wie String-Operationen — dar.



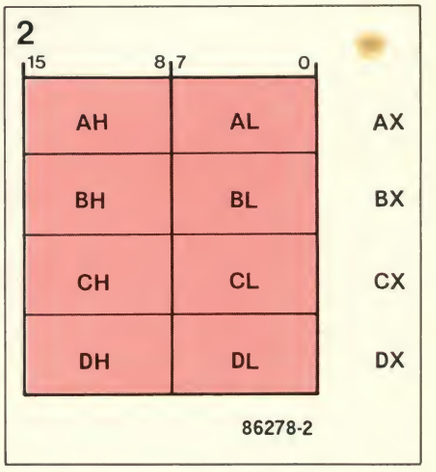


Bild 1. Im Native-Mode der V30-CPU stehen dem Programmierer insgesamt 14 16-Bit Register zur Verfügung.

Bild 2. Jedes der vier 16 Bit breiten Datenregister läßt sich auch in jeweils zwei 8-Bit-Register unterteilen.

Neben dem normalen Stackpointer SP steht beim V30 noch ein zusätzlicher — BP (Base Pointer) genannt — zur Verfügung. Diesen kann der Programmierer für seine eigenen Zwecke nutzen. Er kann unter anderem einen alten Stackpointer-Wert enthalten oder eine Stelle im normalen Stack markieren — unabhängig vom eigentlichen Stack-Pointer.

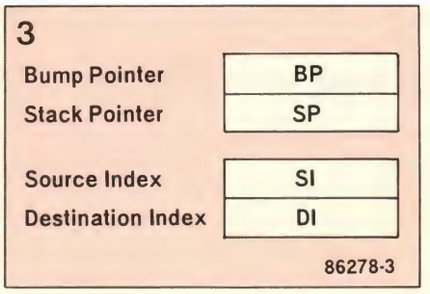


Bild 3. Im V30 sind gleich zwei Stackpointer — SP und BP — integriert. SP wird im Native-, BP im Emulationsmode benutzt. Darüberhinaus steht BP dem Programmierer im Native-Mode zur freien Verfügung.
Außerdem gibt es noch zwei Index-Register, die vor allem bei String-Befehlen benutzt werden und mit denen auch bestimmte Datenstrukturen aufgebaut werden können.

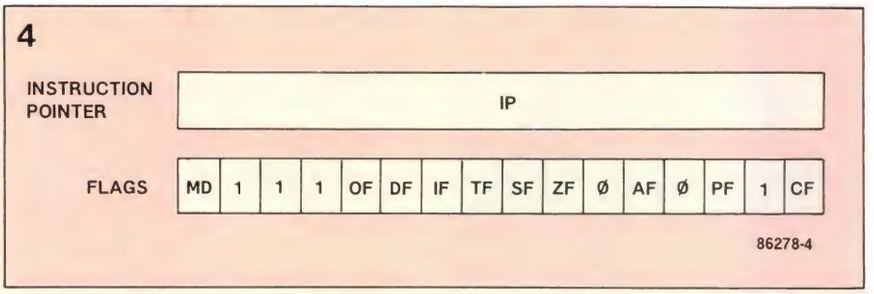


Bild 4 zeigt die beiden Steuerregister. Es fällt auf, daß der Programmzähler (hier "Instruction Pointer" (IP genannt) nur 16 Bit breit ist. Das liegt am Segmentierungskonzept der CPU, auf das im Text näher eingegangen wird.
Das Flag-Register des V30 stimmt fast vollkommen mit dem des 8086 überein. Der einzige Unterschied steckt im höchstwertigen Bit (MD): Ist dieses HIGH, so befindet sich der Prozessor im Native-, andernfalls im Emulations-Mode.

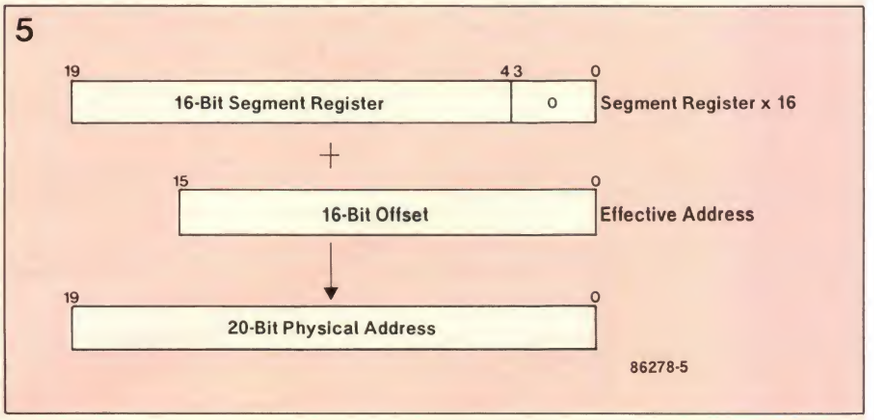


Bild 5. So werden die physikalischen 20-Bit Adressen des V30 gebildet: Zu der durch irgendein Register oder einen Offset definierten 16-Bit Adresse wird der Inhalt des korrespondierenden Segment-Registers addiert, der allerdings vorher um vier Bits nach links verschoben wurde. Das entspricht einer Multiplikation mit $2^4 = 16$. Falls sich verschiedene Segmente überlappen, muß man darauf achten, daß sich nicht auch die eigentlichen Datenbereiche überschneiden, sonst kommts zum Chaos!

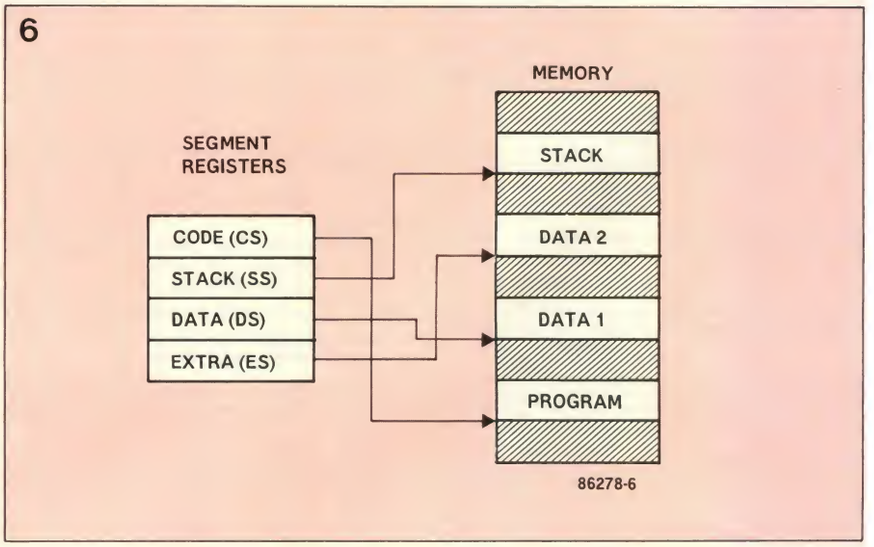


Bild 6. Insgesamt vier Segment-Register sind im V30 untergebracht. Jedes indiziert einen 64 KByte-Speicherbereich, auf den sich dann die effektiven Adressen beziehen. Damit lassen sich Programm-, Stack- und zwei verschiedene Datenbereiche auf einfache Weise voneinander trennen. Außerdem erlaubt diese Methode den gleichzeitigen Zugriff auf 256 KByte Speicher mit 16-Bit-Adressen.

Im Emulations-Modus des V30 wird BP außerdem als 8080-Stackpointer benutzt. Damit werden im Native- und im Emulationsmode verschiedene Stacks verwendet. Die beiden Index-Register SI (source index) und DI (destination index) werden vor allem bei String-Operationen sowie für ein paar besonders leistungsfähige Adressierungsarten genutzt.
Auch die beiden Steuerregister — der Programmzähler und das Flag- oder Statusregister — haben eine Länge von 16 Bit (s. Bild 4). Die Bedeutung der einzelnen Flags wird später erläutert werden.

Segmentierung

Um auf ein Megabyte Speicher zugreifen zu können, benötigt man Adressen mit einer Breite von 20 Bit. Diese werden mit Hilfe der Segment-Register erzeugt. Mit ihnen wird aus dem gesamten Speicherbereich jeweils eine Seite zu 64 KByte ausgewählt. Alle 16-Bit-Adressen (z.B. Programmzähler, Stackpointer etc.) beziehen sich nur auf dieses Segment. Die effektive Adresse berechnet sich aus dem 16-fachen des Segment-Registers, zu dem dann die entsprechende 16-Bit Adresse einfach addiert wird. Bild 5 zeigt ein Beispiel.
Um dieses Verfahren möglichst effektiv zu machen, besitzt der V30 nicht nur ein einziges Segmentregister, sondern gleich vier davon: Für jeden Zweck eines. So bildet das Code-Segment die Basisadresse für den Programmzähler. Das Stack-Segment zeigt auf den Anfang des Stacks, das Daten-Segment zum Beginn eines Daten-Blocks und das Extra-Segment zeigt auf einen zusätzlichen Daten-Block. So kann man also leicht die verschiedenen Bereiche voneinander trennen. Auch kann man mit Hilfe der Segment-Register leicht Programme relokalisierbar machen, indem man einfach die Segment-Register ändert. Doch dazu später.
Eine Gefahr trägt das Segmentierungs-Verfahren in sich, auf die hier hingewiesen werden soll: Es ist natürlich ohne weiteres möglich, daß sich die verschiedenen Segmente teilweise überlappen. Dann können aber zwei völlig verschiedene logische 16-Bit-Adressen zur selben physikalischen 20-Bit-Adresse führen und damit eventuell ein Chaos auslösen. Ein Beispiel: Enthält das Code-Segment den Wert 1000h, der Programmzähler den Wert 8000h, des Stack-Segment den Wert 1400h und der Stackpointer den Wert 4000h, so sind effektive Programm- und Stackadresse identisch: 18000h.

Raffiniert konstruiert

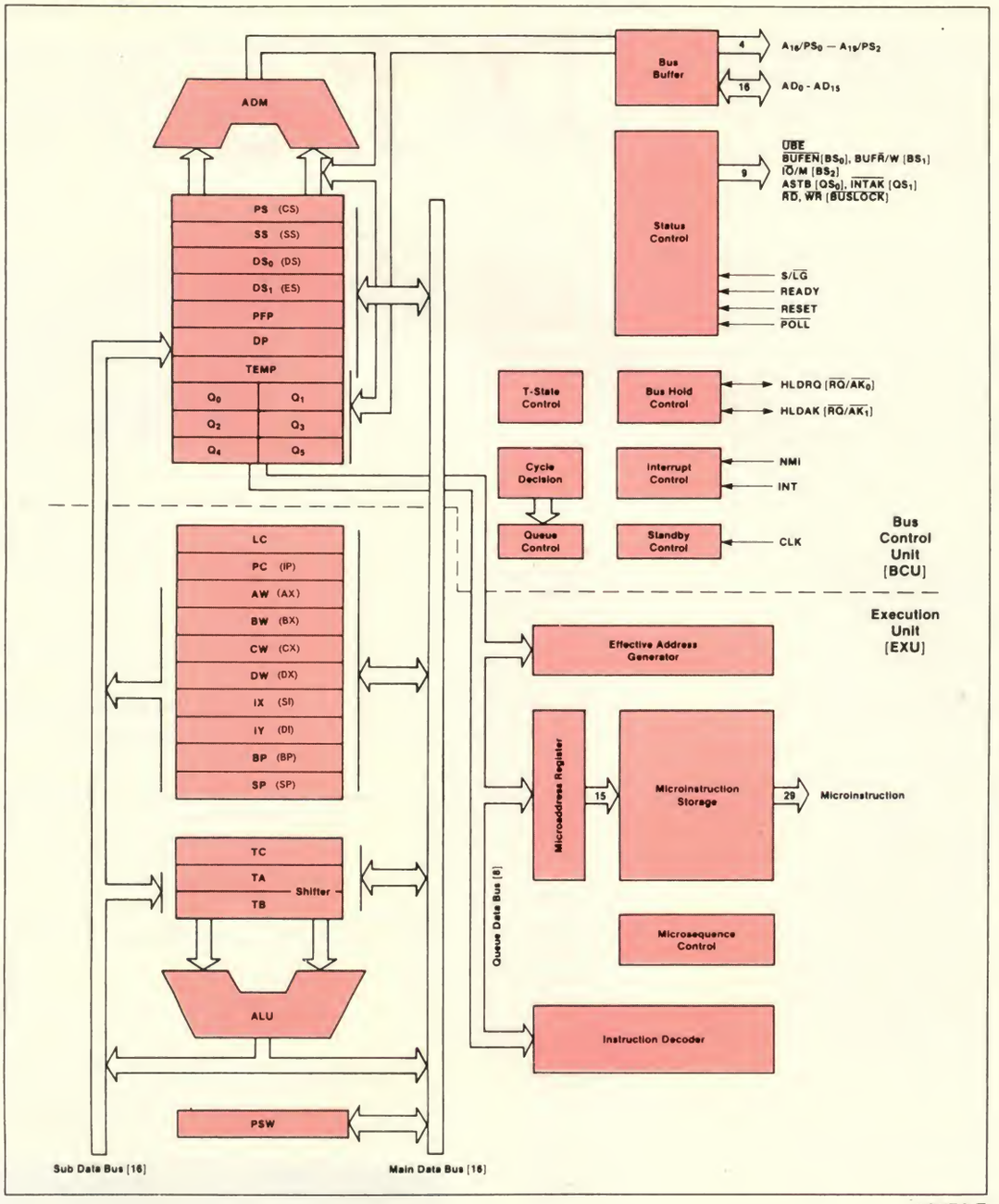
Bild 7 zeigt den internen Aufbau der V30-CPU. Er ist in weiten Teilen mit dem des 8086 identisch, weicht aber doch in einigen Punkten davon ab. Diese (entscheidenden) Unterschiede wollen wir uns jetzt mal etwas genauer anschauen:

Doppelter Datenbus

Was beim ersten Blick auf Bild 7 am mei-

7

Block Diagram



86278-7

Bild 7. Hier sieht man die interne Struktur des V30. Im Text wird auf die Hardware-Verbesserungen gegenüber dem 8086 detailliert eingegangen. Sie sind es, die dem V30 einen teilweise erheblichen Geschwindigkeitsvorteil dem 8086 gegenüber verschaffen. In diesem Bild wurden ausnahmsweise alle Bezeichnungen von NEC übernommen. Die Register-Namen von Intel wurden aber zum besseren Verständnis in Klammern mitangegeben.

sten auffällt, ist zweifellos die Tatsache, daß die meisten der V30-Register nicht nur mit einem, sondern gleich mit zwei Datenbussen verbunden ist. Wozu das gut ist, läßt sich leicht klären, wenn man sich überlegt, wie der Prozessor intern eine Instruktion wie z.B.

ADD AX,BX

abarbeitet. Dazu muß es nämlich zuerst einmal die Inhalte der beiden Register AX und BX in die ALU (Arithmetical/Logical Unit) laden. Nach Ausführung der Addition wird das Ergebnis dann zurück in das Register AX kopiert.

Während nun ein Einzel-Bus-System wie der 8086 zuerst die eine und dann die zweite Hälfte der ALU füllen kann, ist der V30 schneller, denn er füllt beide Hälften gleichzeitig. In Zahlen ausgedrückt sieht das so aus: gegenüber dem 8086 benötigt der V30 statt drei nur noch zwei Takte zur Ausführung des Befehls. Ersparnis: 33 Prozent!

EAG

Nächste Neuerung: der "Effective Address Generator", kurz EAG genannt. Hier werden die physikalischen 20-Bit-Adressen berechnet, wie in Bild 5 gezeigt. Der 8086 besitzt eine solche Einheit nicht. Bei ihm werden die Effektiven Adressen per Microprogramm erzeugt. Das dauert normalerweise zwischen 5 und 12 Taktzyklen. Beim V30 ist die Adresse immer nach nur zwei Taktzyklen berechnet!

LC

Das ist die Abkürzung für "Loop Counter", also "Schleifenzähler". Dieser zusätzliche Zähler wird bei einfachen Blocktransferbefehlen, bei wiederholten I/O-Zugriffen oder bei mehrfachen Schiebe/Rotierbefehlen eingesetzt und erhöht so die Arbeitsgeschwindigkeit der CPU. Dazu wieder ein Beispiel:

ROR AX,CL

Bei der Ausführung dieses Befehls wird

der Inhalt des AX-Registers um so viele Bits rotiert, wie der Inhalt von CL angibt. Ist (CL)=5, so benötigt der 8086 8+(5*4)=28 Takte, der V30 aber nur 7+5=12 Takte. Der Geschwindigkeitsvorteil der CPU mit zusätzlichem Schleifenzähler beträgt mehr als 50 Prozent!

TA, TB und TC

Bei den Registern TA bis TC handelt es sich um Zwischenregister ("Temporary Registers"). Sie sind, wie auch alle übrigen Register, die der V30 dem 8086 voraus hat, dem Programmierer nicht direkt zugänglich, sondern werden nur für interne Operationen genutzt. TA und TB dienen einzeln als zwei 16-Bit- oder zusammen als ein 32-Bit-Schieberegister bei Schiebe/Rotations- sowie Multiplikations/Divisionsbefehlen. Diese Operationen werden deshalb sehr schnell. Auch TC wird in erster Linie bei der Ausführung von Multiplikationen sowie Divisionen verwendet. In ihm werden Zwischenergebnisse abgelegt. Noch einmal wollen wir sehen, was dieser zusätzliche Aufwand gebracht hat: Während der 8086 für eine nicht vorzeichenbehaftete 16 mal 16 Bit Register-Multiplikation je nach deren Inhalt 113 bis 118 Taktzyklen "verbrät" (Microprogramm-Methode), hat der V30 seine Arbeit schon nach 29 bis 30 Takten erledigt! Noch ein Vergleich: Selbst Intels 80186 benötigt 35 bis 37 Zyklen!
Noch größer ist der Unterschied bei Divisionen: NEC schlägt Intel mit 25 zu 144 bis 162 Takten! Diese Werte beziehen sich wiederum auf nicht vorzeichenbehaftete 16 mal 16 Bit Register-Operationen.

PFP

Mit Hilfe des "Prefetch Pointers" wird beim V30 das Pipelining für das Laden der Befehle aus dem Speicher organisiert.
Während "alte" Prozessoren noch die Ausführung des einen und das Laden des nächsten Befehls aus dem Speicher schön eins nach dem anderen erledigen, laufen diese beiden Prozesse nämlich bei neueren Prozessoren, wie dem V30, aber auch schon dem 8086, fast vollkommen getrennt voneinander ab. Das geschieht folgendermaßen: Denken wir uns, daß die CPU per Sprungsbefehl an die Adresse 01000h gelenkt wird. Da dort der nächste Op-Code steht, wird der Programmzähler (PC) mit diesem Wert geladen (das CS-Register enthalte den Wert 0). Dann wird PC in PFP kopiert, diese Adresse auf den Bus gelegt und der gewünschte Op-Code geladen. Soweit ist alles ganz normal. Während aber nun die CPU den geladenen Befehl dekodiert und ausführt, enthält PFP schon die nächste Adresse, legt diese auf den Bus und lädt das folgende Byte in die sogenannte "Pipeline", eine Art CPU-interner Hochgeschwindigkeitsspeicher. Fordert die CPU nun das nächste Byte aus dem Befehlsstrom an, so muß dieses nur noch aus der Pipeline geladen werden. Langsame externe Speicherzugriffe entfallen. Auf diese Weise ist der PFP dem PC immer um eine Nasenlänge voraus.
Das Ganze geht natürlich nur so lange gut, bis der Prozessor einen Sprungbefehl entdeckt. Dann wird nämlich der Inhalt der "Instruction-Pipeline" unbrauchbar und das Spiel geht von vorn los.
Insgesamt gewährleistet diese Methode einen hohen Datendurchsatz bei gleichzeitig großzügig bemessenen Speicherzugriffszeiten.
Im nächsten Teil dieses Beitrages befassen wir uns intensiver mit der Programmierung des V30. Dort werden Adressierungsarten und Befehle sowie die entsprechenden Mnemonics vorgestellt. Auch werden wir uns die Interrupt-Verarbeitung etwas genauer ansehen.
Im dritten und letzten Teil befassen wir uns dann mit der Hardware und dem Interface zu anderen Mikrocomputer-Komponenten. Wir werden sehen, daß man den V30 auf zwei verschiedene Arten betreiben kann. Die eine unterstützt kleinere, die zweite größere Systeme...

Literatur:
µPD 70108/70116 User's Manual, NEC Electronics GmbH, und
Vseries Data Book, NEC Electronics GmbH, Oberrather Str. 4, 4000 Düsseldorf 30.

Anzeigen

Elektor Computing 5

Aktuell zu diesem Heft!

Print 86277	9,30	Auto-Dial (zum Mini-Modem aus EC4)	Teilesatz DM 47,80
Print 86240	71,25	SRAM-Karte für EC65K (4 MHz)	SSO Teilesatz DM 299,20
Print 86238	72,00	Z80-Karte für EC65 und EC65K (4 MHz)	SO Teilesatz DM 164,50
Print 86053	18,00	64 K-DRAM Zusatzspeicher für Kolorator u.a	Teilesatz DM 17,90
86039	23,20	Relaissteuerung für C64 u.a.	Teilesatz DM 35,70
86017	15,40	Echtzeituhr (für I/O-Bus 85058)	Teilesatz DM 89,60
85080.1	61,–	Kolorator (Grafikprozessor 9367)	SO Teilesatz DM 239,60
85080.2	47,35	Kolorator (Farbzusatzkarte)	SO Teilesatz DM 179,50
85079	16,55	Kanalwähler (für I/O-Bus 85058)	Teilesatz DM 33,20
85073	15,70	Serielle Schnittstelle	Teilesatz DM 59,60
65028	39,60	2 × Kassetten Interface + Tracer	Teilesatz DM 125,40
85466	11,45	Tracer für 6502	Teilesatz DM 13,–
85065	11,20	RAM als EPROM mit. Batt.	Teilesatz DM 55,60
85063	16,30	Analogeingang für Computer	Teilesatz DM 56,20
85058	40,45	Universeller I/O-Bus	Teilesatz DM 55,90
85044	27,05	µC-Netzteil bis 25 V/bis 10 A m. Kuhlk. o. Trafo	SO Teilesatz DM 157,90
85020	50,05	X/Y-Plotter o. Druckwerk	SO Teilesatz DM 253,50
85020 D	159,80	Seiko Druckwerk Typ 411 256 oder 411 320	25 m Papier DM 9,90
84094	26,80	Echtzeituhr für µP o. Akku	Teilesatz DM 79,70
85010	11,55	C 64 Kassettenschnittstelle mit Relais	Teilesatz DM 29,90
85007	13,80	EPROM-Umschalter o. EPROM	Teilesatz DM 22,60
84477	23,80	Mikrocomputer-Netzteil	SO Teilesatz DM 159,20
84115.1+2	72,95	Phasenanschnitt per Computer Dia-Umblender	SO Teilesatz DM 184,50
84106	29,85	Thermodrucker 40 Zeichen je Zeile (o. Druckwerk	SO Teilesatz DM 169,40
84106 D	119,90	SO Seiko Druckwerk MTP 401 (25m. Rolle Thermopapier gleich mitbestellen)	
84101	10,75	TV Monitorverstärker	Teilesatz DM 9,90
84078	26,40	RS 232 Centronics-Schnittstelle	Teilesatz DM 179,60
84075	19,95	ZX 81-Impulsputzer m. Netzteil	Teilesatz DM 56,80
84055	20,60	Schreibmaschinen Schnittstelle	Teilesatz DM 79,60
84054	15,30	ZX 81 Puffer	Teilesatz DM 39,50
9344.1+2	33,30	ZX81-Percussion (Rhythmusgen.)	Teilesatz DM 92,20

Oktober 1984 Heft 166

84031	78,30	Telektor: Direkt gekoppeltes Telefonmodem	SO Bauteilesatz DM 387,50
Print 83108.1.2	59,15	CPU-Karte	SO Teilesatz DM 249,90
Print 83102	42,35	Busplatine mit 8 Slots	SO Teilesatz DM 59,80
Print 83082	39,60	VDU-Karte	SO Teilesatz DM 169,90
Print 83058	86,10	ASCII-Keyboard m. Interface	SO Teilesatz DM 269,50
Print 82159	37,45	Floppy-Controller Karte	SO Teilesatz DM 108,70
Print 82017	39,90	64 K-DRAM-Karte	SO Teilesatz DM 165,50
Print 82010 A	38,45	EPROM-Programmer	SO Teilesatz DM 89,50

Gehäuse und Platinen immer extra, soweit nicht anders angegeben. Von Elektor nicht mehr lieferbare Prints nur mit Bauteilesatz lieferbar. Auch für Bestellungen aus dieser Anzeige können Sie das kostensparende Vorauskasse-System benutzen. Überweisen Sie den Betrag auf unser Postscheck- oder Bank-Konto oder senden Sie mit der Bestellung einen Scheck. Bei Bestellungen unter DM 200,– Warenwert plus DM 5,– für Porto und Verpackung (Ausland DM 7,90). Über DM 200,– Lieferwert entfallen diese Kosten (außer Ausland + SO). (Auslandslieferungen nur per Vorauszahlung). Stadtsparkasse Mönchengladbach Konto Nr. 81059 BLZ 31050000. Postscheckkonto Köln 235088-509

HECK-ELECTRONICS
Bauteileversand
Waldstraße 13 · 5531 Oberbettingen Telefon 0 65 93/10 49

18 ec Sonderheft

Space 86 mit 65SC816

Das Mainboard für alle Appple-II Freunde ist da. Mit der 65SC816-CPU und 4 MHz-Takt erhalten Sie durch Space 86 Computerleistung nach Wunsch. Space 86 erhalten Sie fertig aufgebaut, als Leerplatine oder als Computersystem Expert 86.
Ausführliche Informationen liegen für Sie kostenlos bereit. Das Handbuch enthält alle Schaltbilder, die Firmwarelistings sowie viele Tips zum Thema 65SC816.

Handbuchpreis: **DM 30,00**

Interfaces . . .

für Computer mit Applebus.
Ganz frisch für Sie, die 3. Auflage des Erfolgsbuches der ibs Computertechnik zum Thema Interfacekarten für Apple. Ca. 200 Seiten mit (fast) allen Schaltplänen der ibs-Interfaces.

Schutzgebühr incl. Versandk. **DM 10,–**

Aktuell . . .

Die Juni-Info mit vielen Angeboten liegt für Sie bereit. Sie erfahren mehr über Leerplatinen & Halbleiter & Software & Peripherie & Zubehör & Computersysteme & Neuentwicklungen & Partnerschaftsangebote u.s.w.

Zum Thema Space 86, Apple II und Interfaces

Alle Interfaces für den Apple II passen zum Space 86. Wir liefern auch für Ihren Apple: Ramkarten bis 1 MB, & CPU-Karten mit 6809, Z80H-8 MHz & Multifunktion für CP/M 3.0 & schnelle Grafik 512 × 512 Punkte mit 9367 & schnelle AD-Wandler & mc 68000 mit CP/M 68k & MS-DOS am Apple & Universalcontroller auch zum Datenaustausch per Diskette zwischen Apple II und IBM-PC und vieles, was in dieser Aufzählung fehlt.

combit computer gmbh
Olper Str. 10
4800 Bielefeld 14
Tel. 0521-444032
Telex 932 644 ibsd

HARDWARE

EC-65K: Die 64/256K-SRAM-Karte

Eine universelle statische Speicherkarte für den EC-65K

Peter Treytl

Speziell für den EC-65K mit 4MHz-Takt wurde diese Karte entwikkelt. Sie kann mit Speicherbausteinen vom Typ 6264 oder aber auch 62256 (oder äquivalenten Typen) bestückt werden und verfügt damit über einen Speicherumfang von 64 bzw. 256KByte. Mittels Steckbrücken kann die Platine an den gesamten Adressierungsbereich des EC-65K, insgesamt 16MByte, angepaßt werden. In der Bank Ø besteht zusätzlich die Möglichkeit, spezielle Adreßbereiche zum Betrieb von I/O-Karten und eines Monitor-EPROMs auszublenden.

Das Konzept

Vor etwa drei Jahren wurde in Elektor die statische "Universelle Speicherkarte" (1) veröffentlicht. Die Tatsache, daß sie in vielen Junior-, EC-65- und neuerdings auch EC-65K-Rechnern zuverlässig ihren Dienst versieht, bestätigt die Richtigkeit des damals verfolgten Konzeptes. Als Standardbestückung war zur Zeit der Veröffentlichung das 2-KByte-CMOS-RAM HM 6116 LP vorgesehen, das etwa 16 Mark kostete. 8-KByte-CMOS-RAM waren damals kaum im Handel erhältlich und falls, dann zu völlig indiskutablen Preisen.

Die Zeit ist aber nicht stehengeblieben. 8-KByte-Chips gibt es heute bereits für weniger als zehn Mark und die neuen 32-KByte-Chips befinden sich in der Bemusterung. Es ist zu erwarten, daß im Herbst die Preise mit dem Einsetzen der Serienfertigung bereits soweit gesunken sind, daß ein 32-KByte-Chip nicht teurer ist, als vier ICs mit 8 KByte.

Bei der Festlegung des Konzeptes der neuen Speicherchips wurde von der Industrie, wie bei den entsprechenden EPROM's, auf weitestgehende Kompatibilität bezüglich der Anschlußbelegung der Bausteine geachtet. In Bild 1 sind die Pinbelegungen der 8K- und 32K-Bausteine dargestellt. Beiden gemeinsam ist das 28polige Gehäuse. Der beim 6264 nicht belegte (NC) Pin 1 führt beim 62256 das Adreßsignal A14. Das aktive Chipauswahlsignal CS2 auf Pin 26 des 6264 wird beim 62256 durch das Adreßsignal A13 ersetzt. Alle anderen Anschlüsse stimmen überein.

1

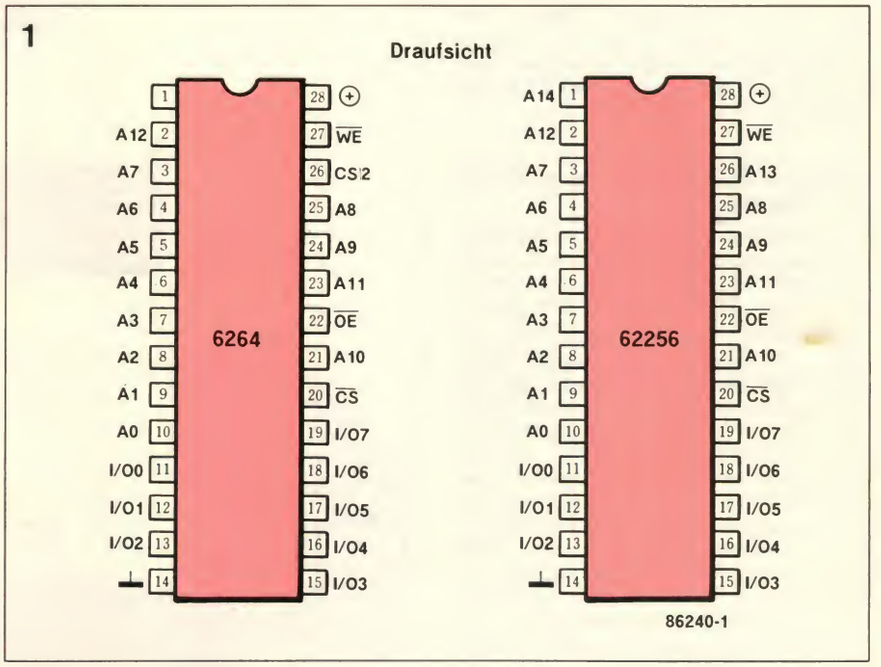


Bild 1. Pin-Belegung der Speicherbausteine 6264 (8K×8) und 62256 (32K×8).

Die innere Organisation des 32-KByte-Speichers zeigt Bild 2. Wesentliches Merkmal ist die Matrix aus 512 × 512 Speicherzellen. Dann gibt es noch Pufferschaltungen für die Zeilen- und Spaltenadressen, Ein- und Ausgabepuffer für die Datensignale und die Kontrollogik für den zeitgerechten Ablauf der Lese- und Schreiboperationen.

Der 8KByte-Speicher besitzt nur 256 × 256 Speicherzellen, ist sonst aber identisch aufgebaut.

Über das Konzept der Speicherkarte ist nicht viel zu sagen. Adreß- und Datenleitungen sind vollständig gepuffert. Die während der Taktphase Φ1 am Datenbus anliegenden Bankadreßsignale werden im Adreß-Latch zwischengespeichert und in der Steuerlogik ausgewertet. Der Adreßdekoder segmentiert den Speicherbereich in 8K-Blöcke und selektiert entsprechend einen von acht Speicherchips. Werden 32-KByte-Speicher verwendet, so werden auch deren Adreßleitungen A13 und A14 angesprochen. Welche Adreßbereiche in welchem der acht Chips gespeichert werden, ist Bild 4 zu entnehmen. Im Falle von Störungen kann so ein fehlerhafter Chip leicht ausgemacht werden.

Die Schaltung

Die Schaltung (Bild 5) entspricht weitestgehend dem Blockschaltbild. die Adreßleitungen AØ bis A15 sind durch die ICs 4 und 5 gepuffert. Die Gatter N1...N3 und N7...N9 ermöglichen es, in Abhängigkeit vom Zustand der Steckbrücken p...t und v, w Adreßbereiche in Bank Ø entsprechend Tabelle 1 auszublenden. Damit können zusammen mit der SRAM-

Karte auch I/O-Karten und das Monitor-EPROM betrieben werden.
Das Latch IC2 speichert die während der Taktphase Φ1 auf dem Adreßbus anliegenden Bankadreßsignale A16...A23. Mittels der Steckbrücken a...j wird entsprechend Tabelle 2 der Adreßbereich eingestellt, in dem die Karte angesprochen werden soll. Über den Eingang $\overline{G}$ von IC3 kann das Selektsignal für den mit den Steckbrücken p...t und v, w definierten Teilbereich gesperrt werden. Die Diode D1 ermöglicht darüber hinaus die Verlangsamung des Prozessortaktes mittels der Busleitung $\overline{SLD}$ (Slow Down) für I/O und das Monitor-EPROM.
Die Einstellung der Speicherkarte auf 8K- bzw. 32K- Speicherchips ist mittels der Brücken h, j und x möglich. Die ersten beiden Brücken machen das Ausgangssignal des Komparators IC3 unabhängig vom Zustand der Adreßleitungen A16 und A17, die nun direkt an die 32K-RAMs geführt werden.
Die Datenleitungen werden mittels IC1 vom Bus entkoppelt. Über den $\overline{OE}$-Eingang dieses Chips wird sichergestellt, daß Buszugriffe nur bei selektierter Karte und während der Taktphase Φ2 erfolgen.

Aufbau, Inbetriebnahme und Test der 256K-SRAM-Karte

Den Bestückungsplan für die doppelseitig kupferkaschierte und durchkontaktierte Platine zeigt die Bild 6. Es lohnt in jedem Fall, die Platine mit einem Durchgangsprüfer auf fehlerhafte Stellen zu testen, vor allem die Durchkontaktierungen. Ist die Platine fehlerfrei, können die IC-Fassungen, Widerstände, Kondensatoren und die Diode entsprechend Bestückungsplan eingelötet werden. Anschließend werden die Pfostensteckverbinder und die 64polige Messerleiste montiert und alle Lötstellen einer sorgfältigen Sichtkontrolle unterzogen. Nach Prüfung der Versorgungsleitungen auf Kurzschluß können die ICs eingesetzt und die Brücken entsprechend den Tabellen 1 und 2 gesteckt werden. Sicherheitshalber kontrolliert man noch die Ruhestromaufnahme, die um 220 mA liegen sollte.
Nach der ersten Inbetriebnahme ist die Karte sorgfältig auf einwandfreie Funktion zu testen. Leser, die bereits den EC-65K aufgebaut haben, prüfen die Karte am besten in einem noch nicht belegten Bereich, z.B. ab $040000, bevor sie die bisherige RAM-Karte in Bank Ø ersetzen.
Sollte es wider Erwarten Probleme bei der Inbetriebnahme der SRAM-Karte geben, sei auf Bild 7 verwiesen. Es zeigt das Timing für den 4-MHz-Betrieb. In diesem Fall dürfen die Zugriffszeiten der Speicherchips nicht über 100ns liegen. An allen kritischen Stellen müssen außerdem, wie in der Stückliste angegeben, F-(Fast) oder AS-(Advanced Schottky) Schaltkreise eingesetzt werden. Gibt man sich mit einem Prozessortakt von 2 MHz zufrieden, reichen LS-Schaltkreise und die Diode D1 kann eine einfache Siliziumdiode vom Typ 1N4148 sein. Bei den Speicherbausteinen genügt dann auch die billigere Ausführung mit Zugriffszeiten um 150ns.

2

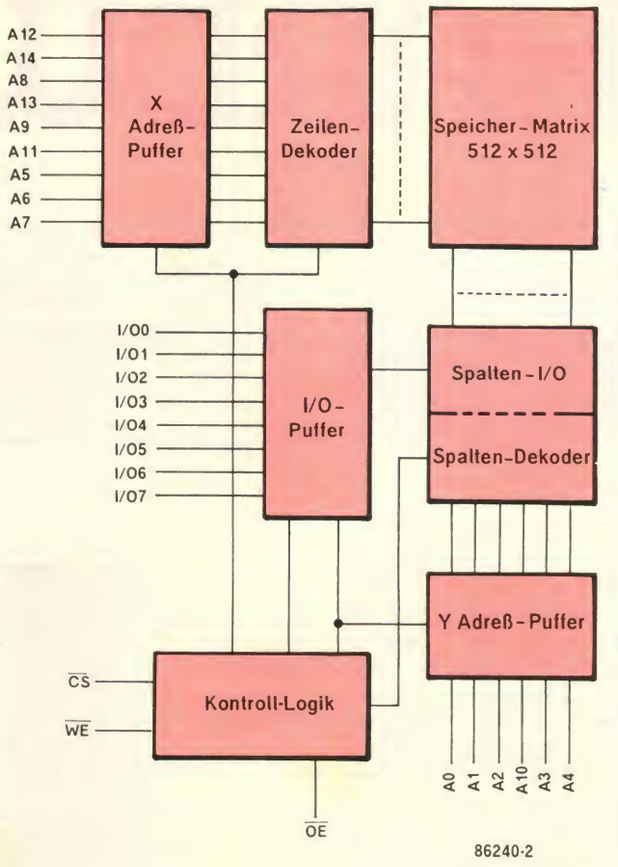


Tabelle 1

RAM ausblenden		Steckbrücke							
von	bis	p	q	r	s	t	u	v	w
$0E000	$0EFFF	●	○	○	●	●	⊕	○	●
$0E000	$0E7FF	●	○	●	○	●	⊕	○	●
$0E000	$0FFFF[1]	○	●	○	●	●	⊕	○	●
nichts ausblenden		○	○	○	○	○	○	●	○

○ Brücke offen
● Brücke geschlossen
⊕ Brücke schließen, wenn für ausgeblendeten Bereich "Slow-down" gewünscht.
[1] EC65K Standard-Konfiguration

Literatur:
(1) Universelle Speicherkarte Elektor, Nr. 147, März 1983, Seiten 60-68
(2) Elektor Computing, 4. Sonderheft, April 1986

Bild 2. Der Speicherbaustein 62256 besteht aus einer quadratischen Anordnung von 256K statischen Speicherzellen mit den dazugehörigen Adreßdekodern, den Ein- und Ausgabepuffern sowie der Kontrollogik, die den zeitlichen Ablauf der Speicheroperationen steuert.

Tabelle 1. So werden die Steckbrücken gesetzt, um in der Speicherbank Ø die angegebenen Bereiche für I/O und EPROM auszublenden. Für diese Bereiche kann mit der Steckbrücke u auch noch der Prozessortakt verlangsamt werden.

Tabelle 2

RAM-Konfiguration			Steckbrücke/Adresse												RAM Typ
			a	b	c	d	e	f	g	h	i	j	x	y	
KByte	von	bis	A23	A22	A21	A20	A19	A18	A17	—	A16	—	—	—	
64	$000000	$00FFFF	●	●	●	●	●	●	●	○	●	○	○	●	6264
64	$010000	$01FFFF	●	●	●	●	●	●	●	○	○	○	○	●	6264
64	$020000	$02FFFF	●	●	●	●	●	●	○	○	●	○	○	●	6264
64	$030000	$03FFFF	●	●	●	●	●	●	○	○	○	○	○	●	6264
256	$000000	$03FFFF	●	●	●	●	●	●	○	●	○	●	●	○	62256
256	$040000	$07FFFF	●	●	●	●	●	○	○	●	○	●	●	○	62256
256	$080000	$0BFFFF	●	●	●	●	○	●	○	●	○	●	●	○	62256
256	$0C0000	$0FFFFF	●	●	●	●	○	○	○	●	○	●	●	○	62256

○ Brücke offen
● Brücke geschlossen

3

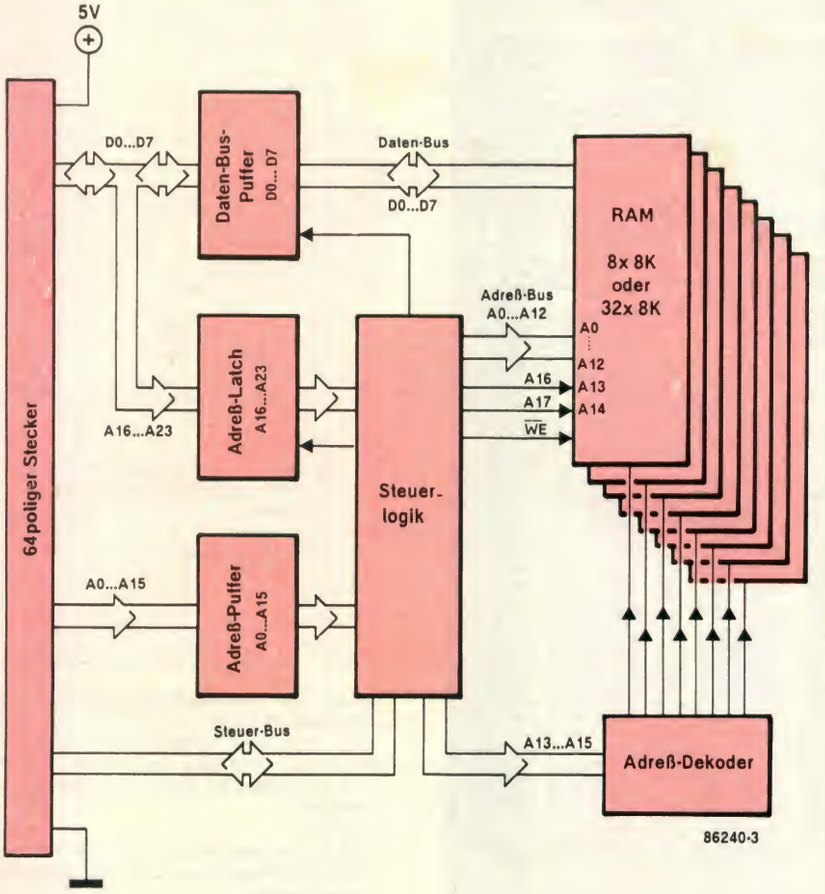
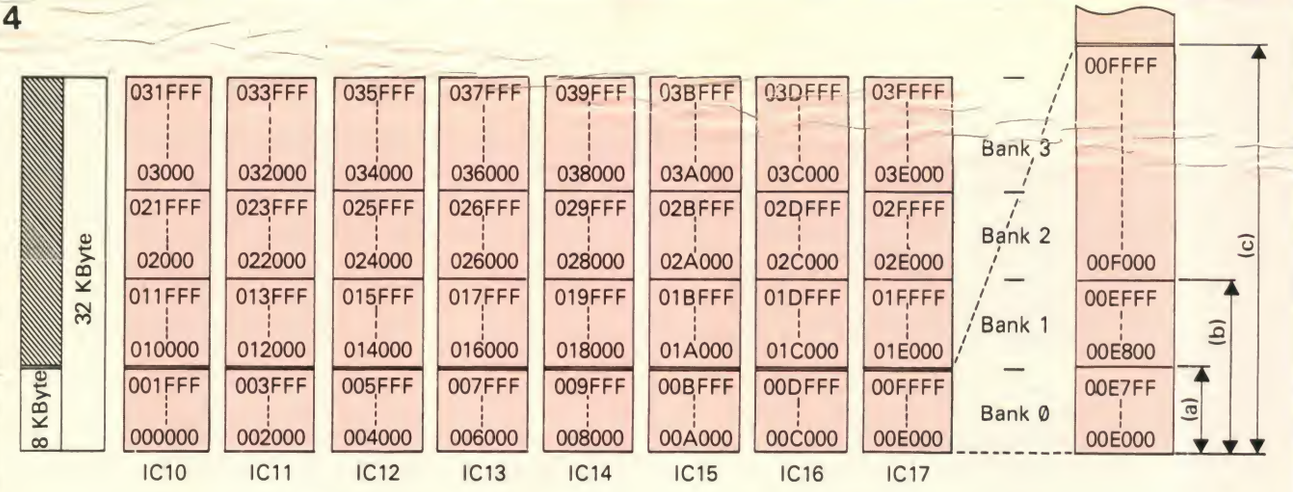


Tabelle 2. Zum besseren Verständnis der Codierung der Adreßauswahlsignale für die SRAM-Karte sind hier je vier Beispiele für Speicherchips vom Typ 6264 und 62256 dargestellt. Bei geschlossener Brücke wird die Speicherkarte selektiert, wenn das entsprechende Adreßbit am Bus Low ist. Bei offener Brücke wird der Zustand High erwartet. Die Steckbrücken h, j, x und y dienen der Umschaltung zwischen Speicherbausteinen mit 64 und 256 KByte Speicherkapazität.

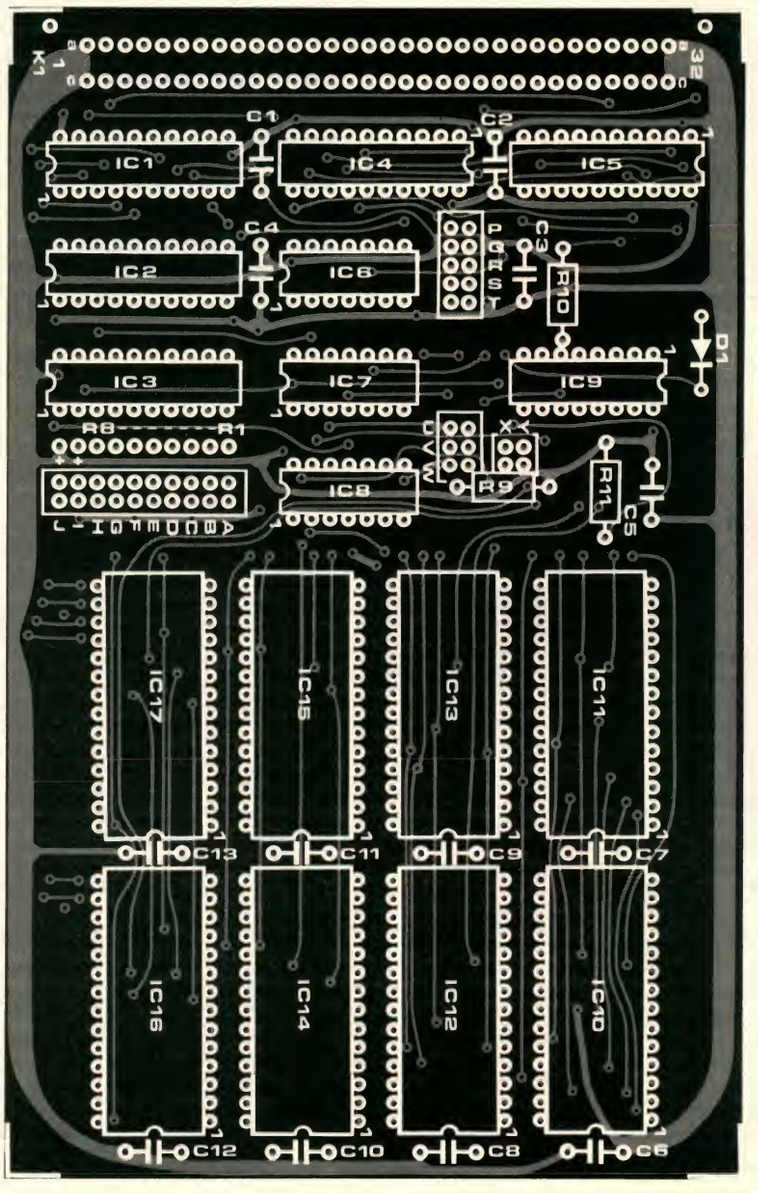
Bild 3. Das Blockschaltbild der SRAM-Karte läßt die wesentlichen Schaltungsteile wie Buspuffer, Steuerlogik und den eigentlichen Speicher gut erkennen.

4



Abhängig von der Einstellung der Steckbrücken p, q, r, s, t können in Bank Ø folgende Bereiche für I/O und EPROM ausgeblendet werden:
(a) 00E000 bis 00E7FF
(b) 00E000 bis 00EFFF
(c) 00E000 bis 00FFFF (EC-65K Standard-Konfiguration)

6



Stückliste:

(für 4MHz-Betrieb)

Widerstände:
R1...R8 = 2k2 (oder Netzwerk)
R9...R11 = 2k2

Kondensatoren:
C1...C9 = 100 n
C10...C13 = 10 µ/12 V Tantal

Halbleiter:
D1 = HSCH 1001 (hp)*
IC1 = 74F245
IC2 = 74F373
IC3 = 74LS688
IC4,IC5 = 74LS541
IC6 = 74F04
IC7 = 74S260
IC8 = 74F00
IC9 = 74F138
IC10...IC17 = 6264-100 oder 62256-100

außerdem:
64polige Messerleiste nach DIN 41612
Pfostenfeldstifte, zweireihig:
1× 20polig, 1× 10polig, 1× 6polig,
1× 4polig
Kurzschlußstecker (Jumper)

Platine EPS

* baugleich: 1N6263!

Bild 4. Der Adreßdekoder segmentiert den Speicher in Blöcke von 8KByte. Das Bild zeigt, wie die einzelnen Adreßbereiche in 8K- und 32K-RAMs abgespeichert werden und erleichtert so eine eventuelle Fehlersuche.

Bild 6. Der Bestückungsplan der SRAM-Karte mit Angabe der Lage der einzelnen Bauelemente.

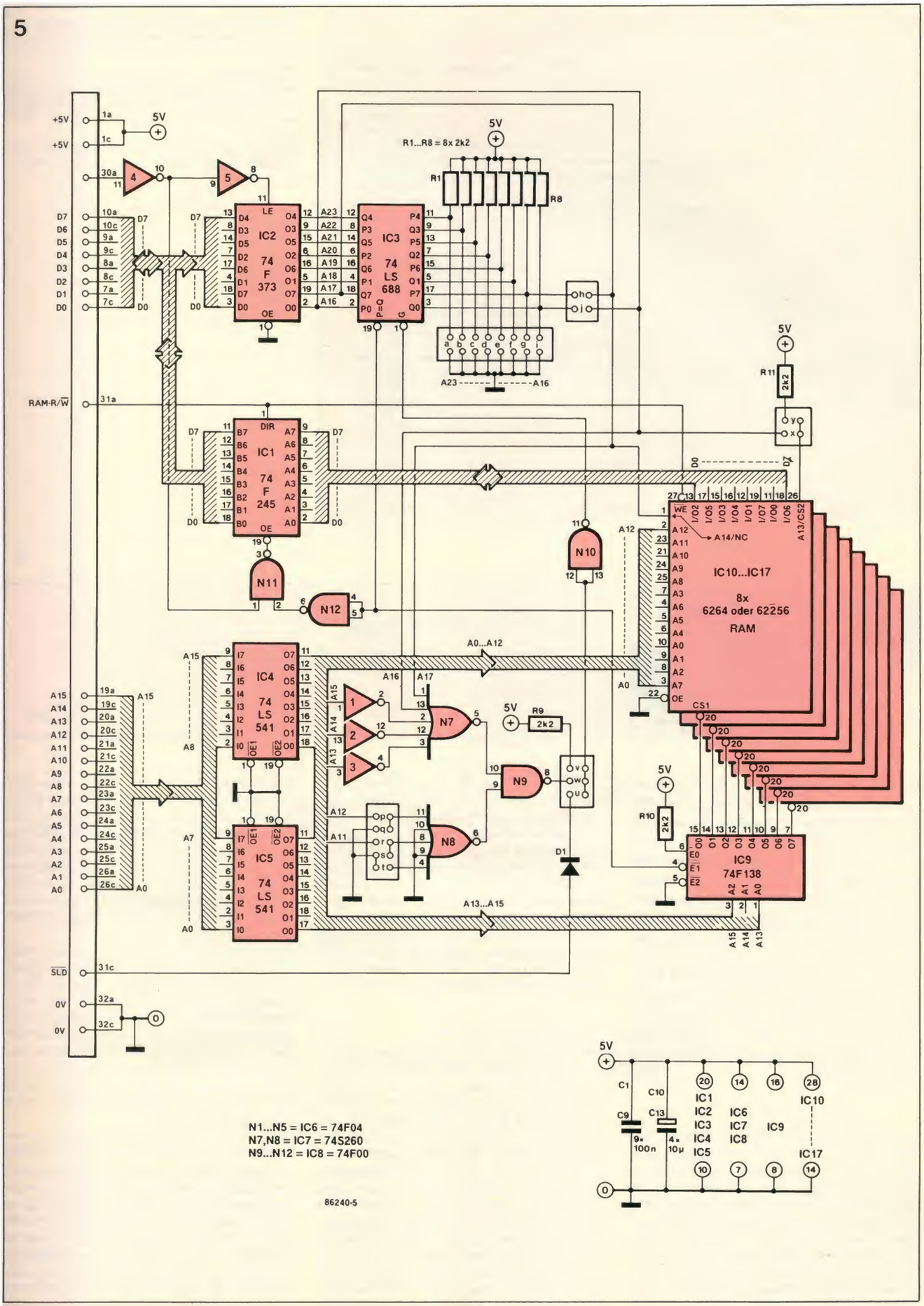


Bild 5. Das Schaltbild der SRAM-Karte. Es entspricht in seinem Aufbau weitgehend dem Blockschaltbild.

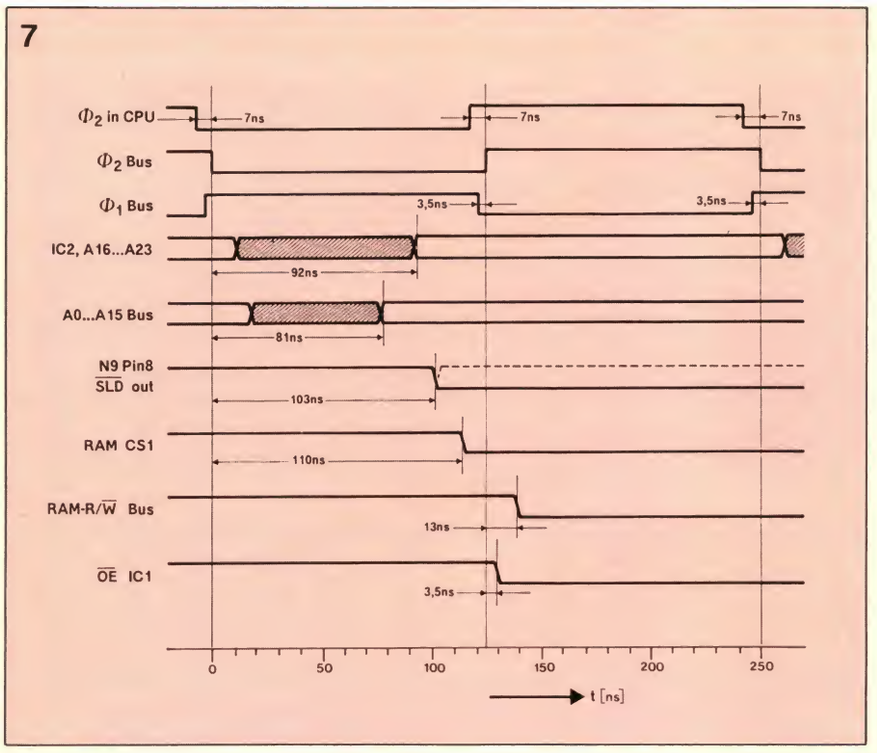
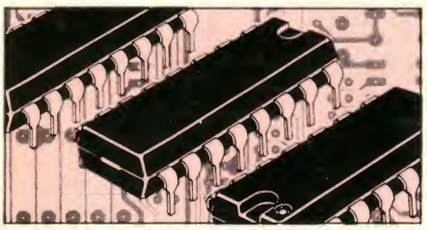


Bild 7. Das Zeitablaufdiagramm für die SRAM-Karte enthält wichtige Hinweise über die Laufzeiten wesentlicher Signale.

Hinweis:

Bei Einsatz der Karte über 64 KByte hinaus (also im 16-Bit-Modus) sind kleine Modifikationen an anderen Karten nötig (FCU-Karte, VDU-Karte, . . .).
Mehr darüber im nächsten ec-Heft, die Software-Verteiler werden vorab per Rundbrief informiert.



Auto-Dial-Interface

Telefon-Selbstwahl für Computer

J. Steeman

Der Datenaustausch per Modem ist längst zur Selbstverständlichkeit für viele Computeristen geworden. Für andere nicht: Ihnen sind die so rapide wachsenden Telefongebühren zu hoch. Ja, wenn man die Computer so programmieren könnte, daß sie in den billigen Nachtstunden selber die Daten austauschen... Warum denn nicht — es muß lediglich eine kleine Zusatzhardware fürs Modem her. Hier ist sie: Das Auto-Dial-Interface. Und dann werden die Siliziumsklaven auf Nachtschicht programmiert!

Das Mini-Modem, dessen Schaltung und Bauanleitung wir im vorangegangenen EC-Heft veröffentlichten, schlägt Brücken von einem Computer zum anderen. Mit dem Modem kann man problemlos Daten beliebiger Art über Telefonleitungen senden und empfangen. Dabei hat das Modem die Funktion eines Bindeglieds zwischen dem Computer und der Fernsprechleitung. Unser Mini-Modem ist zwar vielfältig nutzbar, denn es läßt sich auf mehrere Übertragungsprotokolle umschalten. Es kann jedoch ohne menschliches Zutun weder Anrufe entgegennehmen (Auto Answer) noch allein Rufnummern wählen (Auto Dial). Deshalb beschreiben wir hier eine kleine Schaltung, die, zusammen mit ein wenig computerspezifischer Software, dem Modem (oder dem Computer?!) zu mehr Selbständigkeit im zweiten Punkt verhilft. Unsere Auto-Dial-Schaltung kann übrigens jedes beliebige Modem ergänzen; es wird einfach in die Leitung zwischen Modem und Leitungsanschlußdose eingeschleift.

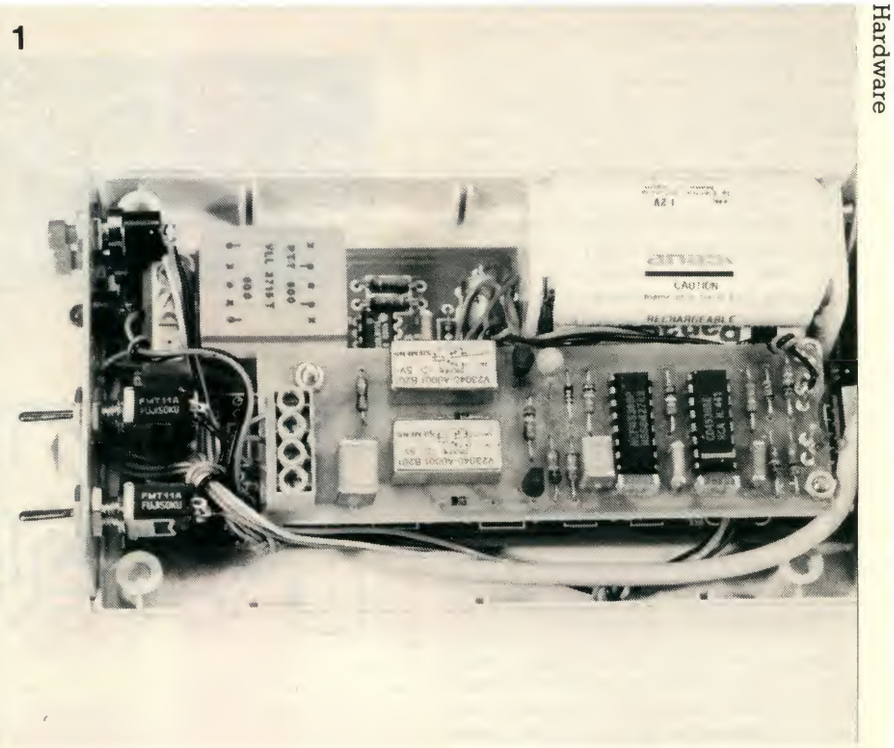
. . . ruf doch mal an!

Die Zeiten, in denen man ein paarmal auf die Gabel tippen mußte, um das "Fräulein vom Amt" zu rufen, sind längst vorbei. Damals fiel am sogenannten Klappenschrank eine kleine Klappe herunter, hinter der die Nummer des rufenden Teilnehmers stand. Die Telefonistin mußte den Teilnehmer nach seinen Wünschen befragen und dann die Leitung von Hand stöpseln. Ja, lang' ist's her. . .! Heute werden sogar interkontinentale Verbindungen vollautomatisch, ohne irgendein "Fräulein vom Amt" aufgebaut. Ausnahmen machen hiervon nur noch Gespräche in einige wenige exotische Länder.
Was geschieht nun heute in unseren modernen Telefonapparaten und auf den Leitungen, wenn wir dem bekannten Werbeslogan der Post folgen, zum Hörer greifen und eine Nummer wählen? Zur Zeit werden hauptsächlich zwei Wählverfahren angewandt, von denen das zweite bei uns bisher noch auf Nebenstellenanlagen beschränkt ist: das klassische Impulswählverfahren (IWV) und das wesentlich jüngere Mehrfrequenzverfahren (MFV).
Beim Impulswählverfahren wird der vom Amt über den Apparat fließende Strom von einem Schaltkontakt in der Wählscheibe (oder im Tastenwahlblock) während des Wahlvorgangs rhythmisch unterbrochen. Wählen wir zum Beispiel die Ziffer "3", so besteht die Impulsreihe aus drei Unterbrechungen, wählen wir eine "5", dann sind es fünf, — und so weiter (bei einer "0" sind es allerdings nicht null, sondern zehn!).
Ganz anders arbeitet das Mehrfrequenzverfahren: Die Wählinformation besteht für jede Ziffer aus zwei bestimmten, gleichzeitig ausgesandten Tönen. Insgesamt stehen acht Töne zur Verfügung, so daß maximal 16 unterschiedliche Zweierkombinationen möglich sind. Wie schon erwähnt, ist das Mehrfrequenzverfahren bei uns auf Nebenstellenanlagen beschränkt (in den USA und verschiedenen anderen Ländern ist das anders!). Deshalb arbeitet das hier beschriebene Auto-Dial-

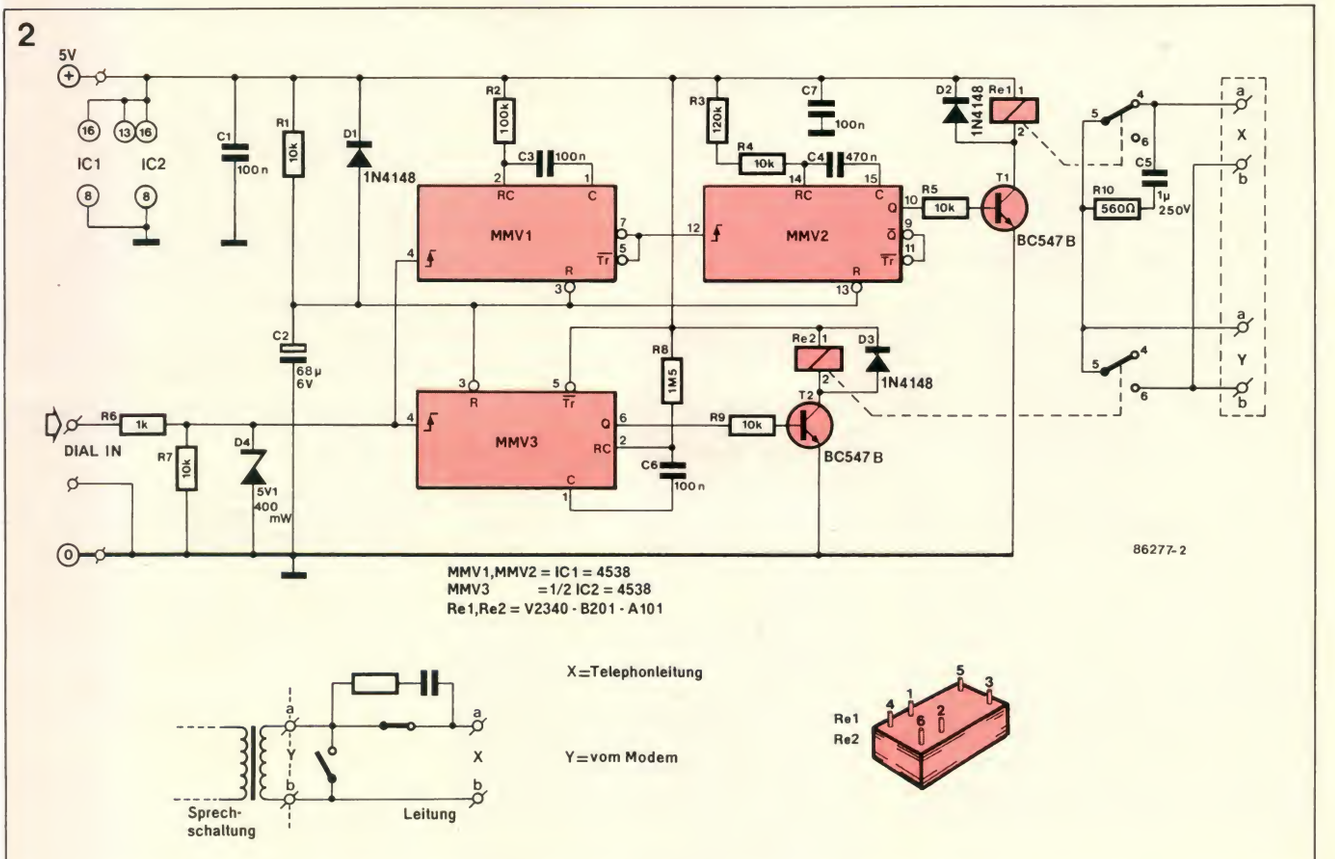
Bild 1. Wie auf diesem Foto zu erkennen ist, kann man die kleine Platine mit dem Auto-Dial-Interface leicht im vorhandenen Modemgehäuse unterbringen.

Bild 2. Die Schaltung verrät auf den ersten Blick: Der Hardware-Aufwand hält sich in Grenzen.

1



2



Interface nach dem Impulswählverfahren. Auf den Einsatz spezieller ICs, wie sie auch in den Tastenwahlapparaten der Post zu finden sind, haben wir bewußt verzichtet. Mit diesen ICs wäre zwar der Aufwand an Software geringer, die IC-Steuerung würde jedoch einen kompletten Userport belegen. Für unsere Schaltung genügt dagegen ein einziges Bit des Userports.

Impulsives

Beim Impulswählverfahren beträgt die Pulsfolgefrequenz 10 Hz, die Impulspausen (Leitungsunterbrechungen) dauern 60 ms, und die dazwischenliegenden Schließzeiten des Wählkontaks sind auf 40 ms festgelegt. Die Impulsserie für die Ziffer "0" (10 Impulse) ist daher genau eine Sekunde lang. Zwischen zwei Ziffern muß eine Wählpause vom mindestens 800 ms einschoben werden, damit die Vermittlungsstelle den Beginn der folgenden Ziffer erkennen kann. Ferner darf der Wählvorgang erst beginnen, wenn die Vermittlungsstelle den sogenannten Wählton (Dauerton nach Abheben des Hörers) sendet. Falls während des Wählens keine freie Leitung in die gewünschte Richtung verfügbar ist, schaltet die Vermittlungsstelle den Besetztton auf die

Teilnehmerleitung; normalerweise wird man dann kurz auflegen und anschließend erneut wählen.
Unser Auto-Dial-Interface kann den Wähl- und Besetztton (und auch den Rufton) leider *nicht* erkennen und auswerten; hierzu wäre ein erheblich höherer Aufwand nötig. In der Praxis genügt es aber, wenn zwischen dem Anschalten des Modems an die Leitung und dem Wählbeginn eine starre Pause von ca. 1 s liegt; diese Pause muß (bei vollautomatischem Betrieb) durch die Software realisiert werden. Daß die Verbindung aus irgendeinem Grund nicht zustandegekommen ist, kann die eigene Software beispielsweise daran erkennen, daß eine im Übertragungsprotokoll festgelegte Beginnsequenz des gerufenen Computers ausbleibt.

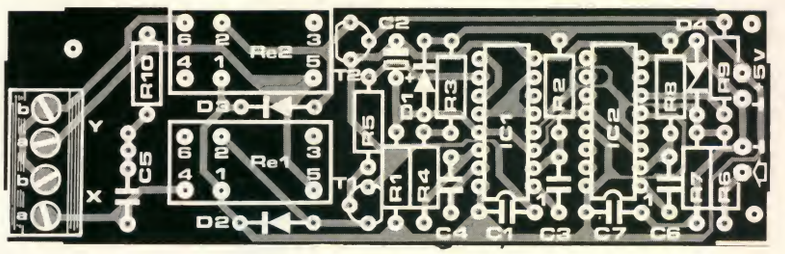
Hard- und Software

Ebenso wie zur Sprechschaltung des Telefonapparats gehört auch zur Eingangsschaltung eines Modems ein Übertrager, also eine Induktivität. Damit die Wählimpulse die vorgeschriebene Impulsform erhalten, wird diese Induktivität während des Wählens kurzgeschlossen. Ferner ist der Unterbrecherkontakt durch ein RC-Glied überbrückt, das ebenfalls die Impulsform beeinflußt und außerdem Spannungsspitzen und hochfrequente Störungen unterdrückt. In Bild 1 ist dies vereinfacht dargestellt. Das Auto-Dial-Interface arbeitet deshalb, wie die Schaltung in Bild 2 zeigt, mit *zwei* Relais: Rel unterbricht den Leitungsstrom, während Re2 das Modem kurzschließt. Die Impuls- und Pausenzeiten und auch die Schließzeit von Re2 werden von drei Monoflops gesteuert. Monoflop 3 ist nachtriggerbar; Re2 zieht deshalb so lange an, wie am Schaltungseingang ein Wählsignal vom Computer eintrifft. Monoflop 1 stellt sicher, daß das Modem kurzgeschlossen ist, bevor der erste Wählimpuls über die Leitung geschickt wird, indem es die Wählsignale des Computers um ca. 10 ms verzögert. Die Zeitkonstante von Monoflop 2 beträgt 60 ms; dies ist die vorgeschriebene Dauer der Leitungsunterbrechung.
Was die Software betrifft, so sind in vielen käuflichen Datenübertragungspaketen bereits Routinen enthalten, die das Auto-Dial-Interface ohne irgendeine Änderung steuern können. Auskunft darüber gibt das zum Programmpaket gehörende Handbuch, oder man frage den Software-Lieferanten. Wer noch keine geeignete Software besitzt, kann sie natürlich auch selbst schreiben: Bei einigen Computern ist dies in Basic möglich, andere lassen sich nur in Maschinensprache dazu bewegen, die notwendigen Signale über einen Port an das Auto-Dial-Interface zu senden.

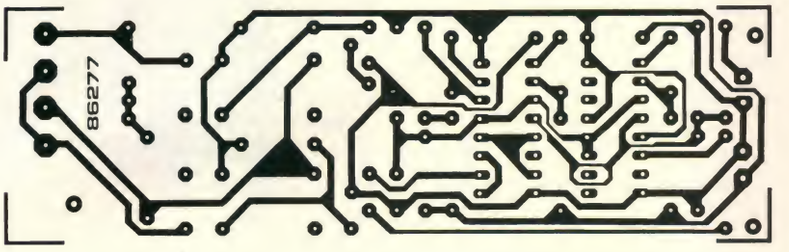
Interface-Bau

Obwohl sicherlich auch andere Relais geeignet wären, haben wir für das Auto-Dial-Interface einen bereits in anderen EC-Schaltungen verwendeten Relaistyp von Siemens gewählt (siehe Stückliste). Dieses Relais hat nicht nur eine niedrige Ansprechleistung, sondern es besitzt auch eine hohe Spannungsfestigkeit zwischen Spule und Schaltkontakt. Die zweite Eigenschaft ist bei Schaltungen, die mit dem Telefonnetz in direkter Verbindung stehen, besonders wichtig.
Wir haben für die Schaltung eine kleine Platine entworfen, die sich sicher noch im Modem-Gehäuse unterbringen läßt. Auf der einen Seite liegen die Anschlußklemmen für Modem und Leitung, auf der anderen die Verbindungspunkte zum Computer.
Der Computer muß positiv gerichtete Wählimpulse mit der schon genannten Folgefrequenz 10 Hz liefern (Pause zwischen zwei Impulsserien mindestens 800 ms); die Impulsbreite muß so bemessen sein, daß MF1 und MF2 sicher triggern. Die Impulshöhe (Spannung) darf auch über TTL-Niveau liegen, denn der Schaltungseingang ist durch eine Zenerdiode gesichert.
Die Betriebsspannung für das Auto-Dial-Interface kann man entweder dem Modem oder dem Computer entnehmen. Im Ruhezustand ist der Strombedarf praktisch gleich null, bei angezogenen Relais beträgt er ca. 50 mA.
Abschließend darf der übliche Hinweis nicht fehlen: Das Auto-Dial-Interface darf in diesem unseren Lande nicht an das öffentliche Fernsprechnetz angeschlossen werden (mit freundlichen Grüßen, Ihr Black-Penny).

3



4



Stückliste

Widerstände:
R1,R4,R5,R7,R9 = 10 k
R2 = 100 k
R3 = 120 k
R6 = 1 k
R8 = 1M5

Kondensatoren:
C1,C3,C6 = 100 n
C2 = 60 µ/16 V
C4 = 470 n
C5 = 1 µ/250 V

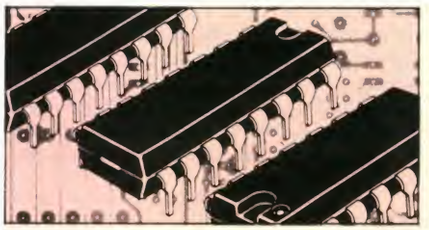
Halbleiter:
D1,D2,D3 = 1N4148
D4 = 4V7/400 mW Z-Diode
T1,T2 = BC547B
IC1,IC2 = 4538

außerdem:
Re1, Re2 Relais Siemens V23040-B-2d-A1
Lüsterklemme für Platinenmontage, 4polig

Platine EPS 86277

Bild 3. Die Platine des Auto-Dial-Interfaces dürfte sich leicht noch im Modem-Gehäuse unterbringen lassen.

Bild 4. Der Bestückungsplan der Platine – damit Sie nicht lange rätseln müssen, wo welches Bauteil hingehört.



CP/M auf dem EC-65 (Teil 1)

Die Z80 Coprozessor-Karte für den EC-65 macht nicht nur das riesige CP/M-Softwareangebot zugänglich...

Frank Schmidt

"Was kann man denn als EC-65-Besitzer mit einer zweiten Prozessor-Karte anfangen?" - Diese Frage haben wir zum Teil schon im letzten ec-Heft geklärt: Man kann damit CP/M-Programme "fahren". Aber mit ein bißchen Fantasie wird man leicht einsehen, daß das noch lange nicht alles war. Ein kleiner Vorgeschmack: Bei rechenintensiven Programmen kann man die beiden CPUs parallel arbeiten lassen. Und der große, auf der Z80-Karte befindliche Speicher braucht auch dann nicht "brach" zu liegen, wenn man nicht unter CP/M arbeitet: Er kann als Printer-Spooler, RAM-Disk oder als Zwischenspeicher für beliebige Daten genutzt werden.

Natürlich benötigt man für alle diese Anwendungen auch die passende Software. Und die will erst mal geschrieben sein. Alle für den Betrieb als CP/M-Rechner notwendigen Programme (und noch eine Menge mehr) stellen wir an anderer Stelle in diesem Heft vor. Auch zeigen wir, wie man den Z80-Computer als Grafik-Daten-Zwischenspeicher für unsere allseits beliebte Kolorator-Karte benutzt (damit kann man sich wiederholende Berechnungen einsparen - das gibt der Grafik eine wesentlich größere Geschwindigkeit).
Die Verwendung als Printer-Spooler beschreiben wir vielleicht im nächsten Heft. Ein bißchen mehr Schwierigkeiten gibt es beim Implementieren einer RAM-Disk. Dafür ist nämlich das OHIO-DOS überhaupt nicht eingerichtet. Mit größter Wahrscheinlichkeit werden wir eine solche Implementation aber in dem neuen Betriebssystem des EC-65K vornehmen, das sich bei uns gerade in der Erprobungsphase befindet.

Konzept

Doch wenden wir uns nach diesem kleinen Überblick endlich mal der Hardware zu! Die Z80-CPU-Karte ist in erster Linie unter folgenden Aspekten entwickelt worden:

— Die gesamte Schaltung muß auf eine Europakarte passen.
— Um einen hohen Datendurchsatz zu gewährleisten, sollte die CPU möglichst schnell sein.
— Aus demselben Grund sollte eine möglichst große Speicherkapazität zur Verfügung stehen, die man notfalls noch auf einer zweiten Platine erweitern kann.
— Aus Platz- und Preisgründen wird außerdem auf einen weiteren Ausbau des Z80-Systems verzichtet.

Was wir von diesen Vorgaben verwirklicht haben, das können Sie dem Kasten auf Seite 32 entnehmen. Dort sind die wichtigsten technischen Daten festgehalten. Sie sehen daraus, daß wir nicht nur die ursprünglich auf drei Eurokarten verteilte Schaltung kräftig abgespeckt haben, sondern ihr darüberhinaus noch ganz schön Beine gemacht haben. Damit können Sie Ihren EC-65 zu einem der schnellsten CP/M-Rechner ausbauen, die es zur Zeit gibt! Das gilt vor allem dann, wenn Sie den Speicher später auf 512 oder gar 1024 KByte ausbauen. Wir haben vor, diesen zusätzlichen Speicher dann als Disketten-Puffer ("Cache-Speicher") zu benutzen. Damit werden Diskettenzugriffe auf ein absolutes Minimum reduziert. Allerdings entfallen dann auch die Kaffeepausen, die bis jetzt noch beim Laden der RAM-Disk entstehen...

1

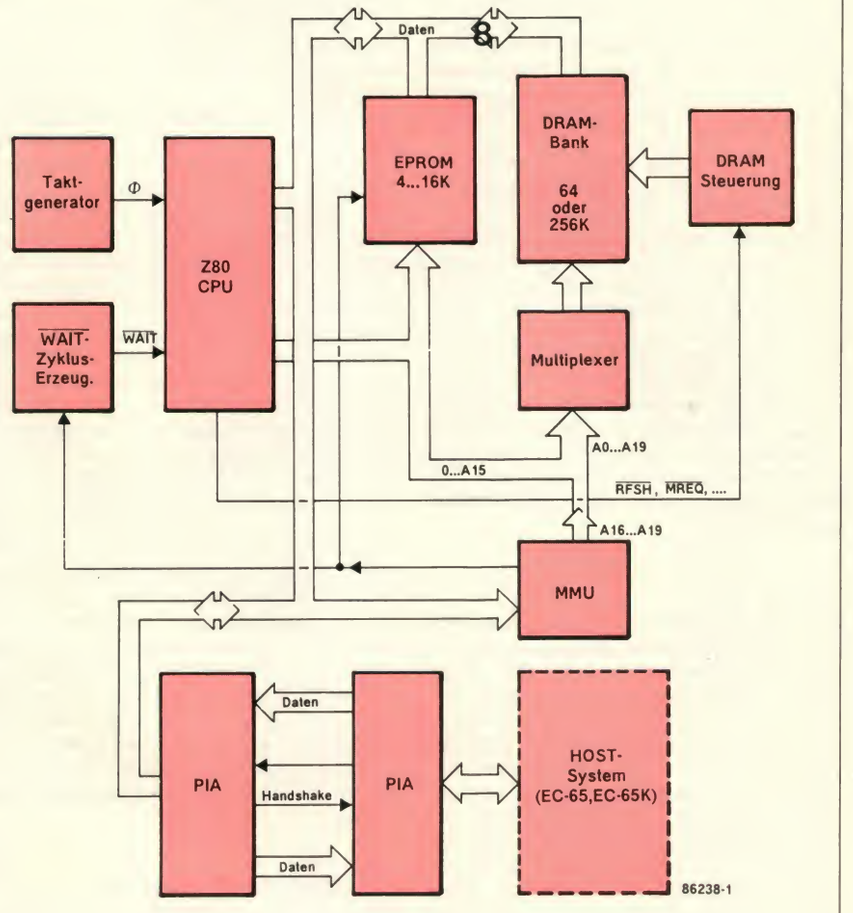


Bild 1. Das Blockschaltbild der Z80-Karte. Vom EC-65 wird sie über ein FIFO, aufgebaut mit zwei PIAs, angesprochen.

Die Schaltung

Werfen Sie nun bitte einen Blick auf das Blockschaltbild (Bild 1). Dort sind - der besseren Übersichtlichkeit wegen - nur die wichtigsten Verbindungen eingezeichnet.
Im oberen Teil befindet sich der komplette Z80-Computer mit CPU, RAM, ROM und I/O. Das mit zwei parallelen Ports aufgebaute FIFO bildet die Schnittstelle zum EC-65. Erwähnenswert ist noch der Kasten "MMU" im rechten Teil des Block-

schaltbildes. Dort werden die Adressen A16 bis A19 erzeugt, die nötig sind, um das System bis zu einem Megabyte (!) Speicher ausbauen zu können. Außerdem blendet die MMU das EPROM nach dem Kopieren in den RAM-Bereich aus, so daß ganze 64 KByte RAM vom Prozessor auf einmal adressiert werden können.

Kommen wir jetzt zu dem eigentlichen Schaltbild (Bild 2). Links oben befindet sich der Taktgeber des Systems. Er ist mit den Gattern N1 und N2 sowie zwei Widerständen, einem Kondensator und dem Quarz aufgebaut. Ihm nachgeschaltet ist ein Flip-Flop, das der Verbesserung der Flankensteilheit sowie einer Division der Frequenz durch zwei dient. Auf diese Weise entsteht ein absolut symmetrisches Taktsignal, das an den entsprechenden CPU-Eingang geführt wird. Der Quarz X1 schwingt dabei mit doppelter CPU-Frequenz.
Die beiden Eingänge $\overline{NMI}$ und $\overline{BUSRQ}$ der CPU werden hier nicht benutzt und liegen deshalb einfach an +5V. Der RESET wird direkt vom EC-65 abgeleitet. Die Flip-Flops FF5 bis FF8 sorgen dafür, daß bei jedem Zugriff auf das EPROM drei Wait-Zyklen eingefügt werden. Damit kann man selbst ein 8-MHz System mit billigen 350ns-oder 400ns-EPROMs bestükken. Außerdem hat diese Verzögerung der EPROM-Speicherzugriffe keine nachteiligen Auswirkungen auf die CPU-Geschwindigkeit. Schließlich wird das EPROM ja nur nach einem RESET in's RAM kopiert und dann ganz abgeschaltet.

RAM - ROM

Das Flip-Flop FF2 stellt den ersten Teil der MMU (Memory Managing Unit = Speicherverwaltungseinheit) dar. Nach einem RESET (also z. B. nach dem Einschalten) ist es zurückgesetzt. Damit ist gleichzeitig der $\overline{CS}$-Eingang des EPROMs aktiviert. Wird jetzt eine Adresse des unteren Speicherbereichs ausgelesen (A15 ist dann LOW), so wird über N7 das Signal $\overline{ROMSEL}$ aktiv. Über N27 wird der Dekoder '155 und damit auch der Zugriff auf das RAM gesperrt. Wird nun weiter $\overline{RD}$ aktiv, so schaltet auch N9 durch, das EPROM wird angesprochen und kann damit ausgelesen werden.
Führt die CPU dagegen einen I/O-Zugriff aus, so geht das Signal $\overline{IORQ}$ kurzzeitig auf Nullpegel und triggert FF2. Der an seinem Eingang liegende HIGH-Pegel wird an den Ausgang übernommen, und dort bleibt er für alle Zeiten (d. h. bis zum nächsten RESET). Damit ist jeder Zugriff auf das EPROM unmöglich gemacht. Gleichzeitig wird das EPROM über den $\overline{CS}$-Eingang in den stromsparenden Standby-Modus geschickt. Allerdings ist jetzt das eine Tor des Multiplexers '155 immer offen (abgesehen von Refresh-Zyklen), und so spricht ein Zugriff im unteren Adreßbereich statt des ROMs nun immer den RAM-Speicher an. Im oberen Speicherbereich ist das sowieso der Fall. Auf diese Weise kann man mit Hilfe eines I/O-Zugriffs nach einem RESET das EPROM ausblenden und hat dann einen Computer mit vollen 64 KByte RAM vor sich.
Damit wäre auch gleich die Beschaltung des EPROMs erklärt. Bleibt nur noch zu sagen, daß man mit PL1 zwischen den EPROM-Typen 2732, 2764 und 27128 wählen kann (s. Tabelle 1). Das 2732 ist so in die (ihm zu große) Fassung einzusetzen, daß die Anschlüsse 1, 2, 27 und 28 frei bleiben.

DRAM

Kommen wir nun zur Ansteuerung der dynamischen RAMs. Wahlweise können 64K- oder 256K-Typen eingesetzt werden. Leider erfordert der zweite Typ etwas mehr Aufwand für die Refresh-Steuerung, doch dazu später.
Das $\overline{RAS}$-Signal für den Speicher wird direkt von $\overline{MREQ}$ abgeleitet. Es kündigt die erste (die untere) Hälfte der Speicheradresse an. Kurze Zeit später werden die Multiplexer '157 und '153 umgeschaltet und liefern den zweiten Teil der Adresse. Diese Verzögerungszeit läßt sich mit PL2 einstellen und wird durch Gatterlaufzeiten erzeugt. Dann wird der Eingang G1b des '155 durchgeschaltet und eines der Signale $\overline{CAS0}$ bis $\overline{CAS3}$ aktiviert. Dieses Signal teilt der angesprochenen Speicherbank mit, daß jetzt auch die zweite Adreßhälfte stabil anliegt. Die Verzögerungszeit ist mit PL2 einstellbar. Die Pegel an den Eingängen SEL1 und SEL2 des '155 bestimmen, welche der Speicherbänke 0..3 angewählt wird. Sie werden von den Adreßleitungen A18 und A19 gesteuert. Jede der Speicherbänke kann maximal 256 KByte groß sein. So ergibt sich ein Adreßbereich von bis zu 1 MByte! Auf der Platine können aus Platzgründen allerdings nur acht Speicherchips, also entweder 64 oder 256 KByte bestückt werden. Man sollte deswegen aber noch lange nicht zu der billigeren (und damit kleineren) Speicherversion greifen. Zum einen sind auch 256er DRAMs mittlerweile zu zivilen Preisen zu erhalten, und zum anderen lohnt sich diese Investition nicht erst bei einer späteren Implementation von CP/M 3.0, sondern von der ersten Sekunde an - mehr dazu im Software-Teil diese Artikels.

Getrickstes Auffrischen

Ein bißchen komplizierter ist schon die Steuerung des Refreshs der DRAMs. Aber da hilft, wie schon erwähnt, die CPU kräftig mit. Nach jedem Op-Code-Fetch, also immer dann, wenn sie mit der Dekodierung eines gerade geladenen Befehls beschäftigt ist, gibt die Z80 eine Refreshadresse auf der unteren Hälfte des Adreßbusses aus. $\overline{RFSH}$ und $\overline{MREQ}$ werden dabei aktiviert. Mit $\overline{MREQ}$ übernehmen die RAMs die Adresse und führen den Refresh durch. CPU-intern wird in einem Register die zuletzt ausgegebene Refreshadresse gespeichert und vor der nächsten Ausgabe erhöht. Leider ist die von der CPU gelieferte Adresse aber nur sieben Bit breit. Das achte Bit ist immer Null. Das schadet zwar bei den meisten 64K-Chips nicht, doch 256k-Chips benötigen einen acht-Bit-Refresh! Also muß der CPU-interne Zähler extern um ein Bit erweitert werden. Das geschieht mit FF3 und FF4. FF3 übernimmt am Ende eines jeden $\overline{RFSH}$ den Pegel des siebten Adreßbits. Das Flip-Flop bildet also einen Zwischenspeicher für diese Stelle des Refreshzählers. FF4 dagegen ist als Teiler geschaltet. An seinem Ausgang steht das gewünschte achte Bit zur Verfügung. Der Dekoder schaltet es auf den Eingang A7 der RAMs, sobald $\overline{RFSH}$ aktiv ist sowie die niedere Hälfte des Adreßbusses anliegen soll.

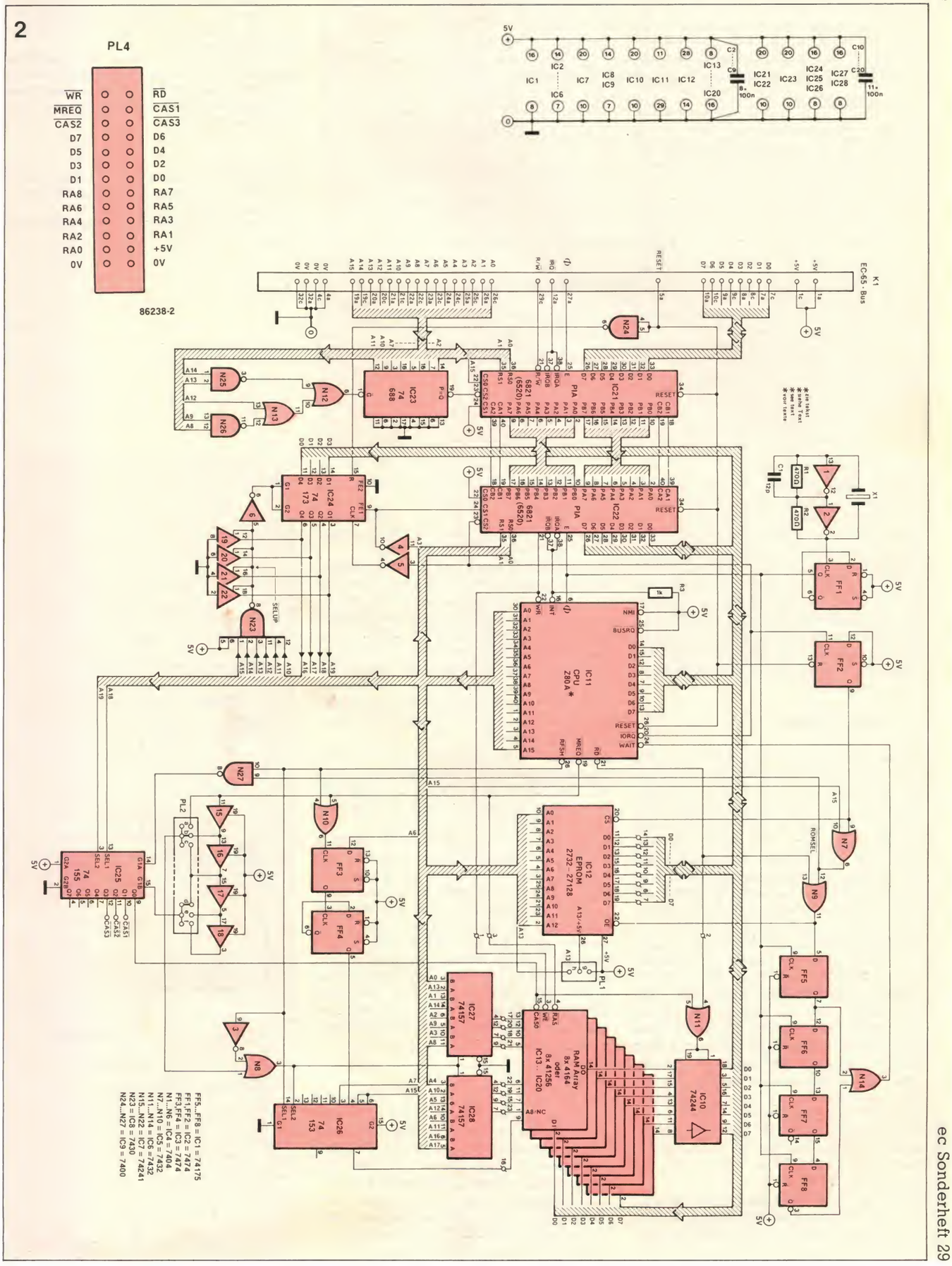
MMU

Wir haben schon weiter oben gesehen, daß die CPU in unserer Schaltung bis zu 1 MByte Speicher adressieren kann. Dazu sind 20 Adreßleitungen nötig. Die Z80 selbst stellt aber nur 16 davon zur Verfügung. Die restlichen vier werden auf der Karte erzeugt und erlauben es, zwischen insgesamt 16 Speicherbänken zu je 64 KByte (oder besser: 63 KByte, siehe unten) umzuschalten. Schauen wir uns nun an, wo die zusätzlichen Adreßleitungen herkommen.
Links unten im Schaltplan (Bild 2) befindet sich ein '173. Das ist ein vierfaches D-Flip-Flop mit Tri-State-Ausgängen. An seinen Eingängen liegen die Datenleitungen D0..D3, an den Ausgängen die neuen Adreßleitungen A16..A19. Die Flip-Flops übernehmen die Daten am Eingang genau dann, wenn sie eine positive Flanke am T-Eingang erhalten und außerdem beide Eingänge FE1 und FE2 LOW sind. FE2 ist das immer, FE1 genau dann, wenn A3 HIGH ist. Man kann also Daten in dieses Flip-Flop schreiben, indem man sie z.B. an den I/O-Kanal 8 sendet. So setzt man (per Programm) die vier höchstwertigsten Adreßleitungen!
Um zu verhindern, daß der Rechner sich "aufhängt", wenn so plötzlich die Speicherbank umgeschaltet und ihm damit der Boden unter den Füßen weggezogen wird, existiert ein sogenannter 'COMMON'Bereich. Das ist ein Speicherbereich, der nicht mit den höchstwertigen Adreßleitungen umgeschaltet werden kann. Er ist bei uns 1 KByte groß und befindet sich bei $FC00..$FFFF. Er kann auch dazu benutzt werden, Daten von einer 64K-Bank in die andere zu transportieren. Dieser COMMON-Bereich wird mit N23, N6 und N19..N22 realisiert. Liegt eine der Adressen aus dem COMMON-Bereich auf dem Bus, so sind die Adreßleitungen A10 bis A15 logisch eins. N23 schaltet durch und über N6 werden die Ausgänge der Adreß-Flip-Flops in den hochohmigen Zustand geschaltet. Gleichzeitig werden N19..N22 aus diesem Zustand zurückgeholt und funktionieren als ganz normale Treiber. Ihre Eingänge liegen auf Masse und damit werden auch die Adreßleitungen A16..A19 auf LOW geschaltet, unabhängig vom Zustand der Flip-Flops im '173.
Bild 4 verdeutlicht die Bedeutung des COMMON-Speichers.

FIFO

Damit bleibt jetzt nur noch der linke Teil des Schaltplanes, der das Interface zwischen EC-65 und Z80 darstellt. Viel gibt es dazu allerdings nicht zu sagen. Es handelt sich in erster Linie um zwei PIAs, von denen jeweils ein Port als Ein- und der andere als Ausgang programmiert wird. Das Schaltbild zeigt die (programmierte) Datenrichtung. Außerdem werden die

Bild 2. Die gesamte Schaltung mit CPU, ROM, RAM, MMU und I/O paßt auf eine einzige Euro-Platine.



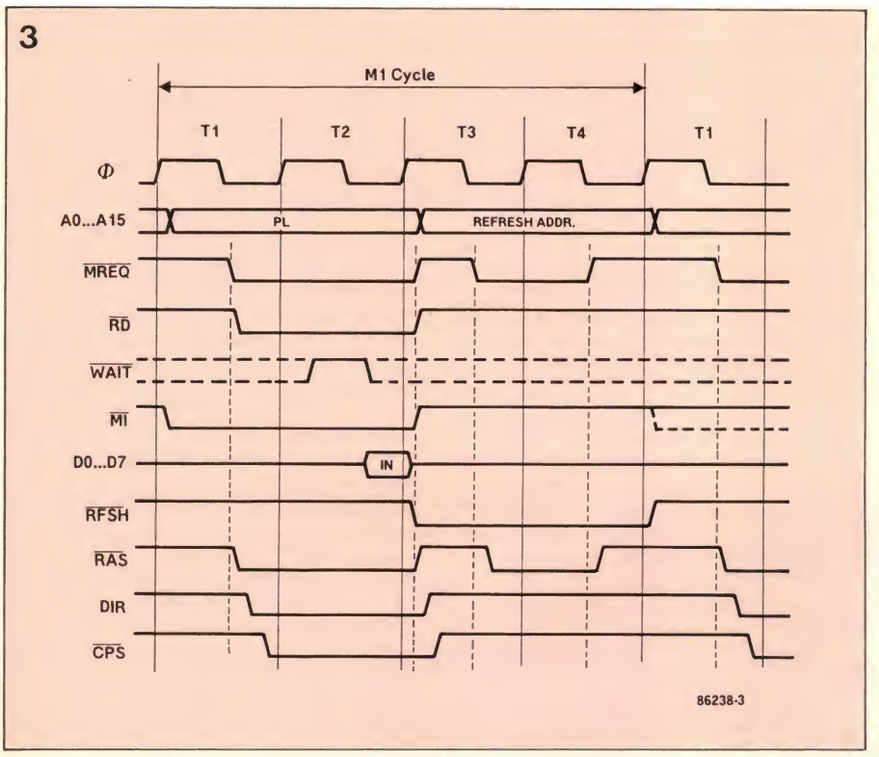


Bild 3. Das Timing-Diagramm zur Z80-Karte.

Bild 4. So sieht die Speicheraufteilung aus, wenn die Platine mit 256 KByte bestückt ist. Am oberen Ende befindet sich der COMMON-Speicher. Hier muß sich die CPU befinden, wenn sie die Speicherbänke umschalten will. Sonst sägt sie sich selbst den Ast ab, auf dem sie sitzt.
Das EPROM wird in alle Speicherbänke dupliziert. Das macht aber nichts, weil es kurz nach einem RESET sowieso ganz abgeschaltet wird. Dann hat die CPU nur noch Zugriff auf's RAM.

Anm. d. Red.:
Wie das Leben so spielt: Die Eingangserwähnte RAM-Disk ist bereits fertig und implementiert — siehe den Software-Teil dieses Heftes.
Und für die Grafik-Zusatz-Software war in diesem Heft kein Platz mehr, obwohl...

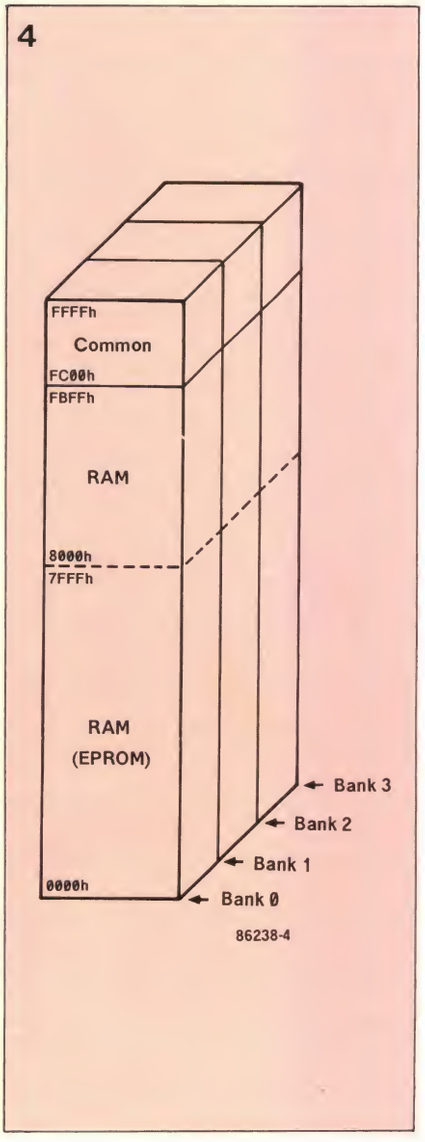
Handshakeleitungen genutzt, damit die beiden CPUs wissen, ob am Eingang neue Daten anliegen bzw. ob die Daten am Ausgang schon übernommen wurden. Das als "Adreßdekoder" geschaltete Gatter N4 sorgt dafür, daß die PIA als Ausgangskanal immer dann angesprochen wird, wenn das '173 nicht selektiert wird. Die zur Z80 gehende PIA belegt die I/O-Kanäle 0 bis 3. Die PIA auf der 6502-Seite die Adressen $E300 bis $E303. Hier werden die Gatter N25, N26, N12 und N13 sowie der Vergleicher '688 als Adreßdekoder verwendet.

4, 6 oder 8 MHz?

Das ist die Frage, die man sich vor dem Aufbau der Platine stellen sollte. Eine Z80 mit 4 MHz-Takt ist Standard. Mit 6 MHz laufen schon eine ganze Menge weniger Rechner. Und 8 MHz ist so ziemlich das höchste, was man im Moment von einer Z80 verlangen kann. Wir wollen uns hier nicht mit Benchmarks herumschlagen, die sowieso wenig aussgagekräftig sind (Vielleicht tun wir das etwas ausführlicher und exakter in einem der nächsten ec-Hefte). Aber eine Zahl wollen wir doch in den Raum stellen, auch wenn sie mit sehr viel Vorsicht zu genießen ist: MBASIC unter CP/M ist auf einer Z80-CPU mit 8 MHz bei Arithmetikroutinen (SIN, COS, LOG, EXP) mehr als doppelt so schnell wie das Microsoft-BASIC des EC-65 unter OSI-DOS bei einem 6502-Takt von 2 MHz! Tabelle 2 zeigt, welche Bausteine man für welche Taktfrequenz einkaufen sollte. Besonders kritisch im Timing sind dabei die CPU selbst sowie die DRAMs. Hier sollte man auf jeden Fall der Empfehlung in der Tabelle nachgehen. Weniger kritisch sind da die TTLs. Aber auch hier gilt: sicher ist sicher.
Wenn Sie trotz allem noch Schwierigkeiten mit hohen Taktfrequenzen haben, kann das auch daran liegen, daß das Taktsignal selbst nicht steil genug ist. In diesem Fall ist es zweckmäßig, FF1 (IC 2) auszutauschen. Notfalls kann man es auch durch einen F-Typ ersetzen.
Falls Sie Ihren EC-65 mit 2 MHz-Takt betreiben, müssen Sie für IC21 einen 68B21 oder einen 6520A einsetzen.
Daß es sich hierbei nur um Richtwerte handelt, beweist die Z80-Karte, auf der dieser Artikel entworfen wurde. Sie arbeitet nämlich bei 8 MHz klaglos und absolut betriebssicher mit normalen LS-TTLs und 150ns-DRAMs. Das liegt aber an Bauteiletoleranzen und ist keinesfalls allgemeingültig! Dennoch können Sie ruhig zuerst mal Ihre langsameren RAMs einsetzen, wenn Sie vom letzten Einkauf noch welche übrig haben.
Wenn Sie sich dagegen sowieso neue besorgen müssen, empfehlen wir, sich für die schnelleren Typen zu entscheiden. Damit können Sie sich unter Umständen eine Menge Ärger ersparen.
Wenn wir schon mal dabei sind, soll hier auch nicht verschwiegen werden, daß den DRAMs in unserem Schaltungsentwurf einiges zugemutet wird. Das sogenannte Precharge-Timing ist hier nämlich arg an der Grenze (Es handelt sich dabei um die Zeit, die ein DRAM nach einem Zugriff "in Ruhe gelassen werden" möchte, um die zuletzt aktivierte Reihe oder Spalte zu "refreshen"). Ein negativer Effekt wird aber durch die hohe Refreshrate (höher als 1500 mal pro Sekunde) verhindert.

Aufbau

Der Aufbau der Platine erfolgt im Wesentlichen so, wie bei allen anderen Projekten auch: Zuerst wird die Platine selbst auf Fehler geprüft (Dazu reichen Ihre Augen vollkommen aus!). Dann geht's an die Bestückung: zuerst Widerstände und Kondensatoren, dann Fassungen und Anschluß- sowie Steckerleisten. Zum Schluß wird noch der Quarz bestückt.
Besondere Beachtung verdienen die für



die DRAMs nötigen Block-Kondensatoren. Da nämlich die Stromversorgung dieser Bausteine vor allem beim Betrieb mit hohen Taktfrequenzen recht kritisch ist, haben wir uns kurzerhand entschlossen, sie **direkt** an die Versorgungsspannungs-Pins der ICs zu löten. Das ist die von der technischen Seite gesehen beste Lösung und kommt damit der Betriebssicherheit der Platine entgegen. Natürlich verlangt diese Methode etwas mehr Aufmerksamkeit bei der Bestückung, aber wir haben uns gedacht, daß das für jemand, der schon einen ganzen EC-65 zusammengebaut hat, eigentlich kein großes Problem sein dürfte.
Die Stromversorgung geschieht bei den DRAMs über die Pins 8 und 16. Wer ganz auf Nummer sicher gehen will, der benutzt Tantal-Elkos, muß dabei aber auf die Polung achten: Masse liegt an Pin 16 und +5V an Pin 8 — also genau entgegengesetzt wie bei TTL-ICs.
Die Kondensatoren werden am besten von unten an die Fassungen gelötet. Es ist angebracht, ihre Anschlußdrähte vorher zu isolieren: Dazu benutzt man entweder kleine Stückchen Schrumpfschlauch oder die Kunststoffummantelung von normalem Schaltdraht. Wer's lieber unkonventionell haben möchte, der kann natürlich auch einfach Klebeband verwenden.
Wem dagegen die Methode des "Dranlötens" zu unprofessionell erscheint, der kann sich auch Sockel besorgen, in die der Blockkondensator schon gleich mitintegriert ist. Industriell wird dieser Weg schon länger beschritten.
Vor dem ersten Test werden dann noch drei Kurzschlußstecker auf die Leisten PL1 und PL2 gesetzt. Über PL1 gibt Tabelle 1 Auskunft. PL2 wird zuerst einmal wie im Schaltplan bestückt.
Da es auf unserem Platinchen ganz schön eng hergeht und die Leiterbahnen darauf einigermaßen grazile Gebilde sind, möchten wir Sie bitten, mit einem feinen Lötkolben nicht zu hoher Leistung an's Werk zu gehen - den Bräter, mit dem Sie gestern noch die Dachrinne repariert haben, lassen Sie getrost im Keller.
Sind Sie soweit fertig mit der Bestückung, werfen Sie noch einmal einen stolzen Blick auf das vollbrachte Werk: Wenn Sie jetzt einen zu groß geratenen Lötkleks oder eine kalte Lötstelle finden, so erspart Ihnen das nachher eine Menge Ärger.

Test, Test, Test

Jetzt ist es Zeit, die Platine ein erstes Mal in den EC-65 einzustecken. Rechner anschalten und irgendein Programm laufen lassen. Verhält sich der Rechner nicht wie normal, so haben Sie wohl doch irgendeinen Kurzschluß übersehen und sollten schleunigst danach Ausschau halten.

Danach überprüfen Sie bitte die Betriebsspannung sämtlicher ICs. Hat auch dieser Test geklappt, so kann man nunmehr IC21, IC23, IC6 und IC9 in ihre Fassungen einsetzen. Wenn Sie die Karte jetzt in den EC-65 einschieben, muß sich die PIA über die Speicherstellen $E300 bis $E303 programmieren lassen.
Als nächstes werden IC2 und IC4 bestückt. Wer einen Frequenzzähler oder ein Oszilloskop sein Eigen nennen darf, der prüft nach, ob der Takt auch wirklich vorhanden ist.
Jetzt endlich wird die Platine fertig aufgebaut, indem man alle restlichen ICs in ihre Fassungen einsetzt. Beim Einschalten nimmt man den Ausgang Q von FF2 unter die Lupe (Logik-Prüfstift oder Spannungs-Meßgerät - nur Snobs benutzen dazu ein Scope). Nach einem RESET muß sich an diesem PIN ein konstanter HIGH-Pegel befinden. Das ist das Zeichen dafür, daß (mit einiger Wahrscheinlichkeit) das Betriebsprogramm vom EPROM in's RAM kopiert wurde und dort brav seine Runden dreht.
Funktioniert das trotz eines fehlerlosen Aufbaus nicht, so ändern Sie die Belegung von PL2. Damit wird das DRAM-Timing eingestellt.

Tabelle 1. So paßt man mit Hilfe von PL1 die verschiedenen EPROM-Typen an die Schaltung an.

Tabelle 2 zeigt, welche Bausteintypen bei welcher Taktfrequenz eingesetzt werden sollten. Bei den TTLs sind die Angaben nur als Anhaltspunkte zu bewerten. Alle anderen Bausteine dagegen sollte man unbedingt wie angegeben dimensionieren, um sicher zu gehen.

Bild 5. Ein kleines Testprogramm in BASIC hilft bei der Inbetriebnahme der Schaltung. Es wird der Inhalt des EC-65 Monitor-EPROMs in den Z80-Speicher kopiert und dann wieder zurückgelesen: Sind die Daten nach dem Hin- und Herschaufeln identisch, ist mit großer Wahrscheinlichkeit alles in Ordnung.

Tabelle 1

EPROM	Verbindung
2732	1-2
2764	keine
27128	2-3

Tabelle 2

Bauteil	4 MHz	6 MHz	8 MHz
IC5, IC6	74LS32	74LS32	74F32
IC11	Z80A	Z80B	Z80H
IC13...20 DRAMs	200 ns	150 ns	120 ns
IC22	6821	68A21	68B21
oder	6520	6520A	6520A
IC25	74LS155	74LS155	74F155
IC26	74LS153	74F153	74F153
IC27, IC28	74LS157	74F157	74F157
X1, Quarz	8 MHz	12 MHz	16 MHz

5

```
1 PRINT CHR$(27);"1";"    TESTING EC-65 Z8Ø-BOARD": PRINT
2 GOTO 1ØØ: REM skip header
3 :
1Ø **********************************************
2Ø *                                            *
3Ø *  Test Program FOR EC-65 Z8Ø Board at $E3ØØ  *
4Ø *                                            *
5Ø *       Written 1986  by  Frank Schmidt       *
6Ø *                                            *
7Ø **********************************************
8Ø :
1ØØ REM === Initialisations ===
11Ø :
115 PRINT "... Initializing"
12Ø REM PIA Addresses
13Ø AD=58112: AC=AD+1: BD=AD+2: BC=AD+3
14Ø EPROM=6144Ø: REM start of EPROM
15Ø GOSUB 92ØØ: REM init PIA
16Ø POKE 23,79: REM Set terminal width for 8Ø columns
17Ø Z=Ø: REM counter for compare fails
18Ø :
19Ø :
2ØØ REM === Move EPROM data to Z8Ø ===
21Ø :
215 PRINT "... Moving Data"
22Ø A=1:   GOSUB 9ØØØ: REM Command: get 65Ø2-data
23Ø A=16:  GOSUB 9ØØØ: REM Store data from $1ØØØ on
24Ø A=Ø:   GOSUB 9ØØØ
25Ø A=INT(4Ø96/256): GOSUB 9ØØØ: REM Handle 4 KBytes
```

Software-Hilfe

Haben Sie Ihrem Platinchen diese ersten Schritte erfolgreich beigebracht, so können Sie mit einem letzten Test einigermaßen sichergehen, daß alles in Ordnung ist. Dazu tippen Sie das kleine BASIC-Programm in Bild 5 ein. Bevor Sie es ungeduldig starten, sollten Sie es aber zuerst einmal auf Diskette abspeichern.

Ist dies geschehen, so "RUNnen" Sie Ihren EC-65 und harren der Dinge, die da kommen sollen: Nach einer kleinen Meldung werden auf dem Bildschirm eine Menge Punkte ausgegeben: für jedes Byte, das zur Z80 geschickt wird, einer. Ist das nicht der Fall, so klappt entweder das Handshaking mit der Platine nicht, oder Sie haben das Programm falsch eingetippt.

Danach erscheint noch einmal eine kurze Meldung, und der EC-65 versucht, die Daten wieder zurückzulesen. Sie werden dann mit dem Original verglichen. Ist alles in Ordnung, wird pro Byte wiederum ein Punkt ausgegeben. Stimmen die Daten nicht überein, erscheint stattdessen ein Ausrufezeichen. Zum Schluß wird noch die Anzahl der Bytes angegeben, die nicht übereingestimmt haben. Erscheint hier eine Null, so können Sie sich in aller Ruhe dem zweiten, dem Software-Teil dieses Artikels widmen...

Stückliste:

Widerstände:
R1, R2 = 470
R3 = 1k

Kondensatoren:
C1 = 12p
C2...C20 = 100n

Halbleiter:
IC1 = 74LS175
IC2, IC3 = 74LS74 (#)
IC4 = 74LS04
IC5, IC6 = 74LS32 (*)
IC7 = 74LS241
IC8 = 74LS30
IC9 = 74LS00
IC10 = 74LS244
IC11 = Z80A (*)
IC12 = EPROM 2732 od. 2764 od. 27128 max. 400ns
IC13...IC20 = 4164 oder 41256, 200ns (*) (#)
IC21, IC22 = 6821 oder 6520 (*) (#)
IC23 = 74LS688
IC24 = 74LS173
IC25 = 74LS155 (*)
IC26 = 74LS153 (*)
IC27, IC28 = 74LS157 (*)

außerdem:
X1 = Quarz 8 MHz (*)
64polige Messerleiste nach DIN 41612
PL1 = Pfostenfeldstifte, einreihig, gerade, 3polig
PL2 = Pfostenfeldstifte, zweireihig, gerade, 12polig
PL4 = Pfostenfeldstifte, zweireihig, gerade, 26polig
3 Kurzschlußstecker (Jumper)

Platine EPS 86238

(*) gilt für 4 MHz-Version, siehe auch Tabelle 2
(#) siehe Text

```
26Ø A=4Ø96-256*INT(4Ø96/256) :GOSUB 9ØØØ
27Ø FOR I = EPROM TO EPROM+4Ø95
275 :  PRINT ".";
28Ø :  A=PEEK(I): GOSUB 9ØØØ
29Ø NEXT: PRINT: PRINT
3ØØ :
31Ø :
5ØØ REM === Return data from Z8Ø and compare ===
51Ø :
515 PRINT "... Comparing Data ('!' means 'no match')
52Ø A=3:   GOSUB 9ØØØ: REM Command: hand data to 65Ø2
53Ø A=16:  GOSUB 9ØØØ: REM Hand data from $1ØØØ on
54Ø A=Ø:   GOSUB 9ØØØ
55Ø A=INT(4Ø96/256): GOSUB 9ØØØ: REM Handle 4 KBytes
56Ø A=4Ø96-256*INT(4Ø96/256) :GOSUB 9ØØØ
57Ø FOR I = EPROM TO EPROM+4Ø95
58Ø :  GOSUB 91ØØ: REM get byte from Z8Ø
59Ø :  IF A=PEEK(I) THEN PRINT ".";: GOTO 61Ø
6ØØ :      PRINT "!";: Z=Z+1
61Ø NEXT
62Ø A=255: GOSUB 9ØØØ
63Ø PRINT
64Ø PRINT "... End of Test: ";Z;" compare fails
65Ø :
66Ø :
67Ø END
68Ø :
69Ø :
9ØØØ REM === Output (A) to Z8Ø Port ===
9Ø1Ø :
9Ø2Ø REM Wait until last byte accepted
9Ø3Ø IF PEEK(BC)<128 GOTO 9Ø3Ø
9Ø4Ø B=PEEK(BD): REM reset PBC flag
9Ø5Ø POKE BD,A : REM store data
9Ø6Ø RETURN
9Ø7Ø :
9Ø8Ø :
91ØØ REM === Read byte from Z8Ø Port to A ===
911Ø :
912Ø REM Wait until PIA contains new data
913Ø IF PEEK(AC)<128 GOTO 913Ø
914Ø A=PEEK(AD): REM store PIA data in A
915Ø RETURN
916Ø :
917Ø :
92ØØ REM === Init PIA on Z8Ø Board ===
921Ø :
922Ø POKE AC,4Ø: POKE AD,Ø:   POKE AC,44
924Ø POKE BC,4Ø: POKE BD,255: POKE BC,44
925Ø POKE BD,255
926Ø FOR I=1 TO 2Ø
927Ø :  A=PEEK(AD): NEXT
928Ø RETURN
```

Technische Daten

* Euro-Platine für den EC-65 (EC-65K)
* Belegt vier Adressen ($E300 bis —E303)
* Z80-CPU mit 4, 6 MHz oder 8 MHz
* Adreßbereich: 1 MByte
* DRAM-Speicher auf der Platine: 64 oder 256 KByte
* RAM über Erweiterungsstecker auf 1 MByte aufrüstbar
* DRAM-Timing wird durch Gatterlaufzeiten erzeugt und ist mit Steckleisten einstellbar
* EPROM: wahlweise 2732, 2764 oder 27128 (4, 8 oder 16K)
* EPROM wird nach dem Booten ausgeblendet
* WAIT-Logik erlaubt den Einsatz von preisgünstigen 350ns-EPROMs selbst bei 8 MHz Taktfrequenz
* Datenaustausch der CPUs ist extrem schnell durch die Verwendung zweier PIAs als FIFO

6

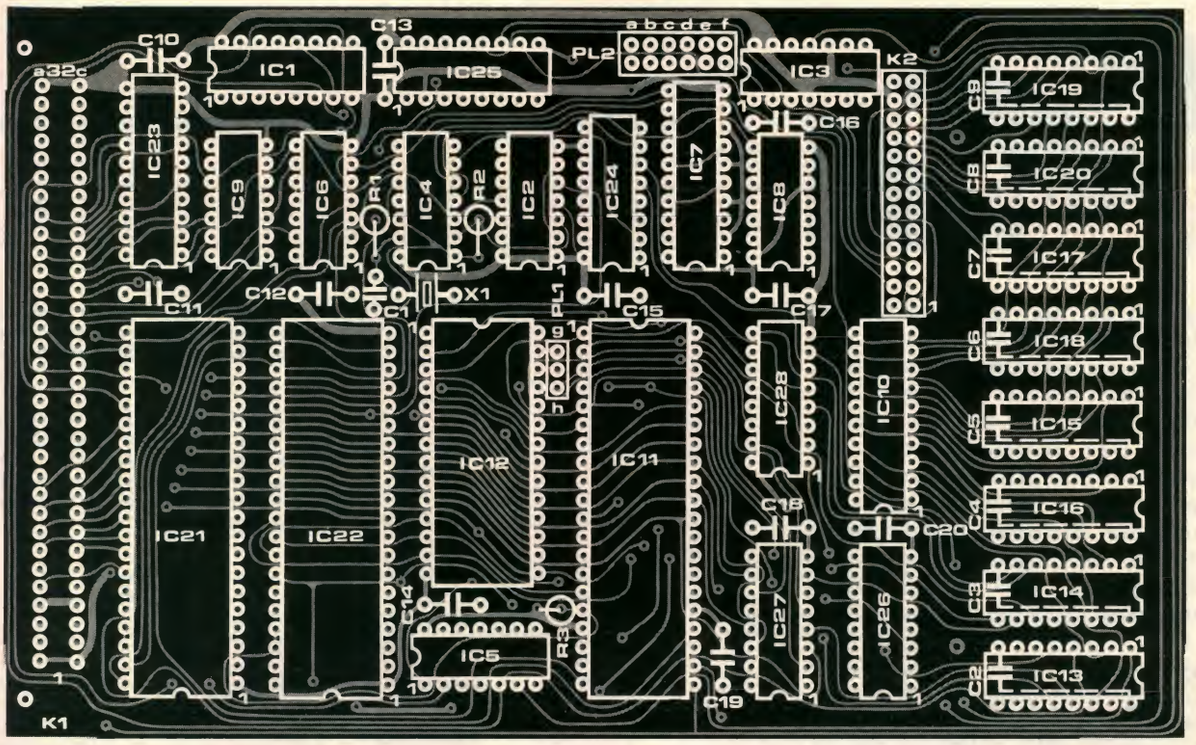


Bild 6. Der Bestückungsplan der Z80-Karte. Man sieht: Es geht ganz schön eng zu.

Anzeigen

Der Elektor-FLEX-Computer
E C — 6 8

Wir liefern Ihnen die Software:
— 2 Systemdisketten (40 und 80 Tr.) mit Source-Code der Treiber
— Diverse Software verschiedener Hersteller (Pascal, Cobol, C, BASIC...)

Unser Angebot wird ständig erweitert und auf unserem Original — EC — 68 getestet

EDV-Systeme & Peripherie

Werner Schorstein
Dipl.-Ing.

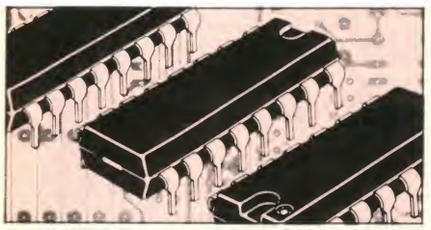
Pankratiusstraße 11
6100 Darmstadt
Tel. (0 61 51) 7 43 50
Telex 4197 280 edvs d

Microcomputer *EC-68K* von Böhm
Kuhlenstr. 130-132 · D-4950 Minden
Telefon 0571/5045-0

So urteilt die Fachpresse:
"Wenn Sie einen Computer besitzen wollen, von dem Sie bisher nur träumen konnten; wenn Sie mit Software arbeiten wollen, die fast alle auf dem Markt befindlichen Software-Pakete in den Schatten stellt: Dann ist der "EC-68K" genau der richtige Computer. Wollen Sie auch in die Welt der 16/32-Bit-Multiprozessor-Computer einsteigen, weil Sie Angst haben, der Zug hierfür ist abgefahren, dann müssen Sie so schnell wie möglich diesen fantastischen Computer bauen. Zwei Prozessoren, ein 68008 und ein 6809, machen den "EC-68K" zu einem "Formel-I-Computer", was die verarbeitungsgeschwindigkeit betrifft. Doch noch nicht genug! Damit man in Assembler bequemer als in einer höheren Programmiersprache programmieren kann, sind Fließkomma-Arithmetik und Ein/Ausgabeprozeduren als zusätzliche Mnemonics eingebaut. Das Operatingsystem des "EC-68K" wurde nicht für den Computer, sondern für die Anwender geschrieben, die mit diesem modernen 16/32-bit-Computer arbeiten oder spielen möchten."
(Elector Computing, Ausg. 3)

- ***Super-Hardware*** - modular ausbaufähig, individuelle Anpassung an Ihre Wünsche.
- ***Super-Software*** - anwendungsorientiert, Programmiersprache nach Wunsch.
- ***Super-Preis*** - Sie werden staunen! Informationen noch heute anfordern.

der flexible 16-Bit-Computer im Bausatzsystem



Mäuse . . .

. . . womit wir konkret Computermäuse meinen, nicht die grauen Nager

Bernd Fasel

Wer mit der Zeit gehen will, muß mindestens eine Maus in — pardon, an seinem Computer haben. Leider fehlt derzeit an vielen Computern die entsprechende Schnittstelle. Das betrifft sowohl die Hardware-Schnittstelle zur Auswertung der Maus-Signale als auch die Software-Schnittstelle, die die Verträglichkeit mit Betriebssystem und Programmbibliotheken garantieren muß.

Prinzipiell sind beim Anschluß einer Maus zwei Wege gangbar: Eine Auswertung der Maus-Signale per Software unter Verwendung eines einfachen Parallelports oder ein spezieller Peripherieprozessor, welcher sich ausschließlich um die Maus kümmert. Beide Verfahren haben ihre Vor- und Nachteile. Die Software-Lösung ist in erster Linie billig, hat aber den Nachteil, daß der Mikroprozessor doppelt beansprucht wird: Einmal zum Umrechnen der Maus-Signale in absolute Koordinaten, und zum anderen für die anschließende Darstellung auf dem Bildschirm. Die Routinen an sich sind nicht sehr aufwendig: Das Problem ist, daß diese Routinen unter Umständen sehr oft aktiviert werden müssen, um ein kontinuierliches Nachführen des MausSymbols auf dem Schirm (und in der entsprechenden Software) zu erreichen. Wird der Prozessor für andere Aufgaben benötigt, zum Beispiel für Diskettenzugriffe, bleiben alle Bewegungen der Maus unerkannt, was unter dem Gesichtspunkt der Benutzerfreundlichkeit natürlich nachteilig ist.

Im Gegensatz dazu ist ein Peripherie-Prozessor nur damit beschäftigt, dauernd die aktuellen Mauskoordinaten zu ermitteln. Es besteht also unter allen Umständen ein fester Zusammenhang zwischen der Wegstrecke auf dem Arbeitstisch und der entsprechenden Abbildung auf dem Bildschirm. Die Abbildung selbst wird zwar vom zentralen Prozessor übernommen, aber das ist jetzt nicht mehr nachteilig: Das Maussymbol springt auf alle Fälle an die Bildschirmposition, die der "gefahrenen" Wegstrecke entspricht. Falls der Prozessor anderweitig beschäftigt ist, kommt es lediglich zu einer Zeitverzögerung. Leider ist dies aber eine etwas teurere Lösung.

Der Maus-Pheripherieprozessor

Die Schaltung, um die es im Folgenden zunächst geht, ist ein reiner Hardwareprozessor. Obwohl ursprünglich für 68.000-Systeme ausgelegt, wurde das Businterface so fortentwickelt, das eine problemlose Adaption an praktisch alle asynchro-

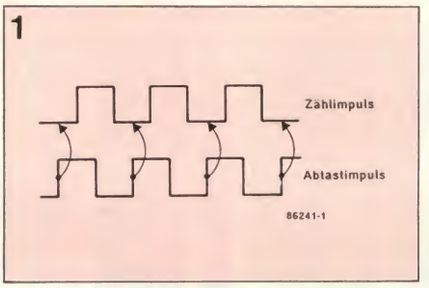
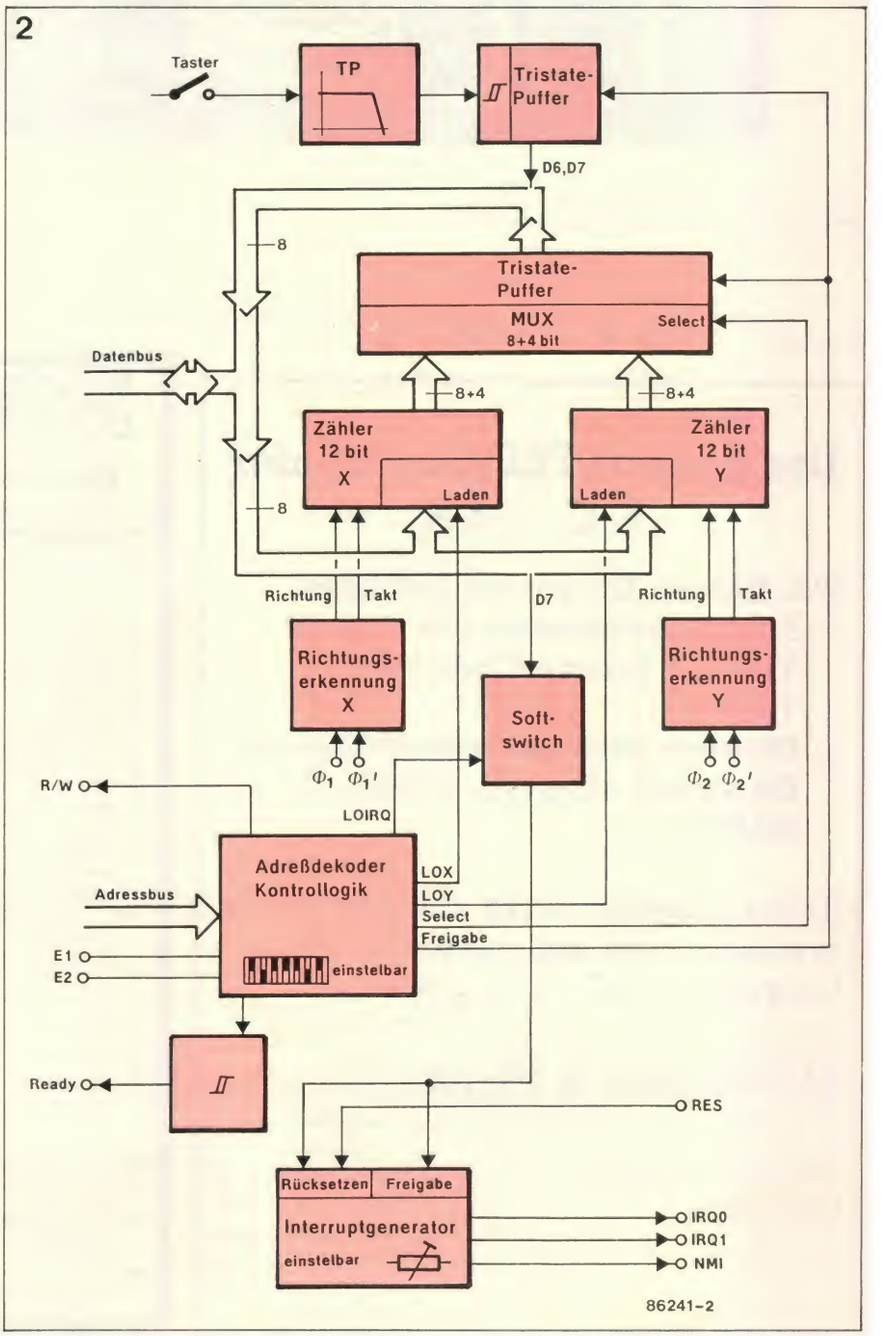


Bild 1. Das Impulsschema verdeutlicht, wie durch Vergleich zweier Taktsignale die Bewegungsrichtung der Maus erkannt wird.

Bild 2. Aus dem Blockschaltbild wird deutlich, daß zwei 12-Bit-Zähler den Kern des Maus-Interfaces bilden. Um diese herum gruppieren sich Richtungserkennung, MUX, Adreßdekoder, Interrupt-Generator und Tasterauswertung.



3

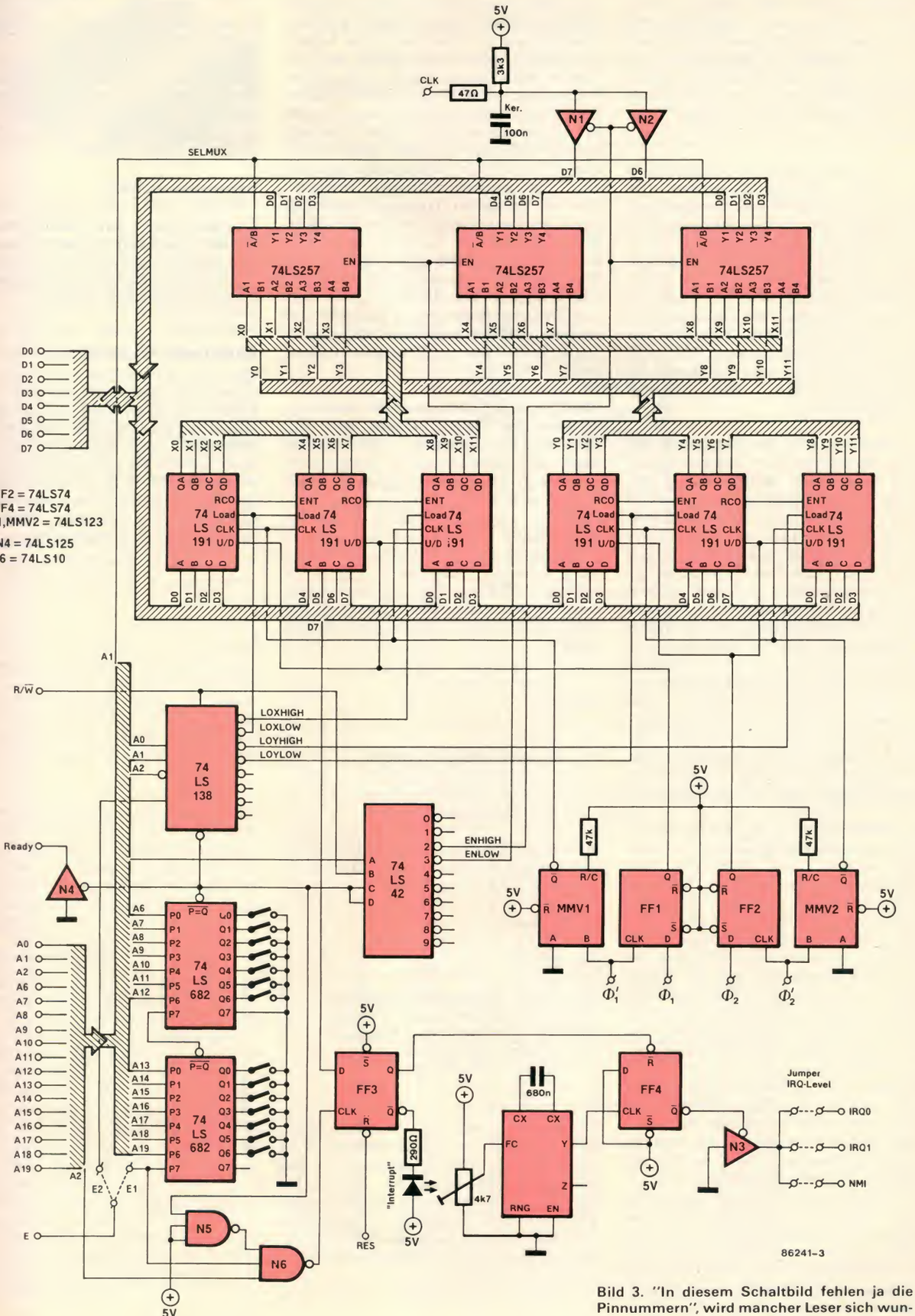


Bild 3. "In diesem Schaltbild fehlen ja die Pinnummern", wird mancher Leser sich wundern. Richtig: Wir haben Sie weggelassen, weil manche Pinnummern erst beim Layouten der Platine festgelegt werden können. Wer die Schaltung jetzt schon wrappen, fädeln oder sonstwie aufbauen will, muß also diesmal zum Datenbuch greifen.

nen Busse (also zum Beispiel 68.000) und synchronen Busse (zum Beispiel 6502) möglich ist. Eine Platine, die direkt auf den Bus des EC-68K gesteckt und darüber hinaus durch ein Adapter-Feld an praktisch jeden Euro-Karten-Computer angepaßt werden kann, ist in Vorbereitung. Der Zugriff erfolgt in allen Fällen byteweise.

Zur Funktion der Schaltung: Wie schon in dem Artikel "Interface-Techniken und Ergonomie" in diesem Heft beschrieben, gibt die Maus für die X-Richtung und für die Y-Richtung je zwei Pulssignale aus. Aus dem Phasenverhältnis der Ausgangsimpulse läßt sich direkt auf die Bewegungdrichtung schließen. Die Phase ist entweder +90° oder -90°. Das bedeutet, daß mit einem einfachen D-Flipflop die Richtung erkannt werden kann. Man muß lediglich mit der einen Impulskette die andere abtasten, wie in Bild 1 schematisch dargestellt.

Die Gesamtfunktion der Schaltung läßt sich am einfachsten dem Blockschaltbild (Bild 2) entnehmen. Neben den Bewegungssensoren verfügen die meisten Mäuse über einen oder mehrere Taster. Die vorliegende Schaltung wurde mit der Apple-Maus erprobt, die in Verbindung mit dem "Mac" und dem Apple IIc eingesetzt wird (die Maus ist auch einzeln erhältlich). Und damit sind wir auch beim Schaltbild (Bild 3) angelangt: Der Taster der Apple-Maus schaltet nach Masse. In Verbindung mit dem 3k3-Pull-Up-Widerstand ganz oben im Schaltbild bedeutet also eine logische 1: Taster nicht gedrückt. Zur Entprellung dient ein aus einer R-C-Kombination bestehendes Tiefpaßfilter. Danach gelangt das Signal über zwei Tristate-Datenbustreiber mit Schmitt-Trigger-Charakteristik auf den Datenbus.

Die Richtungssignale Φ1, Φ1' für die X-Koordinate und Φ2, Φ2' für die Y-Koordinate werden zuerst durch die Richtungserkennung "bearbeitet". Hier finden Sie das schon erwähnte D-Flipflop (74LS74) und zusätzlich ein Monoflop zur Zählimpulsformung. Dieses dient der Verzögerung der positiven Zählflanke, um die Timing-Vorschriften für die Zähler 74LS191 zu erfüllen. Die LS191 sind 4-Bit-Binärzähler, bei denen die Zählrichtung über den Eingang U/D (Up/Down) gewählt werden kann. Die Zähler bedürfen einer Ruhezeit von minimal etwa 20 ns zwischen einer Änderung des U/D-Signales und dem nächsten Zähltakt. Die Zeitkonstante des Monoflops ist daher völlig unkritisch. Je drei Zähler werden zu einer Einheit kaskadiert, so daß die Auflösung für jede Koordinate 12 bit = 4096 Werte beträgt. Das reicht sicher auch für eine sehr hochauflösende Grafik. Die Kaskadierung erfolgt über die Anschlüsse RCO (ripple carry output, Pin 13) und ENT (count enable input, Pin 4). Die Zähler arbeiten also synchron, alle Schieberegister haben das gleiche Taktsignal. Daher sind Laufzeiteffekte ausgeschlossen. Über die Multiplexer mit Tristate-Ausgängen (74LS257) ist der Zählerstand direkt vom Datenbus lesbar. Je nach Zustand des A/B-Anschlusses kann entweder die X- oder die Y-Koordinate gelesen werden. Die Zähler können auch direkt durch Schreibzugriffe mit einem Startwert besetzt werden. Dadurch läßt sich der maximal und minimal mögliche Zählerstand durch Software begrenzen, was die direkte Abbildung auf die Bildschirmkoordinaten erleichtert. Um die Einbindung in ein bestehendes Software-System zu erleichtern, wurde in der Schaltung noch ein Interrupt-Generator vorgesehen. Dieser besteht im Wesentlichen aus einem spannungseinstellbaren Oszillator, der mit konstanter Frequenz Interrupts generiert. Die gewünschte Priorität (IRQ0, IRQ1, NMI) kann durch einen Stekker gewählt werden. Die Funktion des Interrupt-Generators wird durch das IRQ-Flag festgelegt. Mit IRQ-Flag = Low wird eine Interrupt-Anforderung gelöscht und weitere Interrupts verhindert. IRQ-Flag = High gibt den Interrupt-Generator wieder frei.

Bleibt noch der Adreßdekoder. Die Basisadresse des Maus-Interfaces wird mit zwei 8-Bit-Komparatoren 74LS682 dekodiert. Der Adreßvergleich startet aber erst, wenn das Freigabesignal E1=High ist. Die gewünschte Adresse ist mit DIL-Schaltern einstellbar. Externe Pull-Up-Widerstände werden nicht benötigt, da der 74LS682 an den Q-Eingängen intern entsprechende Widerstände nach +5 V auf dem Chip hat. Die folgende Logik sorgt für die Selektierung der Registeradressen. Es ergibt sich folgende Aufteilung:

Lesen

A1	A0	
0	0	X-Koordinate Highbyte
0	1	X-Koordinate Lowbyte
1	0	Y-Koordinate Highbyte
1	1	Y-Koordinate Lowbyte

Bei "Lesen Highbyte" steht in D6 und D7 der Zustand des Maustasters, D5 und D4 enthalten keine Information.

Schreiben

A2	A1	A0	
0	0	0	X-Koordinate Highbyte
0	0	1	X-Koordinate Lowbyte
0	1	0	Y-Koordinate Highbyte
0	1	1	Y-Koordinate Lowbyte
1	x	x	Schreibe IRQ-Flag

Das IRQ-Flag wird von D7 geschrieben. Jeder Zugriff auf Mausadressen wird durch Ready = Low signalisiert. Das zweite Freigabe-Signal E2 ist nur für Schreibzugriffe zuständig. E2 muß dann High sein. Es ist besonders darauf zu achten, daß zu dem Zeitpunkt, an dem E2 wieder Low wird, die Schreibdaten noch stabil sein müssen!

Anpassungen

Die Schaltung kann unter Verwendung der Anschlüsse E1, E2 und Ready an praktisch alle Standard-Bus-Systeme angepaßt werden.

Anpassung an EC-68K-Bus:

E1 und E2 werden zusammen mit E verbunden. Ready wird nicht benutzt. Für die DTACK-Generierung wird die dafür vorgesehene Schaltung auf der CPU-Platine des EC-68K eingesetzt. Ansonsten stimmen alle Signalbezeichnungen im Schaltplan mit der Pinbelegung des EC-68K-Busses überein.

Anpassung an 6502:

E1 = High, E2 = Low, Ready wird nicht benutzt. Die Mausadressen können hier voll ausdekodiert werden, indem A3.. ..A15 mit den Eingängen des Adreßkomparators verbunden werden.

Anmerkung der Redaktion:

Wie schon vom Autor erwähnt, ist eine Platine für diese Schaltung in Entwicklung. Geht alles nach Plan, wird sie in ec 6 vorgestellt. Die Software-Entwickler unter unser Lesern können auf der Grundlage dieses Artikels schon einmal beginnen, Maus-Software für die verschiedenen Computer zu entwickeln - denn damit wären wir zeitlich überfordert.

Ein 64-K-DRAM-Zusatzspeicher

Der Adreßbereich des EC-65 wird (für die Grafik) verdoppelt!

Paul Lavigne
Denis Meyer

Aus alt mach neu! Die Elektor-Leser erinnern sich noch daran, wie aus der 16-K-DRAM-Karte eine 64-K-DRAM-Karte geworden ist. Heute zeigen wir, wie man dieselbe Karte als 64-K-Zusatzspeicher umrüsten kann. Besonders bei Anwendern der Grafikkarte ist eine solche Speichererweiterung sicher hochwillkommen. Es handelt sich hier aber nicht um eine sogenannte Bank-Umschaltung ("bank switching"), die noch einige Probleme mit sich bringt, sondern um eine ausgefeilte Lösung, die den Zugriff auf die 64 KByte der Speichererweiterung ganz einfach über drei Adressen aus dem bestehenden Speicher ermöglicht.

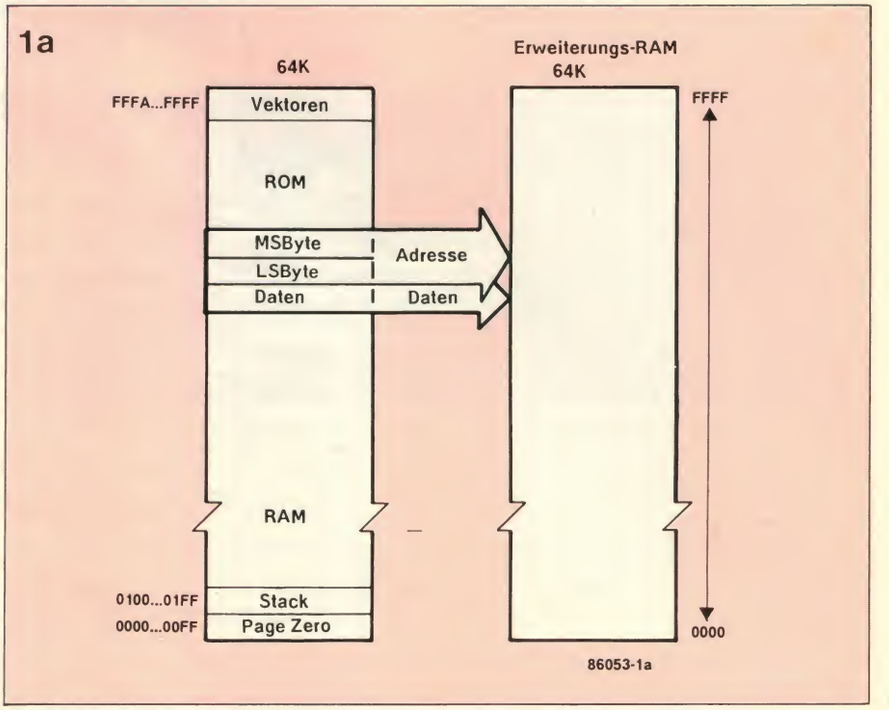
Drei Adressen, das sind also drei Bytes: Eins davon ist das hochwertige Byte der Adresse, auf die man im neuen 64-K-Raum zugreifen will. Ein weiteres ist das niederwertige Byte derselben Adresse. Beide bilden also zusammen einen 16-Bit-Zeiger, dessen Wert zwischen 0000 und FFFF liegt.

Das dritte Byte ist natürlich das Datenwort, das man in den Zusatzspeicher schreiben oder lesen will (Bild 1). Im alten Speicherbereich legt man diese drei Bytes irgendwo in den I/O-Bereich, damit man vom bestehenden RAM-Bereich kein einziges Byte hergegeben werden muß.

Mittelbarer Zugriff

Bemerkenswert ist bei diesem Prinzip, daß die CPU selbst eigentlich nie wirklich in den Zusatzspeicher kommt. Sie kann nur indirekt darauf zugreifen, indem sie die gewünschte Adresse und das dazu gehörende Datenwort in einen Zwischenspeicher schreibt oder daraus liest. Es handelt sich also um einen Zusatzspeicher mit indirektem Zugriff (Bild 2). Für manche Applikationen ist das zweifellos ein Nachteil, für andere dagegen eine elegante und feine Lösung. Keine Schwierigkeiten mit Page Zero, keine Probleme mit dem Stack, auch das ROM mit den Sprungvektoren für die CPU bleibt wo es ist, und der Zusatzspeicher bietet die vollen 65.536 neuen Speicherzellen und keine einzige weniger.

Im Einsatz zusammen mit der Grafikkarte bringt der Zusatzspeicher einiges an Bedienungsfreundlichkeit (wenn nicht sogar Luxus!) mit sich. Als "Grafikfreak" haben Sie doch sicherlich schon etwas von dem Videopuffer gelesen, in dem die



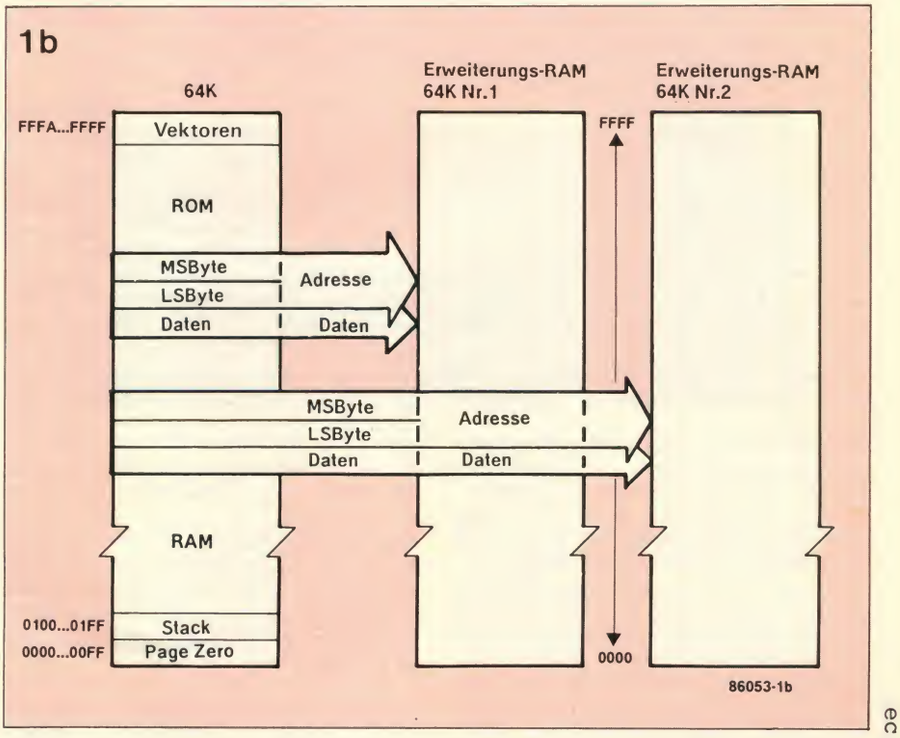


Bild 1a. Drei Adressen aus dem I/O-Bereich des ursprünglichen 64K-Blockes arbeiten als Schleuse zu einem neuen 64K-Block, der von $0000 bis $FFFF ausschließlich für Daten benutzt werden kann.

Bild 1b. Auch mehrere 64-K-Zusatzspeicher können nebeneinander addressiert werden. In der Theorie könnte die Anzahl dieser Blöcke als fast unbegrenzt betrachtet werden.

ausgeführten Zeichenbefehle gespeichert werden können. Der Zweck dieser Zwischenspeicherung ist das schnelle Wiederholen von komplexen Figuren, nachdem alle Punktkoordinaten einmal berechnet worden sind.

Leider stößt man gerade bei komplexen Figuren ziemlich schnell auf die Grenzen der Speicherkapazitat eines 56-K-RAM-Bereichs, in dem schon die Grafiksteuersoftware selbst, der BASIC-Interpreter und auch noch das BASIC-Programm untergebracht worden sind. Aus dieser Not ist die Idee entstanden, einen neuen Speicher aufzubauen, und ihn als Videopuffer zu reservieren. Damit haben wir eine vorbildliche Applikation für diese Speichererweiterung. Es sind immerhin noch viele anderen denkbar.

Einfach und billig

Der neue Speicher ist nicht weniger und kaum mehr als die gute alte DRAM-Karte von Elektor. Da inzwischen die meisten EC-65-Nachbauer ihr System mit statischem RAM ausgerüstet haben, liegen wohl einige DRAM-Karten herum. Hoffentlich haben Sie Ihre Karte nicht weggeschmissen! Jetzt ist die Zeit gekommen, wo sie wieder gute Dienste leisten kann. Dazu kommt nur noch eine kleine Anpassungsschaltung mit vier ICs, die so auf eine Aufsteckplatine gesetzt werden, daß es noch nicht einmal nötig ist, Drahtverbindungen mit der Speicherkarte herzustellen.

Den rechten Teil der Schaltung in Bild 3 dürfte jeder erkennen: Das ist der 64-K-DRAM-Speicher, der auch im ersten ec-Sonderheft auf Seite 31 zu sehen war. Im linken Teil: die neue Adreßdekodierung, die Multiplexfunktion (PROM 82S123) und die drei oben erwähnten Zwischenspeicher.

Der dritte Zwischenspeicher erlaubt es, zugleich Lese- und Schreibinstruktionen wie INCrement und DECrement zu benutzen: die CPU ist dadurch in der Lage, den Inhalt der zwei Adressenspeicher über das dritte Register zurückzulesen, ihn zu ändern (z.B. inkrementieren oder dekrementieren) und wieder zurückzuschreiben. Es müssen also keine Pointer gesetzt werden, und die Steuersoftware wird dementsprechend einfacher.

Geändert wird nur wenig. Die alte Adreßdekodierungsschaltung verschwindet, ebenso die Adreßmultiplexerschaltung. Um die Verdrahtung zu vereinfachen, wird der Datenbustreiber IC 20 von der Speicherkarte heruntergenommen, er wird aber wieder auf die Aufsteckplatine gesetzt. Auf den dadurch freigewordenen IC-Sockel wird die Aufsteckplatine mittels zwei Reihen Lötstiften gesteckt. Die IC-Sockel von IC 9 und IC 10, die jetzt auch frei sind, dienen demselben Zweck. Alle Signale laufen hierdurch über die IC-Sockel, sogar $\overline{EX}$, das von der Grafikkarte kommende Adreßdekodierungssignal. Die Änderungen, die auf der DRAM-Karte durchgeführt werden müssen, sind alle nochmals in Bild 4 eingetragen worden. IC 11, IC 21, IC 22 und R16 "verschwinden" von der DRAM-Karte. Das bedeutet, daß N45 ersetzt werden muß. Wie in Bild 2 zu sehen ist, hat N29 diese Funktion übernommen. Reine Kosmetik! Wichtiger sind die Änderungen, die rund um IC 9 durchgeführt werden müssen. Man unterbricht die Leiterbahn, die zu Pin 13 von IC 9 führt (A4), und verbindet diesen Pin mit Pin 31a am Buskonnektor (oder mit der Brücke 1-1'). Auch die Adreßleitung A6 wird von Pin 14/IC 9 getrennt, da jetzt das Adreßdekodierungssignal $\overline{EX}$ von der Grafikkarte über diesen Pin zur Aufsteckplatine geführt wird. Dafür haben wir beim EC-65 den noch unbenutzten Pin 2a des Buskonnektors verwendet. Wenn Ihnen das nicht paßt, können Sie sich selbst einen anderen auswählen. Wichtig ist nur, daß die Leitung auch tatsächlich frei ist, und daß Sie an beiden Enden denselben Pin benutzen. Nun ja, Dummheiten die bringt ja jeder fertig!

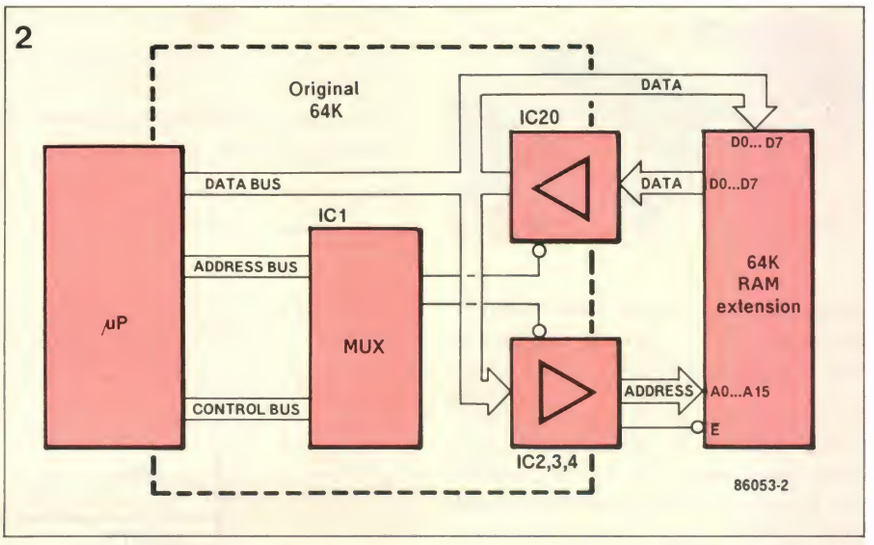


Bild 2. Zwischen CPU und 64-K-Erweiterung kommt eine kleine Schaltung, deren Funktion hauptsächlich darin besteht, den Datenbus der CPU in einen Adreßbus für die RAM-Karte umzuwandeln.

Beim EC-65K ist die Sache wegen des Umschalttaktes nicht ganz so einfach und auch noch nicht von uns erprobt. Folgende Hinweise seien denen gegeben, die es selbst versuchen wollen: Sie müssen jedenfalls ein konstantes 1MHz-Taktsignal mit der Phasenlage von Φ1 anstelle des jetzt ja ebenfalls der Umschaltung unterliegenden Φ1-Taktes verwenden. Prinzipiell ist hierzu das Taktsignal ΦF auf Busleitung 28a geeignet, es muß jedoch invertiert werden. Also zum Beispiel Brücke A-B entfernen und eine Drahtbrücke von N2/Pin 11 nach Busleitung 28a legen. Für das Signal $\overline{EX}$ kann man die Busleitung 29a verwenden: Die ist zwar eigentlich für das Signal ΦAUX reserviert, aber das haben wir auf der neuen CPU-Karte bewußt nicht auf den Bus geführt, damit eine Busleitung für Sonderzwecke wie diesen frei ist (man muß von einem bestimmten Lötpunkt auf der neuen CPU-Karte eine Drahtbrücke legen, wenn man es auf dem Bus haben will).

Das RAS/CAS-Multiplex-Signal wird von Pin 6/IC 8 nach Pin 1/IC 9 geführt. Außerdem muß Pin 15/IC 9 von Masse getrennt und mit Pin 10/IC 8 (N31) verbunden werden. Jetzt noch N45 durch N29 ersetzen, und dann haben wir's geschafft. Halt, nein: R6 muß auch weg!. Keine Mißverständnisse: Wenn hier die Rede ist von "IC 9" und "IC 10" ist, handelt es sich eigentlich um die dazugehörenden IC-Sockel und nicht die IC's selbst!

Genaue Arbeit

Beim Entwurf der Aufsteckkarte ist die Tatsache, daß auf einem Bus nur eine beschränkte Anzahl Konnektoren zur Verfügung stehen, sehr stark berücksichtigt worden. Die Speichererweiterung (das heißt die DRAM-Karte samt Aufsteckplatine) darf nicht mehr als einen Slot auf dem Bus besetzen. Da ist es ja sowieso schon knapp genug!
Deswegen empfehlen wir auch dringend die durch uns ausgetüftelte Bauweise. Aus den beiden Fotos und auf Bild 5 erkennt man alle Einzelheiten des richtigen Aufbaus. Weichen Sie davon ab, so gehen Sie das Risiko ein, einen zu hohen Aufbau zu erhalten und dadurch zwei Steckplätze auf dem Bus zu besetzen.

Genaue Arbeit ist auch beim Multiplexen der RAS- und CAS-Adressen nötig. Diese Funktion und die Adreßdekodierung übernimmt jetzt das bipolare PROM 82S123. Nehmen wir Tabelle 1 zur Hand. Da sehen wir, daß alle nicht gültigen Adressen durch das PROM herausgefiltert werden. Ist eine gültige Adresse angesprochen, dann muß zwischen Lese- und Schreiboperationen unterschieden werden. Dazu liegt die RAM-R/W-Leitung am Adreßeingang R1 des PROMs. Weiter unterscheidet der logische Pegel auf der S-Leitung am RØ-Adreßeingang zwischen den RAS- ($\overline{OELO}$) und den CAS-Adressen ($\overline{OEHI}$).

Nimmt man das $\overline{E17X}$-Signal - Pin 7 von IC 2 auf der Grafikkarte - als Adreßdekodierungssignal für die Speichererweiterung, so liegt das hochwertige Adreßbyte (WRHI) in E171 (und E175), und das nieder-

wertige Byte (WRLO) in E172 (und E176). Oder umgekehrt, das spielt eigentlich überhaupt keine Rolle. Das Datenwort ($\overline{RD}$) steht dann in E170 (und E174). Die Doppeladressierung in diesem Bereich entsteht dadurch, daß die Adressleitung A2 bei der Adreßdekodierung nicht verwendet wird. Wenn man 64 KByte dazubekommt, kann man wohl auf die drei Adressen verzichten. Oder hätten Sie vielleicht lieber ein IC mehr auf der Aufsteckplatine?

Falls Sie ein anderes Adreßdekodierungssignal als $\overline{E17X}$ verwenden, verschieben sich die oben erwähnten Adressen dementsprechend.

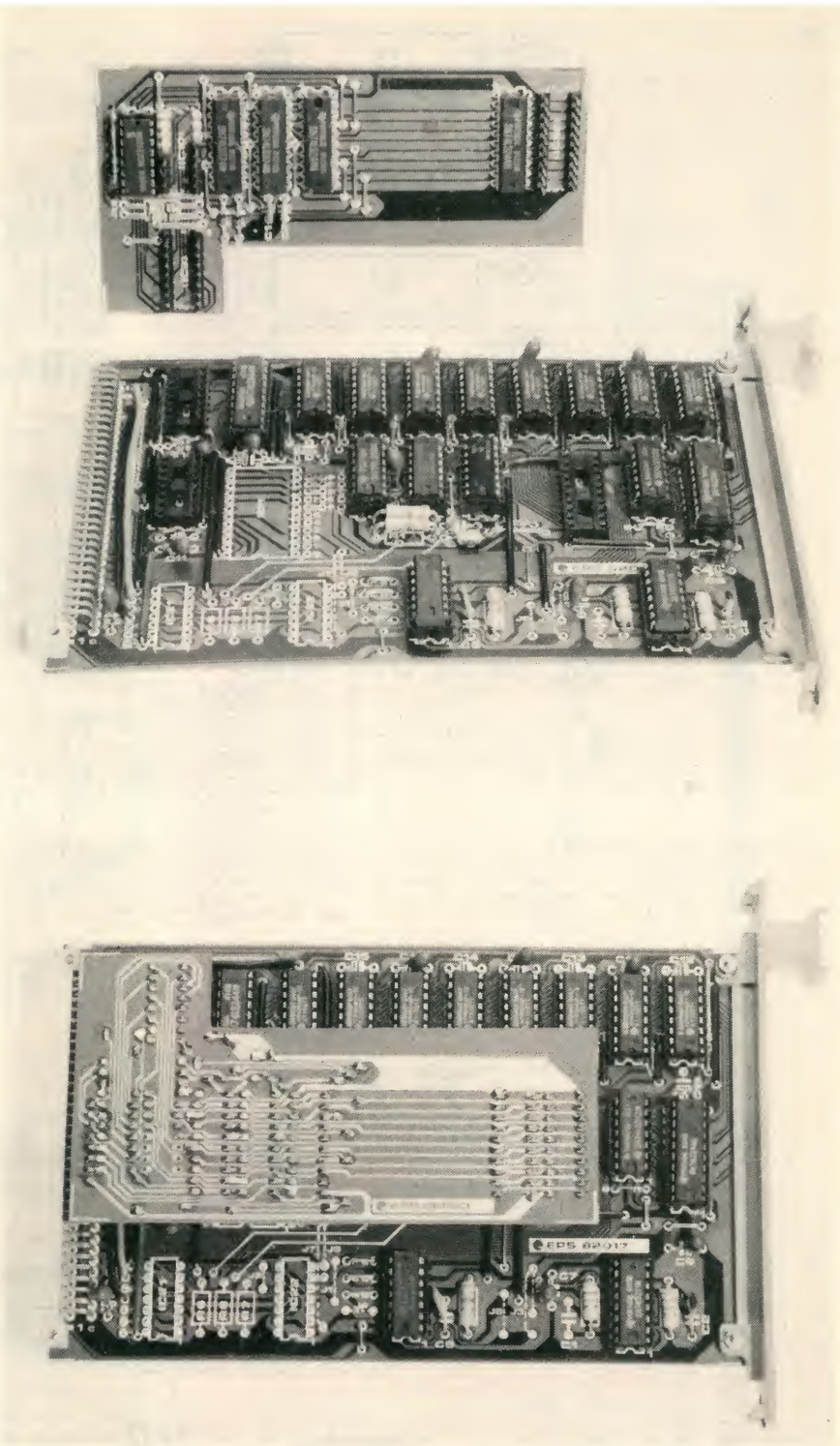
Erwähnen möchten wir nur noch, daß mehrere dieser Speichererweiterungen eingesetzt werden können. Bei jedem neuen Exemplar verlieren Sie zwar einige Bytes im ursprünglichen I/O-Bereich, dafür bekommen Sie aber jeweils weitere 65.536 dazu!

Die Ansteuerung

Es sollte nun jedem deutlich sein, daß die CPU nicht unmittelbar auf den Zusatzspeicher zugreifen kann. Sie muß erst die Adresse, auf welche sie im Zusatzspeicher lesend und/oder schreibend zugreifen will, in die dafür vorgesehenen Adreßspeicher setzen. Erst dann liest oder schreibt sie das Datenwort auf die vorprogrammierte Adresse. Diese Adreßregister bilden also eine Art Schleusenkammer zwischen den beiden 64-K-Bereichen: Der Datenbus der CPU dient als Adreßbus in der RAM-Erweiterung!
Dadurch wird die Ansteuerung der "Schleuse" durch die Software aber kaum komplizierter. Unter Umstanden kann sie sogar etwas einfacher sein. Auch mit den simpelsten PEEKs und POKEs kann auf den Zusatzspeicher zugegriffen werden.

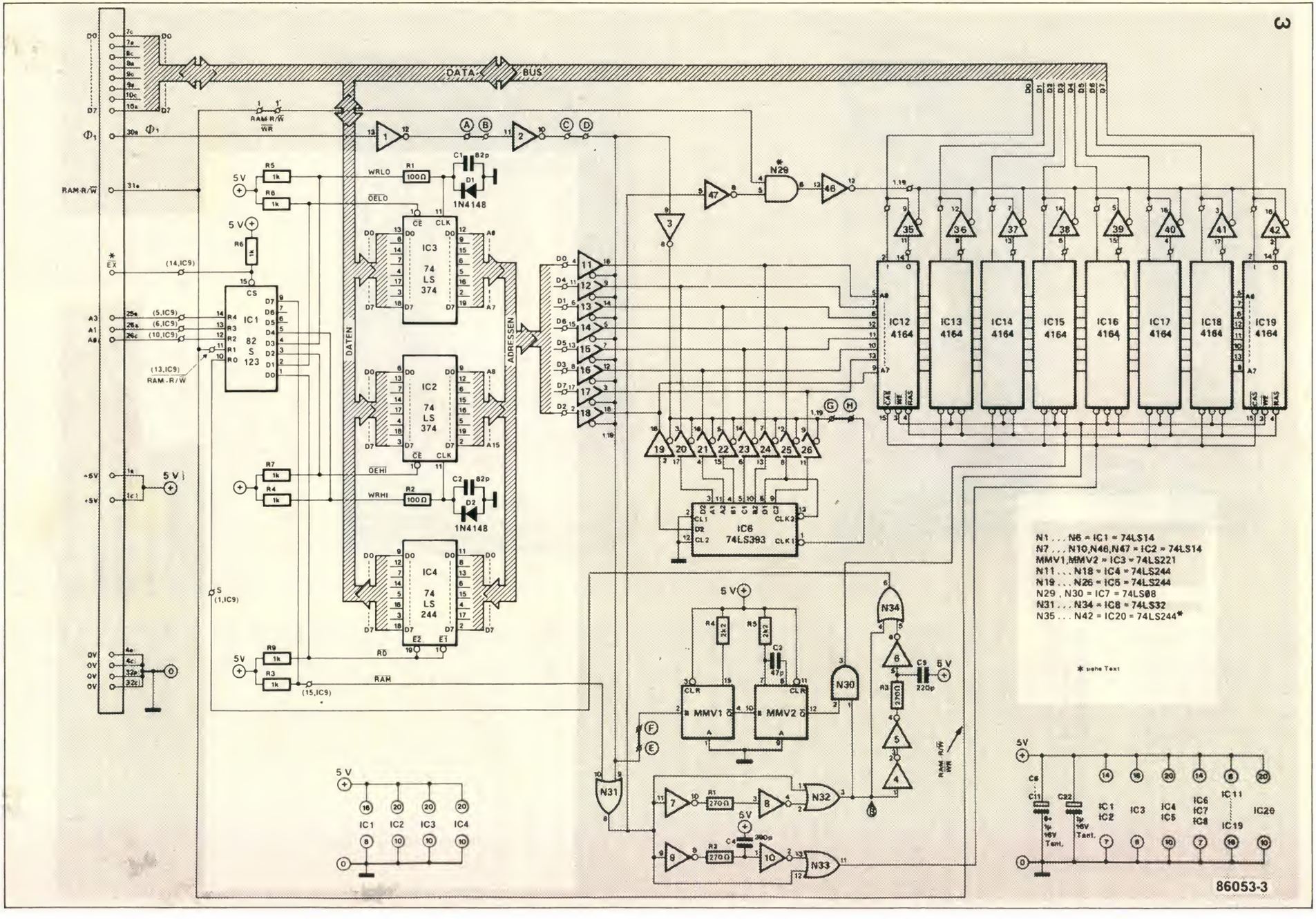
In unserer Applikation zum Beispiel wird der Zusatzspeicher als Videopuffer durch die Grafiksoftware voll geschrieben oder ausgelesen. Dann steht ein Byte neben dem anderen, ohne daß adreßmäßig hin und her gesprungen wird. Darum braucht man auch nur ganz unten bei Adresse $0000 anzufangen und nach jedem Zugriff die Adresse zu inkrementieren. Dabei wird üblicherweise das niederwertige Adreßbyte zuerst inkrementiert, und danach, wenn die Carry-Flag gesetzt ist, auch das hochwertige Adreßbyte. So einfach ist der Zugriff auf den neuen Speicher. Natürlich funktionieren in dem neuen Bereich weder Page Zero noch der Stack! Es handelt sich ja um eine Erweiterung des Datenspeichers und nicht des normalen Arbeitsspeichers.

In Tabelle 2 geben wir den Inhalt des Adreßdekodierungs- und RAS/CAS-Multiplex-PROMs der Aufsteckplatine wieder. Die Datenbits D5 und D6 werden nicht benutzt, sowie alle Adressen zwischen 10 und 1F. In Tabelle 3 sind die Änderungen der Grafiksoftware eingebracht. Es ist uns nicht möglich, das gesamte Listing hier nochmal auszudrucken. Daher werden Sie in dieser Tabelle nur die Stellen zurückfinden können, an denen für die neue Speicherkarte etwas geändert werden muß. Der Rest bleibt unverändert. Die drei Teile der Grafiksoftware müssen jedoch neu assembliert werden. Die angepaßte Source-File "64K VIDEO BUFFER" werden sich alle Anwender der Grafikkarte auf dem EC-65 auf der über den Software- Verteiler erhältlichen Diskette 24 näher ansehen können.



Hier noch eine kurze Zusammenfassung der Anpassungen:

- der 64K-VIDEO-BUFFER-Zusatzspeicher liegt in E170
- in der INIT0-Routine wird der neue 64K-Speicher initialisiert (Zeilen 1210 bis 1230
- die Routine ASTX (Zeile 4620) ist neu geschrieben: der Befehl CHR$(2) (früher: "öffne den Bildspeicher") ist jetzt ein Toggle-Befehl, der den Bildspeicher öffnet und schließt! Dadurch entfällt der Befehl CHR$(3). Solange dieser nicht für neue Zwecke benutzt wird, kann er in der CMDTAB-Tabelle stehen bleiben. Auf diese Weise braucht man dann in bestehenden BASIC-Programmen überhaupt nichts zu ändern. Also eventuell in Zeile 4720 nach dem existierenden Befehl


3
DATA BUS
DATEN
ADRESSEN
IC1 82 S 123
IC2 74 LS 374
IC3 74 LS 374
IC4 74 LS 244
IC6 74LS393
IC12 4164
IC13 4164
IC14 4164
IC15 4164
IC16 4164
IC17 4164
IC18 4164
IC19 4164
MMV1
MMV2
N1 . . . N6 = IC1 = 74LS14
N7 . . . N10,N46,N47 = IC2 = 74LS14
MMV1,MMV2 = IC3 = 74LS221
N11 . . . N18 = IC4 = 74LS244
N19 . . . N26 = IC5 = 74LS244
N29 , N30 = IC7 = 74LS08
N31 . . . N34 = IC8 = 74LS32
N35 . . . N42 = IC20 = 74LS244*
* siehe Text
86053-3

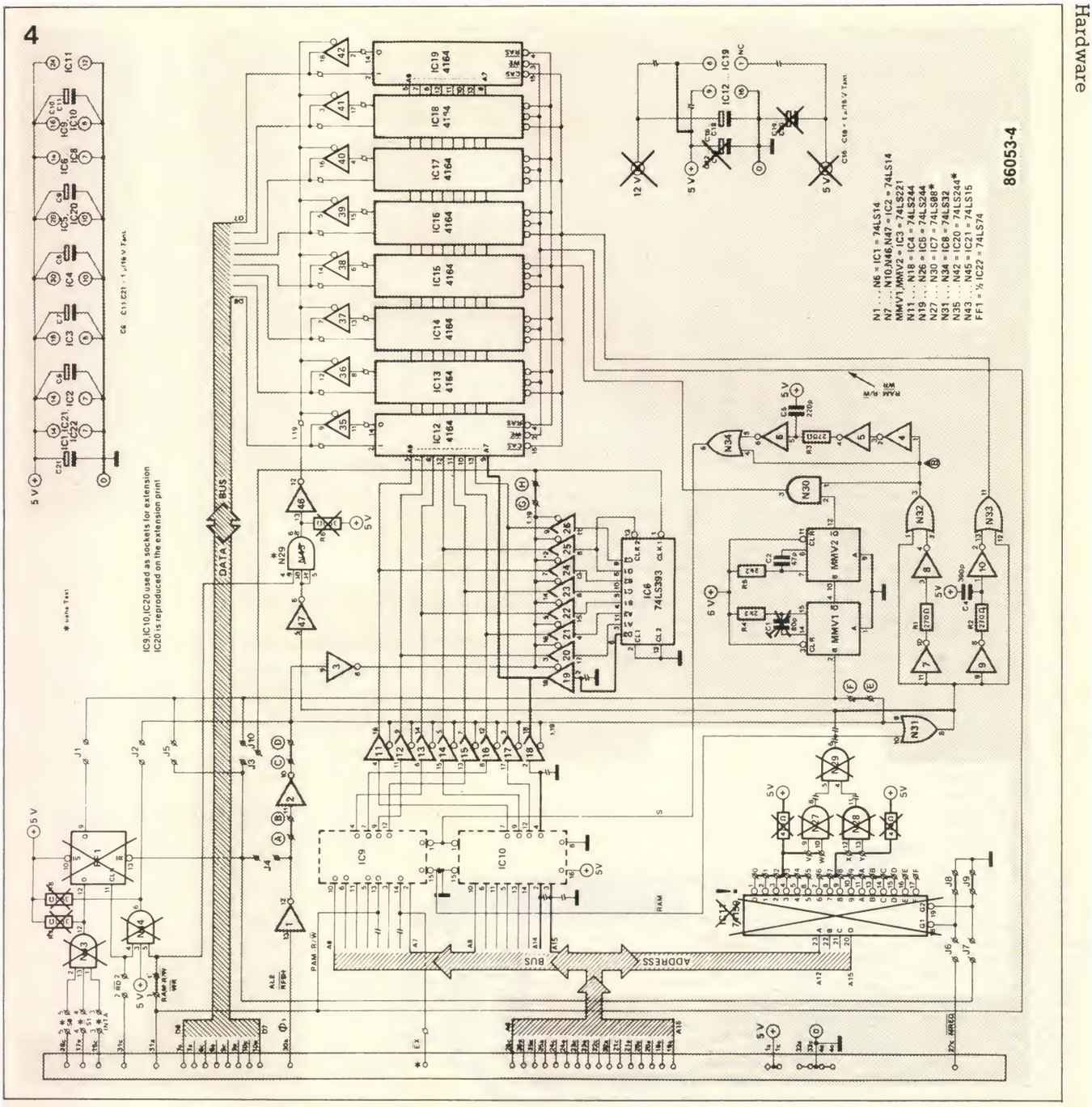


Bild 3. Den größten Teil dieser Schaltung kennt wohl jeder Elektor-Leser; der nicht gefärbte Teil stellt die eigentliche Neuerung dar: Ein PROM zur RAS/CAS-Multiplex-Steuerung und zur internen Adreßdekodierung, zwei Schreiberegister und ein Leseregister, das ist alles.

Bild 4. An der 16K-DRAM-Karte ist schon einiges geändert worden, um sie auf 64K umzurüsten. Hier müssen noch einige Bahnen umgelegt werden, damit die Aufsteckplatine mit dem PROM und den drei Latches direkt in die IC-Sockel von IC 9, IC 10 und IC 20 gesteckt werden kann.

ASTX, die FFFF (früher: ETX) durch einen zweiten ASTX ersetzen.

■ da die Routine SWAP nicht mehr nötig ist, wird auch das Programmsegment ab VIDTXT (Zeile 5590) viel einfacher. Auch die SOH-Routine für den Befehl CHR$(l) (Repeat Video Buffer) wird um die Hälfte kürzer!

■ und zum gutem Schluß: Sehen Sie sich doch mal an, wie einfach die Adreßverwaltungsroutine INCBUF ist.

Sind diese Änderungen im Programm SAMTEX eingebracht worden, dann müssen die Programme SAMGRA und SAMCIR auch noch an ihre neue Origin-Adresse angepaßt werden. Und dann kann es losgehen mit der rasenden Grafik. Bemerkenswert ist, daß die Software tatsächlich wesentlich schneller die Zeichenbefehle aus dem Zusatzspeicher holt, als sie es früher aus dem normalen Arbeitsspeicher tun konnte! Dazu kommt noch, daß man in volle 64 KByte auch einiges an komplizierten Zeichnungen stopfen kann.

Stückliste

Widerstände:
R1, R2 = 100 Ω
R3...R9 = 1 k

Kondensatoren:
C1, C2 = 82 p

Halbleiter:
D1, D2 = 1N4148
IC1 = 82S123 (ESS 548)
IC2, IC3 = 74LS374
IC4 = 74LS244

außerdem:
Platine EPS 86053

Tabelle 1. PROM zur Adreßdekodierung und $\overline{RAS}$/$\overline{CAS}$-Multiplexung

ADRESSEN							DATEN									
$\overline{EX}$	A3	A1	R/$\overline{W}$	S			$\overline{RAM}$	NC	NC	WRHI	WRL0	$\overline{OEHI}$	$\overline{OEL0}$	$\overline{RD}$		
CS	R4	R3	R2	R1	R0	HEX	D7	D6	D5	D4	D3	D2	D1	D0	HEX	Anmerkungen
1	X	X	X	X	X	– –	1	1	1	1	1	1	1	1	FF	Adresse
0	1	X	X	X	X	– –	1	X	X	1	1	1	1	1	9F	nicht
0	0	1	1	X	X	– –	1	X	X	1	1	1	1	1	9F	gültig (Anm. 1)
0	0	0	0	0	0	00	0	X	X	1	1	1	0	1	1D	WRITE
0	0	0	0	0	1	01	0	X	X	1	1	0	1	1	1B	XXX0 (XXX4): RAM
0	0	0	0	1	0	02	0	X	X	1	1	1	0	1	1D	READ
0	0	0	0	1	1	03	0	X	X	1	1	0	1	1	1B	(Anm. 2)
0	0	0	1	0	0	04	1	X	X	0	1	1	1	1	8F	WRITE
0	0	0	1	0	1	05	1	X	X	0	1	1	1	1	8F	XXX1 (XXX5): HIGH AD.
0	1	0	1	1	0	06	1	X	X	1	1	0	1	0	9A	READ
0	0	0	1	1	1	07	1	X	X	1	1	0	1	0	9A	(Anm. 3)
0	0	1	0	0	0	08	1	X	X	1	0	1	1	1	97	WRITE
0	0	1	0	0	1	09	1	X	X	1	0	1	1	1	97	XXX2 (XXX6): LOW AD.
0	0	1	0	1	0	0A	1	X	X	1	1	1	0	0	9C	READ
0	0	1	0	1	1	0B	1	X	X	1	1	1	0	0	9C	(Anm. 4)

Anm. 1: Entweder gehört die Adresse auf dem Bus nicht zum Block EX-X, oder X ist gleich 7...F
Anm. 2: Lesen oder Schreiben eines Datenwortes ins RAM.
RAM = 0 OEL0 = S OEHI = S
(XXX wird durch das externe Adreßdekodierungssignal EX bestimmt)
Anm. 3: Lesen oder Schreiben des hochwertigen Adreßbytes.
WRHI = R/W OEHI = R/W = R/W
(XXX wird durch das externe Adreßdekodierungssignal EX bestimmt)
Anm. 4: Lesen oder Schreiben des niederwertigen Adreßbytes.
WRL0 = R/W OEL0 = R/W RD = R/W
(XXX wird durch das externe Adreßdekodierungssignal EX bestimmt)

5

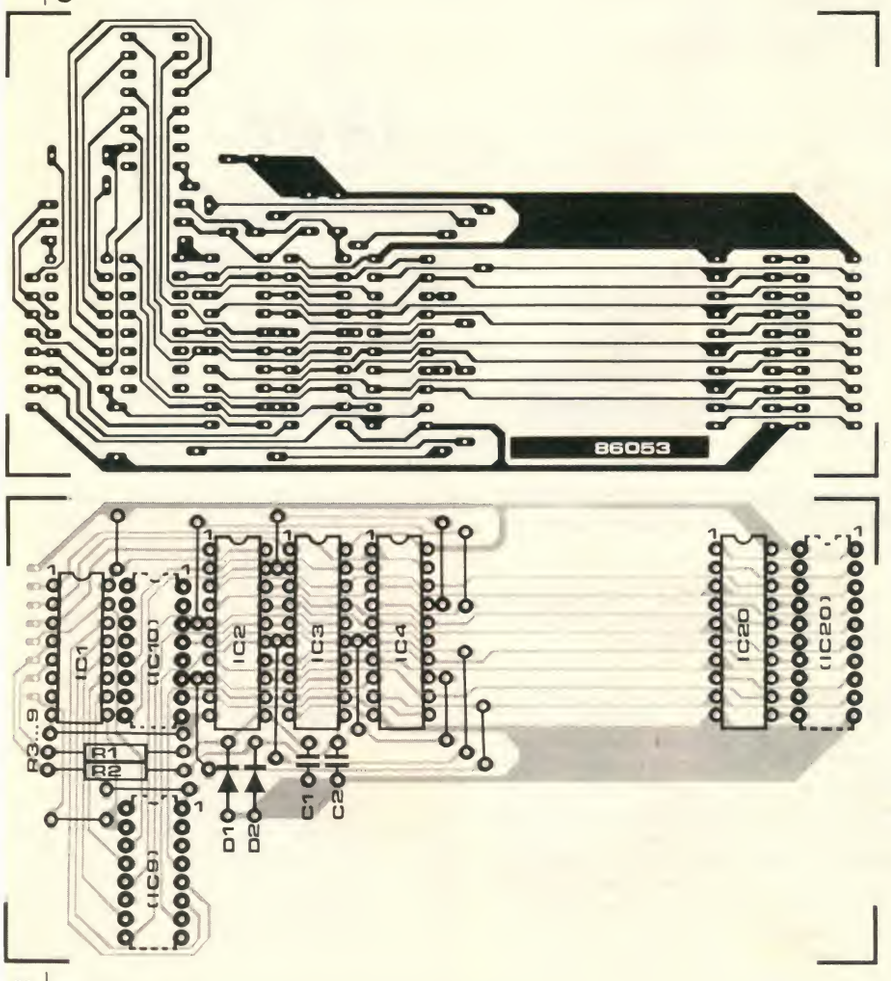


Tabelle 2.

Programmierung des PROMs

ADRESSEN	DATEN
00	1D
01	1B
02	1D
03	1B
04	8F
05	8F
06	9A
07	9A
08	97
09	97
0A	9C
0B	9C
0C	9F
0D	9F
0E	9F
0F	9F
10	FF
⋮	⋮
1F	FF

Tabelle 1. In dieser Tabelle erkennt man die "Arbeitsweise" des PROMs zur Adressendekodierung und RAS/CAS-Multiplexung.

Tabelle 2. Noch eine handliche Arbeitshilfe: So muß das PROM programmiert werden.

Bild 5. Platine und Bestückungsdruck für die Ergänzungsschaltung: Damit der Zusatzspeicher leicht realisierbar ist, hier diese handliche Arbeitshilfe.

```
                ; 64 K VIDEO BUFFER
                ;
                ; UPDATED 1/9/85 DM
                ;
                ;
C49B=           SAMGRA = GRAPH      ;START OF NEXT MODULE
C085=           SAMRED = READY      ;COMMON SUBROUTINE
                ;
                ;
                ;############################
                ;#       V I D E O          #
                ;#                          #
                ;#--------------------------#
                ;#       P.LAVIGNE          #
                ;#--------------------------#
                ;#                          #
                ;#     PART I : TEXT        #
                ;#                          #
                ;############################
                ;
E150                  * = $E150       ;GDP REGISTERS
E150=           STATUS = *
E150=           CMD    = *
E151=           CTRL1  = *+1
E152=           CTRL2  = *+2
E153=           CSIZE  = *+3
E155=           DELTAX = *+5
E157=           DELTAY = *+7
E158=           MSBX   = *+8
E159=           LSBX   = *+9
E15A=           MSBY   = *+10
E15B=           LSBY   = *+11
                ;
E164                  * = *+$14       ;HARDWARE REGISTERS
E164=           COLOR  = *
E165=           SCROLL = *+1
E167=           HSCROL = *+3
E166=           PAGE   = *+2
                ;
E170                  * = $E170
E170=           VIDBUF = *            ;VIDEO BUFFER ADDRESS REG.
                ;
CF80                  * = $CF80       ;VIDEO SCRATCH BUFFER
CF80=           ASAV   = *
CF81=           XSAV   = *+1
CF82=           YSAV   = *+2
CF83=           SAVA   = *+3
CF84=           SAVX   = *+4
CF85=           MIRXH  = *+5
CF86=           MIRXL  = *+6
CF87=           MIRYH  = *+7
CF88=           MIRYL  = *+8
CF89=           CHAR   = *+9
CF8A=           PIXBUF = *+10
CF8B=           CNTR   = *+11
CF8D=           GRCNT  = *+13
CF8E=           CURSOR = *+14
CF8F=           CRSCNT = *+15
CF90=           CRSPNT = *+16
CF91=           COLRPT = *+17
CF92=           SCRLPT = *+18
CF93=           PAGEPT = *+19
CF94=           MODE   = *+20
CF95=           TXTMOD = *+21
CF96=           VIDFLG = *+22
CF97=           BACKGR = *+23
CF9C=           INDJMP = *+28
CF9E=           DATA2  = *+30
CFA0=           DATA1  = *+32
CFA2=           DATA0  = *+34
                ;
                ;PARAMETERS FOR 'CIRCLE '
                ;
CFA4=           XL     = *+36
CFA5=           YL     = *+37
CFA6=           SL     = *+38
CFA7=           SH     = *+39
                ;
0100=           STACK  =$100
                ;
E10D=           VIAFR  = $E10D        ;VIA INTERRUPT FLAG REGISTER
F78F=           BRKOUT = $F78F        ;KEY DEPRESSED DURING OUTPUT
0819=           BREAK  = $0819        ;BASIC BREAK HANDLER
F724=           INPUT  = $F724        ;LOAD CHAR. (NO KBD STROBE)
F71D=           INPUT1 = $F71D        ;INPUT CHAR. (WITH KBD STROBE)
                ;
C000                  * =$C000
                ;
C000 4C2EC0     INITAL JMP INIT00     ;INITIALISE ALL PARAMETERS
C003 4C28C4     CHROUT JMP VIDTXT
C006 4C6EC3     CHRINP JMP CRSOR
                ;
C009 208FF7     BRKTST JSR BRKOUT     ;SAMSON SPECIFIC !
C00C C903              CMP #3         ;CTL-C ?
C00E D00F              BNE TSTRTS
C010 A911              LDA #$11       ;SET TEXT MODE
C012 205FC1            JSR COCHAR
C015 4C1908            JMP BREAK      ;GO TO YOUR BREAK HANDLER
                ;
C018 AD0DE1     TSTINP LDA VIAFR      ;KEY DEPRESSED?
C01B 2902              AND #2
C01D D001              BNE GETINP     ;GET KEY
C01F 60         TSTRTS RTS            ;ELSE CONTINUE
                ;
C020 68         GETINP PLA            ;ADJUST STACK
C021 68                PLA
C022 2024F7            JSR INPUT      ;LOAD CHAR. WITHOUT STROBE
C025 4CE0C3            JMP CONTCR     ;CONTINUE IN CURSOR ROUTINE
                ;
C028 201DF7     INPIT  JSR INPUT1     ;GET KEY WITH STROBE
C02B 20E0C3            JSR CONTCR     ;CONTINUE IN CURSOR ROUTINE
                ;
```

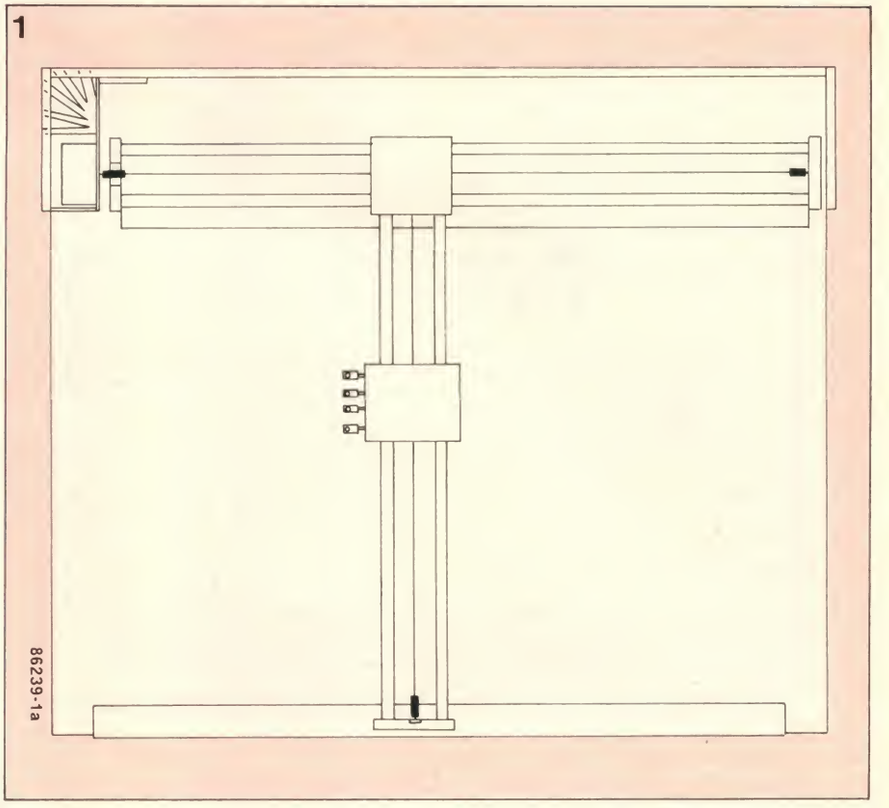
```
1080 C02E A203    INIT00 LDX #3          ;CLEAR ALL PAGES, NO RMW
1090 C030 8E66E1  CLRPG  STX PAGE        ;GET PAGE 'X'
1100 C033 8E93CF         STX PAGEPT      ;AND CLEAR THIS PAGE
1110 C036 A907           LDA #7
1120 C038 2082C0         JSR CMDRED
1130 C03B CA             DEX             ;ALL PAGES CLEARED
1140 C03C 10F2           BPL CLRPG
1150              ;
1160 C03E A900    INIT0  LDA #0
1170 C040 8D64E1         STA COLOR       ;RESET BLACK&WHITE
1180 C043 8D91CF         STA COLRPT
1190 C046 8D94CF         STA MODE        ;TEXTMODE
1200 C049 8D95CF         STA TXTMOD      ;NO GRAPH. ROUTINES IN TEXTMOD
1210 C04C 8D71E1         STA VIDBUF+1    ;RESET VIDEO BUFFER ADDRESS
1220 C04F 8D72E1         STA VIDBUF+2
1230 C052 8D70E1         STA VIDBUF      ;END MARKER IN VIDEO BUFFER
1240 C055 A203           LDX #3
1250 C057 9D85CF  NXMIR0 STA MIRXH,X     ;RESET MIRROR COORDINATES
1260 C05A CA             DEX
1270 C05B 10FA           BPL NXMIR0
1280 C05D 8E96CF         STX VIDFLG      ;NO VIDEO BUFFER
1290 C060 A900    INIT1  LDA #0          ;CLEAR SCREEN & RESET SCROLL
1300 C062 8D65E1         STA SCROLL
1310 C065 8D67E1         STA HSCROL
1320 C068 8D92CF         STA SCRLPT
1330 C06B 8D90CF         STA CRSPNT
1340 C06E A948           LDA #$48        ;BLINKRATE ABOUT 1 SECOND
1350 C070 8D8ECF         STA CURSOR
1360 C073 A90F           LDA #$F
1370 C075 8D97CF         STA BACKGR      ;BACKGROUND IS BLACK
1380 C078 20FFC0         JSR TOPLFT      ;SET CURSOR TOPLEFT
1390 C07B A90B           LDA #$B         ;CYCLIC &
1400 C07D 8D51E1         STA CTRL1       ;PEN DOWN & WRITING
1410 C080 A907           LDA #7          ;CLEAR SCREEN
1420 C082 8D50E1  CMDRED STA CMD
1430 C085 A904    READY  LDA #4          ;TEST FOR COMND EXECUTED
1440 C087 2D50E1         AND STATUS
1450 C08A F0F9           BEQ READY       ;IF NOT DONE: WAIT!
1460 C08C 60             RTS
1470              ;
4580 C316 A911    DC4    LDA #$11        ;CHR$(20) SET MINSIZE
4590 C318 8D53E1         STA CSIZE
4600 C31B 60             RTS
4610              ;
4620 C31C AD96CF  ASTX   LDA VIDFLG      ;CHR$(2):TOGGLE VIDBUF FLAG
4630 C31F 49FF           EOR #$FF
4640 C321 8D96CF         STA VIDFLG
4650 C324 60             RTS
4660              ;
4670 C325 A901    CHR4   LDA #1          ;CHR$(4)"Nn1(,n2,n3,)...
4680 C327 8D95CF         STA TXTMOD      ;DO GRAPHICS ROUTS DURING
4690 C32A 8D94CF         STA MODE        ;TEXTMODE
4700 C32D 60             RTS
4710              ;
4720 C32E FFFF    CMDTAB .WORD $FFFF,SOH,ASTX,$FFFF
4720 C330 5FC4
4720 C332 1CC3
4720 C334 FFFF
4730 C336 25C3            .WORD CHR4,$FFFF,$FFFF,$FFFF
4730 C338 FFFF
4730 C33A FFFF
4730 C33C FFFF
4740 C33E A0C1            .WORD BS,HT,LF,VT,FF,CR,$FFFF,$FFFF,HT
4740 C340 F7C1
4740 C342 15C2
5590 C428 207EC4  VIDTXT JSR SAVALL
5600 C42B 2031C4         JSR VDTXT
5610 C42E 4C88C4         JMP GETALL
5620 C431 AC96CF  VDTXT  LDY VIDFLG      ;CHECK IF VIDEO BUFFER HAS
5630 C434 D020           BNE OUTVID      ;TO BE FILLED
5640 C436 C903           CMP #3          ;CHR$(1...3) NOT STORED
5650 C438 901C           BCC OUTVID      ;IN VIDEO BUFFER
5660 C43A AC94CF         LDY MODE        ;IN GRAPHICS MODE, SPACES
5670 C43D F00C           BEQ SWP         ;ARE SKIPPED
5680 C43F C0D0           CPY #'P+$80     ;BIT 7 OF MODE IS SET
5690 C441 F008           BEQ SWP         ;SPACES KEPT ONLY WITH "P"
5700 C443 C920           CMP #'          ;SPACE ?
5710 C445 F015           BEQ REPEND
5720 C447 C90A           CMP #$A         ;SKIP LINE FEED
5730 C449 F011           BEQ REPEND
5740 C44B A000    SWP    LDY #0          ;CHAR. STORED IN VIDEO
5750 C44D 8D70E1         STA VIDBUF      ;BUFFER
5760 C450 2092C4         JSR INCBUF      ;CONTENT OF NEXT LOCATION
5770 C453 8C70E1         STY VIDBUF      ;IS END MARKER
5780 C456 2088C4  OUTVID JSR GETALL      ;RESTORE REGISTERS
5790 C459 205FC1  STOPIT JSR COCHAR      ;EXECUTE AND
5800 C45C 4C09C0  REPEND JMP BRKTST      ;TEST FOR BREAK
5810              ;
5820 C45F AD96CF  SOH    LDA VIDFLG      ;CHR$(1): REPEAT SCREEN
5830 C462 F0F8           BEQ REPEND      ;NO REPEAT IF BUF NOT CLOSED
5840 C464 A900           LDA #0          ;RESET VIDEO BUFFER POINTER
5850 C466 8D71E1         STA VIDBUF+1
5860 C469 8D72E1         STA VIDBUF+2
5870 C46C AD70E1  REPEAT LDA VIDBUF
5880 C46F F0EB           BEQ REPEND      ;STOP IF END MARKER FOUND
5890 C471 8D80CF         STA ASAV        ;SAVE CHARACTER AND
5900 C474 2059C4         JSR STOPIT      ;PRINT IT
5910 C477 2092C4         JSR INCBUF
5920 C47A D0F0           BNE REPEAT
5930 C47C F0DE           BEQ REPEND
5940              ;
5950 C47E 8D80CF  SAVALL STA ASAV
5960 C481 8C82CF         STY YSAV
5970 C484 8E81CF         STX XSAV
5980 C487 60             RTS
5990              ;
6000 C488 AD80CF  GETALL LDA ASAV
6010 C48B AC82CF         LDY YSAV
6020 C48E AE81CF         LDX XSAV
6030 C491 60             RTS
6040              ;
6050 C492 EE71E1  INCBUF INC VIDBUF+1
6060 C495 D003           BNE ENDIT
6070 C497 EE72E1         INC VIDBUF+2
6080 C49A 60      ENDIT  RTS
6090              ;
6100 C49B=        GRAPH  = *
```

Selbstbau-Plotter

Teil 1: Die Mechanik

Franz Blaha

Ein Selbstbauplotter, dessen Mechanik unter 200,- DM kosten soll? Und der in vier Farben auf DIN A3 plottet - geht das denn? Es ist kaum zu glauben und doch wahr. Allerdings hat die Sache einen Haken, den wir nicht verschweigen wollen: Man muß selbst sehr viel "Hand anlegen" - würde man die Mechanik als Einzelstück von einem Betrieb anfertigen lassen, müßte man aufgrund der Lohnkosten tief in die Tasche greifen. Also nichts für die professionellen Computer-Anwender unter den ec-Lesern, dafür um so mehr für die Hobbyisten, die nicht nur über genug Zeit, sondern auch über einiges mechanische Geschick verfügen. Und möglichst auch noch über einige sehr präzise Werkzeuge...



Die Grundidee zu diesem System war, einen 4-Farb-Plotter zu entwickeln, der in erster Linie durch seine Zeichenqualität überzeugt, mechanisch robust und vor allem vom Kostenaufwand her sehr, sehr preiswert ist. Daß dabei die Plotgeschwindigkeit nicht allzu sehr ins Hintertreffen geraten durfte, mußte ebenfalls berücksichtigt werden. Bestimmte Kompromisse waren dabei unvermeidlich. Der entscheidende dürfte der Zeitaufwand sein, den man aufbringen muß, soll dieses Projekt ein Erfolg werden. Denn gerade bei der Mechanik kann jedes überhastete Arbeiten den Erfolg des Ganzen in Frage stellen. Wer sowieso über die berühmten zwei linken Hände verfügt und sich auch nicht der Hilfe eines Bekannten oder Verwandten mit ausreichendem mechanischem Geschick vergewissern kann, sollte sich den Nachbauversuch gründlich überlegen. Die positive Kehrseite der Medaille: Preiswerter als auf diesem Wege wird man einen Din-A3-Vierfarb-Plotter sicher nicht bekommen.

Vorbereitungen

Als erstes muß man sich die notwendigen Bauteile besorgen - doch diesmal nicht im Elektronik-Fachhandel (zumindest nicht in dieser ersten Baustufe), sondern im Bau- oder Heimwerkermarkt. Einige Teile kann man vielleicht auch dort gleich passend zurechtschneiden lassen, daß kann einige Zeit ersparen und ist meist sehr preiswert. Zunächst sind da die Holzteile: Man benötigt eine Grundplatte von 650 x 550 mm, hierfür ist eine Preßspanplatte mit 19 mm Dicke am besten geeignet. Bei den Seitenteilen reicht Preßspan mit 8 mm Dicke, man benötigt eines in der Größe von 650 x 120 mm und zwei von je 120 x 120 mm.

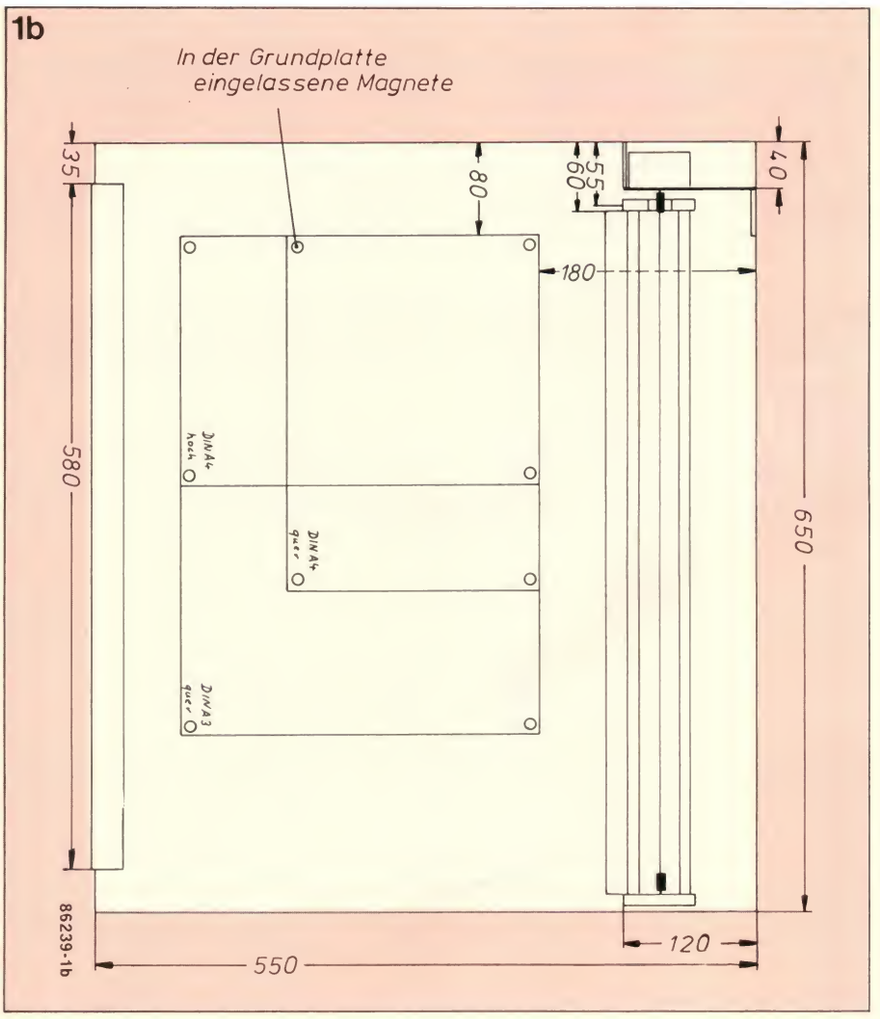
Als nächstes kommen Aluminium-L-Schienen mit einer Materialstärke von 1,5 mm an die Reihe, und zwar eine mit den Maßen 150 x 120 x 30 mm und zwei von je 1000 x 15 x 20 mm. Außerdem benötigt man noch etwas Aluminiumblech mit 3 mm Dicke.

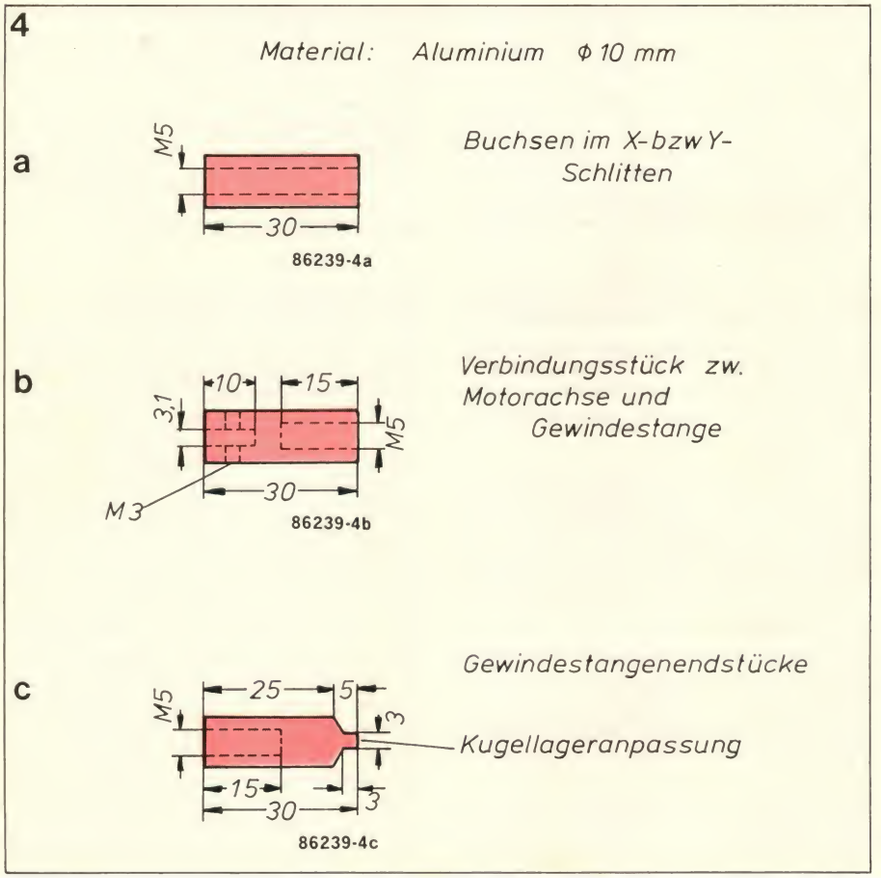
Jetzt wird es rund: Auf der Einkaufsliste stehen vier Aluminium-Rohre mit einem Außendurchmesser von 10 mm und 1000 mm Länge sowie einige Stückchen Kupferrohr mit einem Innendurchmesser von 10 mm und einer Dicke von 1 mm. Von letzterem benötigt man nicht allzu viel, am besten ist es, wenn man einige Reste "abstauben" kann. Das gilt auch für einige Stückchen Plexiglas oder Pertinax mit 5 und 10 mm Dicke: Auch hier benötigt man nicht allzu viel.

Dann stehen noch sechs Kugellager auf der Einkaufsliste (Innendurchmesser 3mm), dazu kommen noch 9 kleine, am besten runde Magneten, und als letztes kommen für den Antrieb zwei Gewindestangen M5 x 0.8. Da man die nicht in "Zwischenlängen" bekommen wird, muß man sich jeweils einen ganzen Meter kaufen... Und damit ist der Besuch im Bau- oder Heimwerkermarkt beendet.

Herstellung der Einzelteile

Nun beginnt der eigentlich schwierige Teil. Um ihn zu erleichtern, finden Sie nicht nur in diesem Artikel einige Bauzeichnungen, sondern auch genau in der Heftmitte ein Doppelblatt zum Herausnehmen: Dort sind einige besonders wichtige Teile im exakten Maßstab 1 : 1 als Bohrschablone abgebildet. Nach dem Herausnehmen der Seiten aus dem Heft sollten Sie zunächst diese beiden Seiten kopieren, bevor Sie sie in die einzelnen Bohrschablonen "zerschnippeln": Dort befinden sich nämlich auch einige Angaben über die Durchmesser von Bohrungen! Benutzen Sie aber nicht die Kopien als Bohrschablonen: Viele Fotokopierer verkleinern die Vorlagen unmerklich für ein ungeübtes Auge - und das kann "ins Auge gehen". Lesen Sie außerdem erst den Artikel genau durch: Manche Teile werden zweimal benötigt. Bei Verwendung der Abbildungen als Bohrschablonen sollte man also vorsichtig mit einem Körner arbeiten und nicht einfach nach





dem Motto "Aufkleben, Durchbohren"!

Nun zu den einzelnen Teilen:

Bild 2a und 2b zeigen die Alurohrhalter für die X-System-Gleitrohre, sie werden jeweils als einem Kunstoffstück mit den Außenmaßen 70 x 40 mm und einer Dicke von 10 mm angefertigt. Bild 3a zeigt die Aluminium-Trägerplatte für den X-Motorträger, Bild 3b entsprechend für den Y-Motorträger: Beide werden aus 3 mm dickem Alu-Blech gefertigt. Diese Bilder finden Sie im Baubogen in der Heftmitte.

Bild 4a zeigt die Gewindebuchsen im X- bzw. Y-Schlitten, Bild 4b die Verbindungsstücke zwischen Motorachse und Gewindestange und Bild 4c die Gewindestangen-Endstücke. Von den Buchsen benötigt man vier Stück, von den beiden anderen Teilen je 2. Der Außendurchmesser beträgt jeweils 10 mm. Dies sind die schwierigsten von allen vorkommenden Teilen! Und auf die Gewindestangenendstücke müssen noch die Kugellager eingepaßt werden... Werden diese Teile nicht sauber gefertigt, kann man sich den Rest der Mühe sparen.

Bild 5 zeigt die Teile des X-Schlittens: Bild 5a die Seitenteile (es müssen zwei Stück angefertigt werden), Bild 5b das Vorderteil und Bild 5c das Rückteil. Letzteres ist gleichzeitig das Endstück des Y-Armes. Auch diese Teile werden aus 10 mm dickem Kunststoff angefertigt. Die drei Bilder befinden sich ebenfalls im Baubogen. Bild 6 zeigt eine Draufsicht des X-Schlittens, hierauf kommen wir bei der Montage zurück.

Die in Bild 7 und Bild 8 gezeigten Teile des Y-Schlittens werden aus 5 mm dickem Kunststoff gefertigt, um Gewicht zu sparen. Diese Teile gehören zum weitaus kompliziertesten Teil der Gesamtkonstruktion und bedürfen deshalb besonderer Aufmerksamkeit. Die Bilder finden Sie wieder im Baubogen (Bild 7a, 7b, 7c, 7d, 8a und 8b).Das Seitenteil (Bild 8a) muß wiederum zweimal aufgebaut werden.

Als nächstes sind aus Alu-L-Trägerschienenstücken die Teile gemäß Bild 10a und 10b anzufertigen: Bei Bild 10a handelt es sich um eine Trägerschiene für zwei Kugellager zur Entlastung X-Achse, bei Bild 10b um einen Haltewinkel zur Eliminierung des X-Motorsspiels.

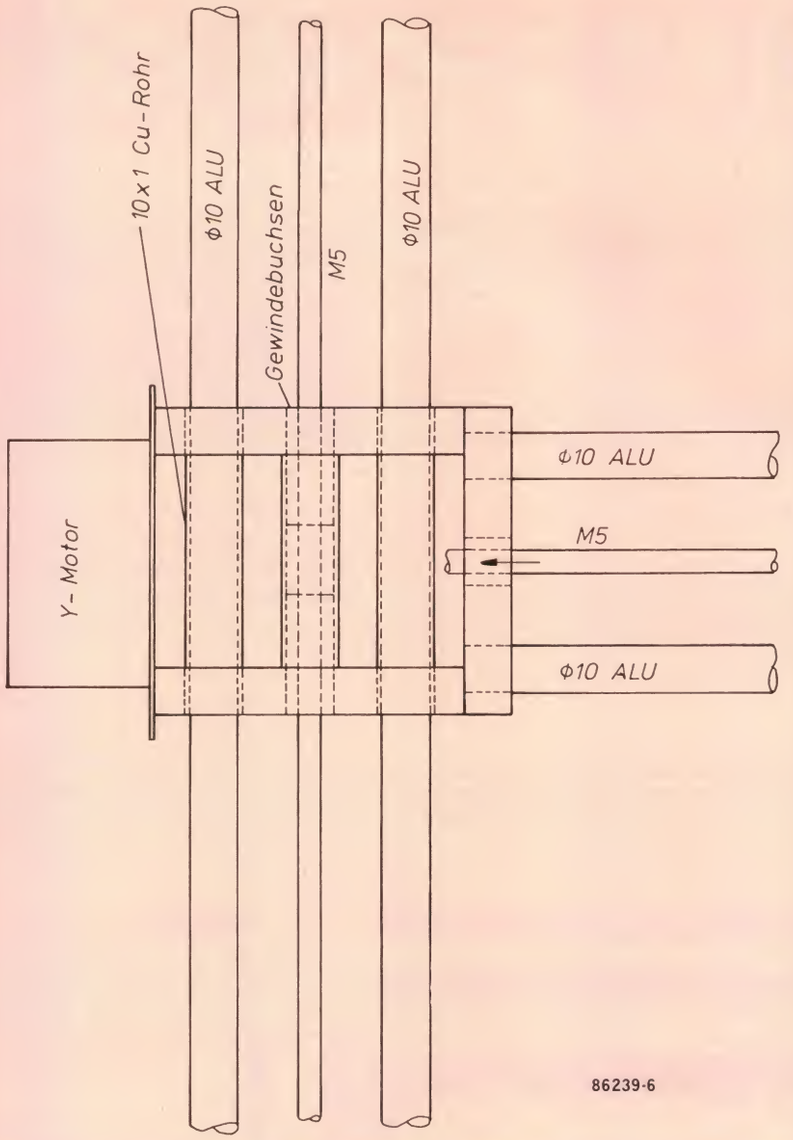
Als letztes ist dann noch die Halteplatte zur Parallel-Kugellagerung des Y-Armes an der unteren Alu-L-Schiene anzufertigen, das entsprechende Bild 10c befindet sich wieder im Baubogen.

Montage

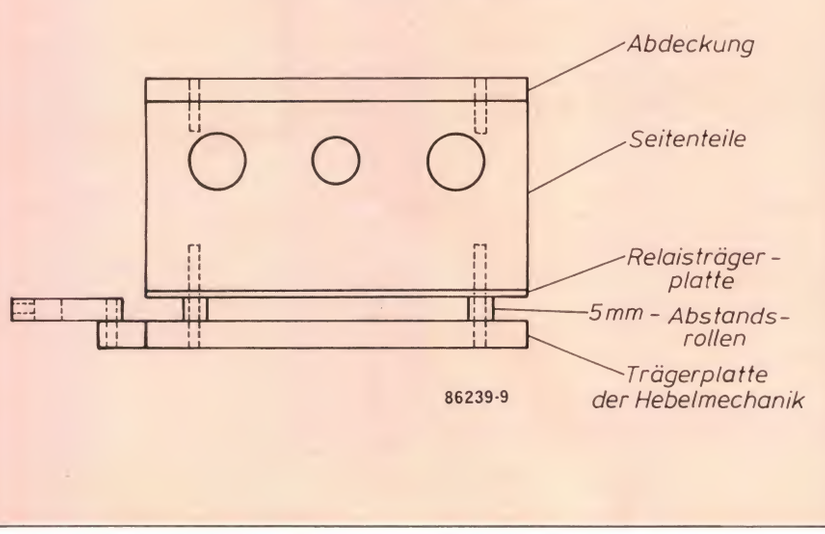
Wenn Sie bisher sauber gearbeitet haben, werden Sie hier kaum auf Probleme stoßen, wenn nicht... Wenn mal etwas nicht passen will: Es ist besser, ein Teil gleich nochmal sauber anzufertigen, als hinterher den ganzen Plotter wieder zu zerlegen!

Bild la zeigt eine Gesamtdraufsicht des

6



9



Plotters, in Bild 1b fehlt zur besseren Übersicht der Y-Schlitten. Zunächst werden in die Grundplatte die Magneten so wie in Bild 1b zu sehen eingelassen. Sie dienen später zusammen mit Metallstreifen oder metallhaltigen Kunstgummi-Streifen, wie sie sich in Zeichenplatten finden, zum Festhalten des Papiers während des Plottens.

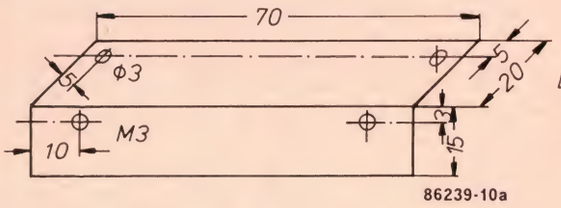
Dann beginnt die eigentliche Montage der Plotter-Mechanik. Als erstes wird die X-Motorträgerplatte (Bild 3a) mit Hilfe von zwei Winkeln oder Pertinaxresten im rechten Winkel zur Oberkante der Grundplatte im linken oberen Eck montiert. Dann erfolgt die Montage der Alurohr-Halter: Bild 2a links, Bild 2b rechts, wobei der rechte zur Aufnahme der Gewindestangen-Buchse mit einem Kugellager zu versehen ist. Um das Längsspiel des Motors auszugleichen, bringt man noch den Alu-Winkel (Bild 10b) am rechten Ende der Gewindestange an.

Als nächstes nimmt man Bild 6 zur Hand, den nun wird der X-Schlitten zum Einbau hergerichtet. Dazu werden die Teile nach Bild 5a und 5b miteinander verschraubt und anschließend die Y-Motorträgerplatte montiert. Danach werden die Kupferrohre für die Gleitführung eingetrieben und fixiert. Hierzu genügen einige Tropfen Sekundenkleber. Dann kommen die Gewindebuchsen an die Reihe: Am besten werden sie mit einem zusätzlichen Stück Kupferrohr im Innern des X-Schlittens ausgerichtet, damit sie nicht verkanten. Dadurch, daß hier zwei Gewindebuchsen verwendet werden, wird einem Flankenspiel des Gewindes vorgebeugt. Nun wird der Alu-Winkel mit den Kugellagern nach Bild 10a zur Entlastung der X-Achse montiert. Die Kugellager müssen dann beide leichtgängig auf der oberen Alu-L-Schiene zu liegen kommen.

Jetzt wird der X-Schlitten montiert. Dazu werden von der rechten Seite aus die Alu-Rohre zunächst durch die rechte Halterung, dann durch den Schlitten und zuletzt in die linke Halterung gesteckt. Dann wird von der linken Seite aus die Gewindestange "durchgedreht" (Motor noch nicht montiert). Teil 4a darf erst auf die Gewindestange aufgedreht werden, nachdem diese die Gewindebuchsen im Schlitten passiert hat. Bei dieser Montage stellt sich heraus, ob die Montage im Schlitten und in den Halterungen in einer Flucht sind. Minimale Abweichungen können durch Ausfeilen der Halterungen ausgeglichen werden. Bei zu großer Abweichung sollte man das entsprechende Teil neu fertigen. Der Motor wird erst dann mit dem Verbindungsstück nach Bild 4b angeflanscht, wenn alles andere "paßt".

Im nächsten ec-Heft geht es weiter mit der Montage des schwierigsten Einzelstückes des Plotters: Dem Y-Schlitten mit der Stiftmechanik. Daran schließt sich dann die Beschreibung der Elektronik inklusive der Schrittmotorsteuerung an.

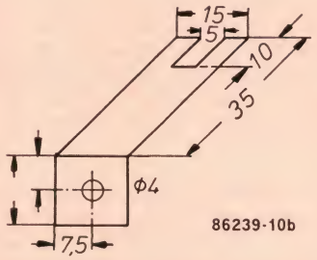
10a



L- Trägerschiene mit zwei Kugellagern zur Entlastung der X-Achse

86239-10a

b



Alu-Haltewinkel zur Eliminierung des X-Motorspiels

86239-10b

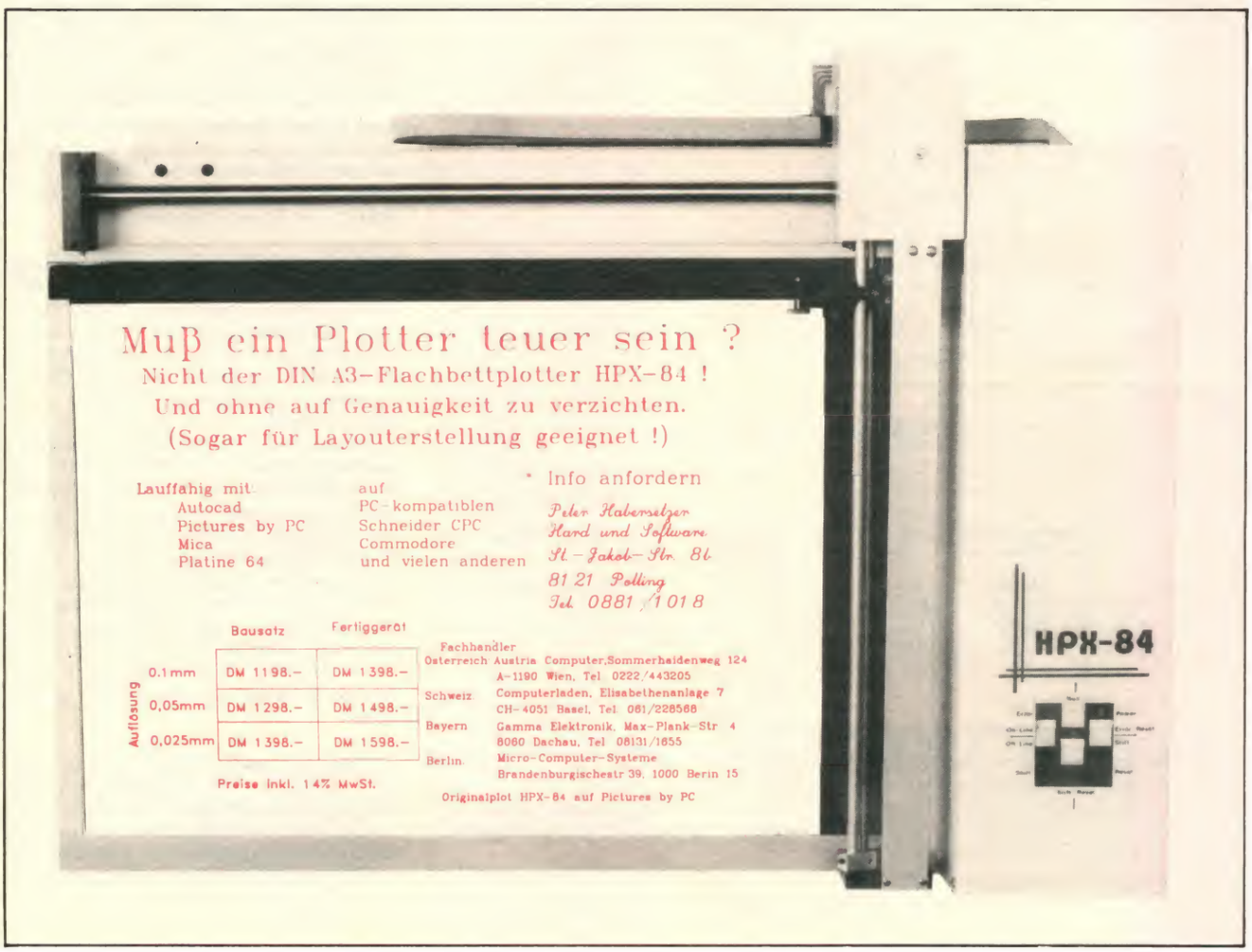
Anm. der Redaktion:

Wir sind bei diesem Projekt auf die Reaktionen unserer Leser gespannt: Dürfen wir Ihnen soviel Mechanik zumuten, oder sollen wir uns lieber auf die Beschreibung, den Praxistest von Fertigbausätzen beschränken? Wir meinen damit nicht nur Plotter: Das Gebiet der Robotik ist eng mit der Computertechnik verwandt. Also schreiben Sie uns ruhig auch dann Ihre Meinung zu dieser Frage, wenn Sie diesen Artikel nur interessehalber gelesen haben. Ein anderer Punkt: Reicht diese Beschreibung oder ist sie zu knapp oder gar schon zu ausführlich? Wir können recht gut einschätzen, wie lang ein Artikel zum Beispiel zu einem Betriebssystem sein muß - doch in diesem speziellen Fall sind wir etwas unsicher. Auch dazu interessiert uns Ihre Meinung!

Zum Schluß: Üblicherweise finden Sie neben jedem Bild eine Bildunterschrift zur schnellen Information, worum es bei diesem Bild geht. Die haben wir angesichts der vielen Detail-Bilder hier aus Platzgründen mal weggelassen, zumal ein ganzer Teil der Bilder sich sowieso im "Baubogen" befindet.

(PL)

Anzeige



SOFTWARE

FORTH — komfortabel

Eine Fülle von Systemerweiterungen

Dieter Reithinger

Ein recht bescheidenes FORTH für den EC-65 ist schon seit längerer Zeit als Free-Soft erhältlich. Seitens der Redaktion war es als kleine Beigabe zum EC-65 gedacht. Den Autor dieses Artikels — der seit mehreren Jahren mit FORTH arbeitet — hat dies gestört: Er hat nicht nur die ursprüngliche Version optimal an den EC-65 angepaßt, sondern wesentliche Verbesserungen und Erweiterungen vorgenommen. Das Ergebnis kann sich sehen lassen, wie jeder an FORTH Interessierte bereits beim Lesen dieses Artikels feststellen wird. Noch mehr allerdings merkt man dies, wenn man mit dem neuen FORTH arbeitet. Es ist wieder als Free-Soft erhältlich.

Der Grundwortschatz eines fig-FORTH-Systems umfaßt nur etwa 200 Worte. Auf der FORTH-Diskette für den EC-65 findet man zusätzlich einen Zeilen-Editor und ca. 30 weitere Hilfsworte vor. Durch diese etwas spartanische Ausstattung wird FORTH seinem Anspruch, auch Betriebssystem zu sein, nicht besonders gerecht. Normalerweise lädt man daher entsprechende als Source-Code vorliegende Hilfsprogramme bei Bedarf nach. Das Spart eventuell Speicherplatz, kostet aber viel Zeit, da diese Programme jedesmal neu compiliert werden müssen. Ein FORTH-System benötigt im allgemeinen nur 8 bis 10 KByte RAM. Da der EC-65 genügend Speicherplatz zur Verfügung stellt, bietet es sich an, derartige Hilfsprogramme compiliert im RAM bereitzuhalten. Hierzu bestehen zwei Möglichkeiten:
1. Diese Erweiterungen werden als Bestandteil des FORTH-Kernals beim Booten gleich mitgeladen — das ist schnell, bedarf jedoch einiger Änderungen im DOS, in dem FORTH-Wort, das die Diskette verwaltet, und im grundsätzlichen Aufbau der FORTH-Diskette. Der Benutzer hat dann allerdings keine Einflußmöglichkeit darauf, welche Programme geladen werden.
2. Nach dem Booten compiliert man sofort alle in Frage kommenden Hilfsprogramme. Das kann bis zu 20 Minuten dauern, funktioniert ohne Änderungen auf der Diskette und gibt dem Programmierer die volle Entscheidungsfreiheit, was geladen wird.

Beide Verfahren weisen gravierende Nachteile auf. Bei der hier vorgestellten Erweiterungssoftware wurde deshalb ein Kompromiß entwickelt, der die Vorteile beider Möglichkeiten verbindet.

Dieser Artikel erläutert zunächst grundsätzlich das FORTH-Disketten-Handling auf dem EC-65 und beschreibt anschließlich die Möglichkeiten, die das umfangreiche, über die Software-Verteiler erhältliche Erweiterungspaket bietet. Dieses Paket enthält zum Beispiel Assembler, String-Package, HEX-Editor, Verarbeitung von Zahlen doppelter Genauigkeit, Decompiler, Disassembler (auch direkt von Diskette), Spiele, Grafik, Systemkorrekturen und als Krönung einen Full-Screen-Editor, der keine Bedienerwünsche mehr offen läßt (mehr als 40 Kommandos werden "verstanden").

Grundsätzliches

FORTH arbeitet nach seinem Start grundsätzlich als Interpreter: Alle über Tastatur eingegebenen Kommandos (bei FORTH Worte genannt) werden unmittelbar ausgeführt — für BASIC-Programmierer nichts Neues. Um ein BASIC-Programm zu schreiben, gibt man die auszuführenden Befehle mit Zeilennummern ein. Diese stehen dann — eventuell in TOKENS umgewandelt — im RAM. In FORTH wird ein "Wort" definiert. Jede Wortdefinition beginnt mit einem ":" (daher der Begriff Colon-Definition), der das System in den Compile-Modus umschaltet. Nach dem Namen, den das Wort bekommen soll, werden auch hier die auszuführenden Befehle eingegeben, die jedoch FORTH sofort compiliert im RAM ablegt. Compilieren bedeutet vereinfacht ausgedrückt, daß die Adresse, an der der betreffende Befehl im RAM steht, ermittelt und in das neue Wort eingetragen wird. Den Abschluß bildet ein ";", das unter ande-

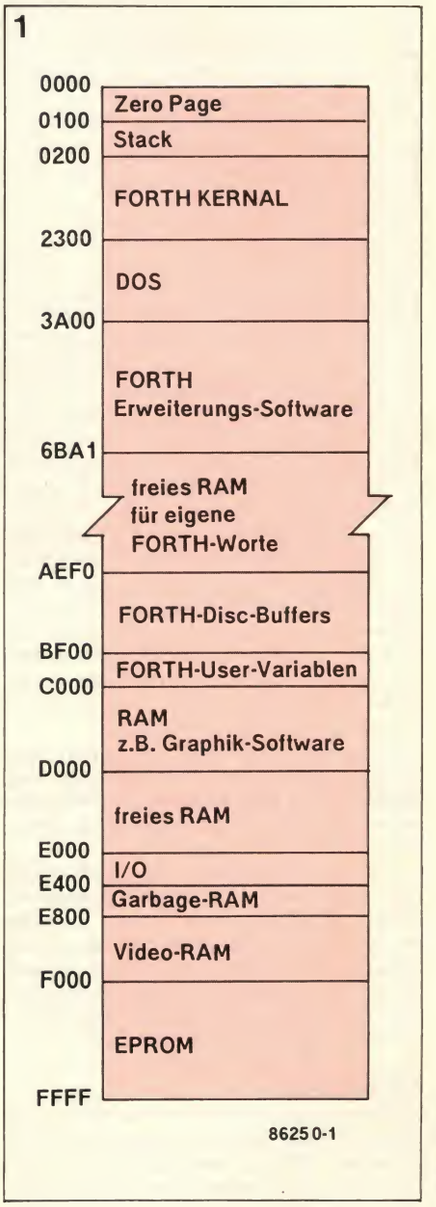


Bild 1. Hier erkennt man, wie der Speicher des EC-65 bei Verwendung der neuen FORTH-Diskette aufgeteilt ist.

rem wieder in den Interpreter-Modus zurückschaltet. Das jetzt definierte Wort ist Bestandteil des FORTH-Kerns und läßt sich in folgenden Wort-Definitionen mitverwenden. Beim Ausschalten des Computers sind alle neuen Worte verloren.

DISC=RAM

Hier kommt die Diskette als derzeit noch bester Kompromiß zwischen Preis, Geschwindigkeit und Fassungsvermögen ins Spiel. FORTH betrachtet die Diskette als einfache Fortführung des RAMs und erledigt alle nötigen Schreib- und Lese-Operationen vollautomatisch. Beim EC-65 ist die Diskette in Blöcke zu je 1024 Bytes aufgeteilt (= 2 Sektoren mit je 4 Pages pro Track). Im RAM werden vier Pufferspeicher gehalten, die je einen Block aufnehmen können. Jeder dieser Blöcke entspricht einem FORTH-Screen mit 16 Zeilen zu 64 Zeichen, die von 1 aufsteigend durchnumeriert sind. Die Zuordnung Screen-Nummer zu Track/Sektor auf der Diskette erledigt das System. In diesen Screens kann der Programmierer seine Source-Codes, aber auch interpretativ abzuarbeitende Befehlsfolgen mittels eines Editors ablegen. Mit Eingabe der Screen-Nummer und dem Wort LOAD <cr> wird der Inhalt dieses Screens eingelesen und wie eine Eingabe von der Tastatur abgearbeitet. Am Ende des Screens muß entweder das Wort "-->" (= nächsten Screen laden) oder ";S" (=zurückschalten auf Tastatureingabe) stehen.

Mit dem Wort **UPDATE** wird der zuletzt bearbeitete Screen als geändert gekennzeichnet. **FLUSH** schreibt alle so gekennzeichneten Screens auf die Diskette zurück. Sowohl der beim EC-65 schon vorhandene Zeileneditor als auch der neue Full-Screen-Editor kennzeichnen selbst die jeweils bearbeiteten Screens mit UPDATE. Die vier Pufferspeicher zur Aufnahme der Screens werden zyklisch belegt: Jeweils der Puffer mit dem ältesten Inhalt wird beim Einlesen eines neuen Screens überschrieben. War der ursprüngliche Pufferinhalt mit UPDATE gekennzeichnet, wird dieser vorher auf Diskette gesichert. Der Programmierer braucht sich daher normalerweise um das Diskettenhandling nicht zu kümmern, jedoch gibt es zwei Ausnahmen:

1. Bei jedem Kalt- oder Warmstart von FORTH sollten die Pufferspeicher mit **EMPTY-BUFFERS** gelöscht beziehungsweise in einen definierten Anfangszustand gebracht werden.
2. Vor dem Ausschalten des Computers rettet man mit FLUSH eventuell noch im Puffer vorhandene modifizierte Screens auf die Diskette. Da dieser Artikel keine Einführung in FORTH bieten kann, sei auf (1) und (2) verwiesen. Die Speicheraufteilung des EC-65 unter FORTH zeigt Bild 1, die Belegung der Zero-Page Bild 2.

Zusätzliches

Im Kernal des FORTH für den EC-65 sind weitere Worte für das Diskettenhandling enthalten. **DOS** mit nachfolgender Eingabe eines der gewohnten DOS-Befehle (und natürlich <cr>) führt diesen aus. Dieses Wort kann aber nur im Interpreter-Modus, nicht in einer Wort-Definition (!) verwendet werden. Außerdem besteht

```
2          ZERO - PAGE

Belegung durch FORTH

Hex

20      Bottom of )
                  )   DATA - Stack
9E      Top of    )

A6      scratch workspace

AE      Interpretive-Pointer

B1      Code-Field-Pointer

B3      User-Area-Pointer

B5      temporary for X-Register

                          86250-2
```

Bild 2. Die Belegung der Zero-Page durch FORTH.

die Möglichkeit, Screens mit Namen zu belegen. Dazu wird in den Screens 6, 7 und 8 ein Directory geführt, das zunächst initialisiert werden muß (**DIRINIT** in Screen 10 der alten FORTH-Diskette). Neue Namen trägt man mit

(Screen-Nummer) ENTRY (NAME) <cr>

in das Directory ein. Mit

(Name) DELETE <cr>

kann man Directory-Eintragungen löschen und mit

RENAME (alter Name) (neuer Name) <cr>

vorhandene Namen ändern.

NLOAD (Name) <cr>

lädt (compiliert) den entsprechenden Screen.

EDIT (Name)

stellt ihn mit dem Zeilen-Editor zur Verfügung.

DIR

druckt das Directory aus.

Übrigens, falls Sie bei der alten Diskette schon versucht haben, mit NLOAD den Screen **SEDIT** zu laden, werden Sie nur eine Fehlermeldung erhalten haben. Der Fehler liegt in Screen 59, Zeile 8: Hier muß am Anfang der Zeile statt 8 10 nur 60 62 eingetragen werden, dann funktioniert das Laden.

Neue Software

Wollten Sie bisher in einem selbst definierten Wort DOS-Kommandos verwenden (etwa: Umschalten auf zweites Laufwerk), blieb Ihnen nur der kompliziert zu programmierende Sprung von FORTH in die entsprechende Maschinensprache-Routine im DOS. Hier bietet die neue FORTH-Erweiterungs-Software gleich zwei erleichterte Möglichkeiten an. Wir wollen dieses umfangreiche Software-Paket (immerhin über 12 K compilierter FORTH-Code) jedoch der Reihe nach betrachten. Diese Programme wurden mit 80-Track-Laufwerken entwickelt. Um sie auch Benutzern von 40-Track-Laufwerken zur Verfügung stellen zu können, mußten die Programme auf zwei Disketten im 40-Track-Format verteilt werden:

— Disk 19 enthält die gesamte Software auf 80 Tracks.
— Disk 20 ist die neue System-Diskette mit 40 Tracks.
— Disk 21 beinhaltet die übrigen Source-Screens auf 40 Tracks.

Übrigens, ein Inhaltsverzeichnis der Diskette auf dem Drucker erhält man durch Eingabe von

PON 1 145 INDEX POFF

Besitzer von 40-Track-Laufwerken ersetzen 145 durch 65. Will man die Ausgabe nur am Bildschirm, müssen PON und POFF weggelassen werden.

BOOT

Die BOOT-Routine der FORTH-Diskette lädt zunächst Track 0 und 1 mit dem OHIO-DOS nach $2200 — $31FF und danach Track 2 bis 5 und Track 6, Sector 1 mit dem FORTH-Kernal nach $0200 — $22FF. Ab Track 7 befinden sich bereits die FORTH-Screens auf der Diskette. Bisher meldete sich dann das System und wartete auf Eingaben von der Tastatur. Jetzt lädt es nach einer System-Meldung auf dem Bildschirm automatisch Screen 9 von der Diskette. In diesem Screen (auf Disk 19 und 20) wird zunächst das Input-Distributor-Flag im DOS auf Tastatureingabe gesetzt (vom 10 auf 01) und Platz für die Erweiterungssoftware geschaffen. Es folgt die Definition des Wortes, das die bereits compiliert in Screens abgelegten Hilfsprogramme in den Speicher lädt. Dieses Wort (**LOAD-EXT**) wird unter Angabe der Zieladresse ($3A00), des ersten Screens (15) und der Anzahl der zu ladenden Screens (13) sofort aufgerufen. Schließlich müssen die Erweiterungen noch in das System eingebunden und zwei Korrekturen (**NEW-EMIT** und **NLST**) durchgeführt werden. Die Korrekturen im alten FORTH sind weiter unten beschrieben. Nach einer endgültigen Meldung am Bildschirm stehen damit eine große Anzahl neuer Worte und Dienstprogramme zur Verfügung. Eine detaillierte Beschreibung aller Möglichkeiten, die sich dem Benutzer hier bieten, ist aus Platzmangel gar nicht möglich, außerdem sind einem phantasievollen Programmierer in FORTH auch (fast) keine Grenzen gesetzt. Wir müssen uns daher auf einige wenige besonders erklärungsbedürftige Dinge beschränken. Darauf folgt eine kurze Beschreibung (Bedienungsanleitung) der enthaltenen Dienstprogramme und eine tabellarische Auflistung wichtiger Worte. Am Ende wird auf die Software eingegangen, die als Source-Code in Screens vorliegt, und bei Bedarf/Interesse dazugeladen werden kann (zum Beispiel Grafik für Kolorator).

Hilfsworte

Eines der ersten Worte, die ein "FORTH-Anfänger" gewöhnlich mit Erfolg ausprobiert, ist VLIST. Beim EC-65 erschienen jedoch Grafik-Zeichen anstelle des jeweils letzten Buchstabens eines Wortes, da dieses FORTH das höchstwertige Bit bei der Zeichenausgabe nicht zurücksetzt. Zur Korrektur wurde NEW-EMIT

definiert, das schon in Screen 9 (beim Booten) aufgerufen wird. Die ursprüngliche Zeichenausgabe kann mit **OLD-EMIT** wiederhergestellt werden. Das ist zum Beispiel dann (vorher) nötig, wenn mit COLD das System zurückgesetzt und damit die Erweiterungssoftware (also auch die neue Zeichenausgabe) gelöscht werden soll. **SYSUPDATE** schützt alle bei Aufruf vorhandenen Worte vor dem "vergessen" (löschen), auch durch COLD. Das Gegenstück dazu bildet **SYSDOWN**, das geschützte Worte, auch jene im ursprünglichen FORTH-Kernal, löscht. Da der compilierte FORTH-Code aus effektiven Adressen besteht, ist er nicht im Speicher verschiebbar. Daher kann immer nur vom "jüngsten" Wort ausgehend in Richtung "älterer" Worte durchgehend gelöscht werden. Löschen eines "alten" Wortes mit Nachrücken der "jüngeren" Worte ist nicht möglich. Ein Beispiel: Sie definieren (in dieser Reihenfolge) die Worte TEST BEISPIEL VERSUCH (zum Beispiel durch : TEST ; usw.). Nun möchten Sie BEISPIEL wieder löschen. Das Standardwort hierzu heißt **FORGET**. FORGET BEISPIEL führt diese Lösung durch. Das "jüngere" Wort VERSUCH wird dabei mitgelöscht. Hätten Sie nach der Definition der obigen drei Worte Ihr System mit SYSUPDATE geschützt, wäre der (versehentliche) Aufruf von FORGET erfolglos. Dann kommt SYSDOWN zu Zuge. Nach Eingabe von ' (Hochkomma, sprich Tick), Name des jüngsten, nicht mehr zu löschenden Wortes (hier TEST), '(Tick), Name des ältesten, noch zu löschenden Wortes (hier BEISPIEL) und SYSDOWN werden die gewünschten Worte "vergessen".

Verschiebungen von Speicherbereichen führt **CMOVE** aus (Source-Adresse, Ziel-Adresse, Länge -->). Überlappen sich die Bereiche (Beispiel: HEX E800 E850 100 CMOVE), wird der Source-Bereich überschrieben, bevor der alte Inhalt in den Ziel-Bereich gebracht werden konnte (CMOVE überträgt ab Anfang der Bereiche). Überlappen ist nur zulässig, wenn der Ziel-Bereich an niedrigerer Adresse als der Source-Bereich liegt. Für den umgekehrten Fall wurde **-CMOVE** (mit gleichartiger Parameterangabe) definiert. **CASES** bietet die Möglichkeit, Sprungtabellen anzulegen (3). Die Eingabe:

(Anzahl der Tabellenplätze) CASES (Name des Arrays) (Initialisierungs-Wort)

führt zum Aufbau eines Arrays, das unter dem angegebenen Namen angesprochen werden kann. Durch Angabe einer Fall-Nummer und dem Aufruf des Array-Namens wird die an der betreffenden Stelle im Array stehende Prozedur (hier steht natürlich nur deren Adresse) ausgeführt. Bei der Definition eines derartigen Arrays wird zunächst an jeder Stelle die Adresse des angegebenen Initialisierungswortes (von (3) abweichend) eingetragen. Das Wort CASE dient zum gezielten Auffühlen des Arrays:

CASE (Nummer des Falles) (Name des Arrays) (Name der auszuführenden Prozedur)

Einige der beschriebenen Worte sind schwer zu verstehen — und auch schwierig zu erklären. Hier gilt jedoch: "Probieren geht über studieren". Beispiele für die Anwendung von CASES/CASE finden sich auf Disk 19 in den Screens 67, 92, 116 bis 118 und auf Disk 21 in den Screens 17, 37, 61 bis 63.

3

```
--KOMMANDOZEILE--

LI ...              LISTet linksbündig angegebenen Screen
+                   nächster  Screen
-                   vorheriger Screen
R                   RESET - löscht alle eingeg. Kommandos
I                   INSERT Screen; Folgende rücken nach unten
D                   DELETE Screen; Folgende rücken nach oben
ER                  ERASE  Screen; bleibt gelöscht bestehen
CO<...              COPY angegebenen Screen in Angezeigten
CO>...              COPY angezeigten Screen in Angegebenen
CC ...              COPY mit X gekennz. Zeilen in angeg. Scr.
MM ...              MOVE mit X gekennz. Zeilen in angeg. Scr.
CAN                 CANCEL loescht eingegebene Änderungen

F 'xxx'             Finde   Zeichenfolge xxx im Screen
X 'xxx'             Lösche Zeichenfolge
C 'xxx' 'yyy'       Change - ersetze Zeichenfolge x durch y

Der F-,X- oder C-Befehl kann nach Ausführung mit
Control-N = NEXT ab Curser-Position wiederholt aufgerufen
werden; - auch in anderen Screens!

Alle Befehle in der Kommandozeile werden mit RETURN
zur Ausführung gebracht!



--ZEILENANFANG--

I     INSERT     Zeile einfügen, Folgende rücken nach unten
D     DELETE     Zeile entfernen, Folgende rücken nach oben
E     ERASE      Zeile löschen, bleibt aber bestehen
R     REPEAT     Zeile wird verdoppelt, Folgende r. n. unten
C     COPY       Zeile kopieren, bleibt aber bestehen
M     MOVE       Zeile verschieben, wird am Ziel eingefügt

B     BEFORE     gibt Ziel bei C und M an: vorher  einfügen
A     AFTER      gibt Ziel bei C und M an: nachher einfügen
X     EXTERN     kennzeichnet Zeile(n), die mit CC oder MM
                 in anderen Screen gebracht werden sollen.

Bei D, E und X können auch erste und letzte Zeile eines
Zeilen-Blockes gekennzeichnet werden.



--CONTROL-CODES--

CNTL H        BS          Cursor links
     I        HT          Cursor rechts
     J        LF          Cursor nach unten
     K        VT          Cursor nach oben
     C        COMMAND     Cursor nach Kommando-Zeile
     T        TOP         Cursor im Screen nach links oben
     F        FRONT       Cursor an Zeilenanfang
     R        REAR        Cursor an Zeilenende
     M        RETURN      CR + LF + Kommandoausführung
     S        INSERT      Zeichen-einfüge-Modus ein/aus
     D        DELETE      Zeichen löschen unter Cursor
     E        ERASE       Zeile löschen ab Cursor
     P        PRINT       Bildschirmseite ausdrucken
     B        BEFEHLE     Anzeige der verfügbaren Befehle
  ESCAPE                  Editor verlassen

                                              86250-3
```

Bild 3. Die Kommandos des FORTH-Full-Screen-Editors wurden einer kommerziellen Großrechenanlage nachempfunden. Hier sieht man sie in einer Übersicht.

Packages

Der HEX-Editor erlaubt einfaches Betrachten und ändern von Speicherinhalten. Gestartet wird mit

(Adresse) PM

Die in Tabelle 1 beschriebenen Kommandos werden (FORTH-untypisch) schon beim Drücken der entsprechenden Taste, ohne folgendes RETURN, erkannt und ausgeführt. Jede andere Taste führt zum Verlassen des Editors.

Einen formatierten Hex/ASCII-Dump erzeugt (wie könnte es auch anders sein

(Beginn-Adresse) (End-Adresse) DUMP

Zur laufenden Eingabe hexadezimaler Zahlen (etwa eines nur als Dump vorliegenden Maschinenprogramms) dient **HEX-INPUT** (mit vorheriger Angabe der Start-Adresse). Jeweils nach Eingabe einer Zeile (16 Hex-Zahlen) muß RETURN (neue Zeile) oder Control-E (Eingabe-Ende) gedrückt werden.

DOPPELT-GENAUE ZAHLEN (32 Bit) lassen sich mit dem Paket 2WORDS handhaben (4). Die Bedeutung der Worte (2DROP, 2DUP, 2SWAP, 2§, 2!, 2+!, 2OVER) ergibt sich aus ihren Entsprechungen für einfachgenaue Zahlen.

Sollte die an sich beträchtliche Verarbeitungsgeschwindigkeit von FORTH bei bestimmten Applikationen nicht ausreichen, steht auch ein 65C02-ASSEMBLER bereit ((5), (6), (7)) um strukturierten Maschinencode zu erzeugen. Beispiele für dessen Anwendung finden sich auf Disk 19/20 in Screen 42 und 62.

Das STRING-PACKAGE (8) enthält so (bei BASIC-Programmierern) bekannte Worte wie VAL, STR$, LEN, MID$, LEFT$, RIGHT$, deren Wirkungsweise als bekannt vorausgesetzt wird. Vor Verwendung der Worte dieses Packages muß allerdings erst das entsprechende Vokabular mit **STRING** aufgerufen werden. Für einen Teil der in Tabelle 2 enthaltenen Worte bestehen im FORTH-Vokabular

Tabelle 1.

+	zeigt den Inhalt der nächsten Adresse
-	zeigt den Inhalt der vorherigen Adresse
V	(vorwärts) schreibt den Inhalt der nächsten 16 Adressen auf den Bildschirm (Drucker)
R	(rückwärts) schreibt den Inhalt der vorhergehenden 16 Adressen auf den Bildschirm (Drucker)
C	(Character) bringt den Hexcode der nachfolgend gedrückten Taste an die an die angezeigte Adresse
X	(HEX) schreibt die nachfolgend eingegebene Hex-Zahl in den Speicher

4

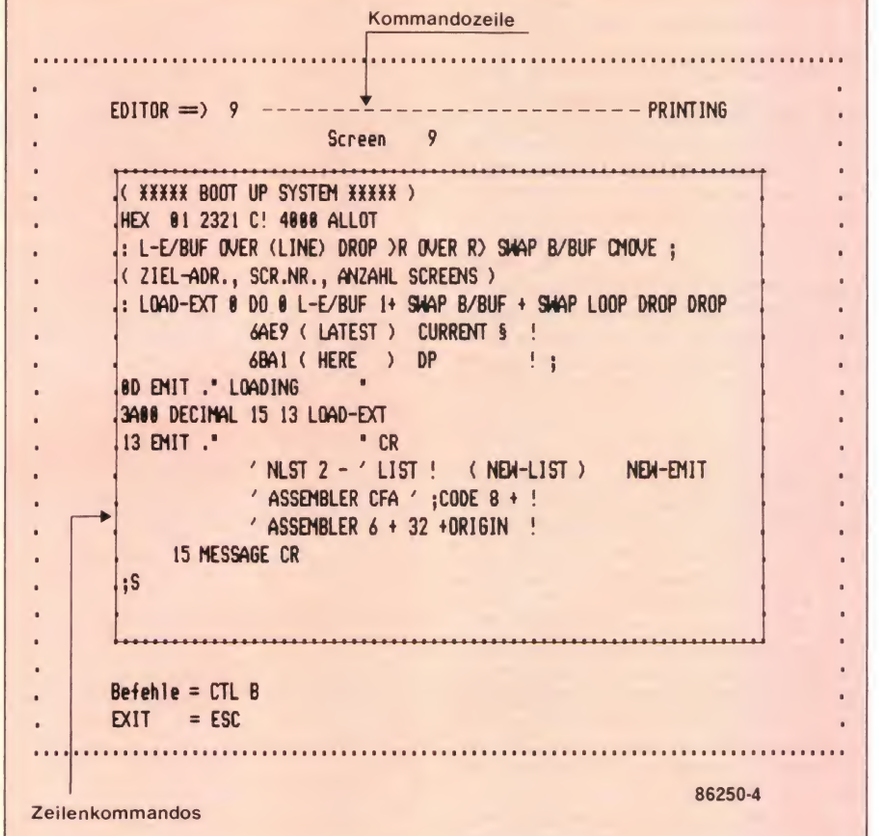


Tabelle 2.

Name ?	Ausgabe des benannten Strings
"	Eingabe eines Strings im Zwischenbuffer (Endemarkierung ebenfalls mit ")
Länge VARIABLE Name	Definition eines (leeren) Strings.
Name !	Abspeichern eines entweder mit " eingegebenen oder mit Namen aufgerufenen Strings in den vorher definierten String mit angegebenen Namen
Name +	addiert zwei Strings (siehe !)
=	vergleicht zwei Strings und übergibt entsprechendes Flag auf dem Stack (0=ungleich)
Name MLEN	legt die bei der Definition des Strings angegebene maximale Länge auf dem dem Stack ab

Tabelle 1. Die Kommandos des Hex-Editors.

Tabelle 2. Die Befehle (Worte) des String-Packages.

Bild 4. Der Full-Screen-Editor arbeitet mit einem Bildschirmfenster, in dem genau ein FORTH-Screen Platz findet.

Definitionen gleichen Namens. Dann muß der Programmierer je nach der gewünschten Funktion erst das entsprechende Vokabular aufrufen. Zum Beispiel addiert "+" im FORTH-Vokabular zwei Zahlen, im String-Vokabular dagegen zwei Strings.

FULL-SCREEN-EDITOR

Das umfangreichste Programm mit immerhin 35 Screens Source-Code stellt der Full-Screen-Editor dar. Er ermöglicht, sehr komfortabel Screens zu ändern, zu löschen, einzufügen und so weiter (der Befehlssatz wurde dem Editor einer kommerziellen Großrechenanlage nachempfunden). Eine Übersicht der verfügbaren Kommandos zeigt Bild 3. Aufgerufen wird der Editor unter dem Namen **EDI**. Auf die Definition eines eigenen Vokabulars wurde in diesem Fall bewußt verzichtet. EDI löscht zuerst den Bildschirm, übernimmt die Cursor-Steuerung und zeichnet ein Fenster, in dem genau ein Screen Platz findet (Bild 4). Die Kommandos aus der Kommandozeile bzw. vom Zeilenanfang im Screen führt der Editor jeweils nach Drucken der RETURN-Taste aus. Eine dritte Kommando-Ebene existiert mit den Control-Codes, die unmittelbar ausgeführt werden. Die Kommandozeile verläßt man mit RETURN, Control-J (LF) oder Control-T (Top des Screens). Die Cursor-Tasten entsprechen dem ASCII-Standard. Verläßt man den Editor mit ESCAPE, werden vorher alle eventuell noch eingegebenen Kommandos (Kommando-Zeile, Zeilenanfang) ausgeführt und alle geänderten Screens mit FLUSH automatisch

auf die Diskette zurückgeschrieben. Die Zeilenkommandos sind auf die linke Umrandung des Screens zu schreiben und bei Falscheingabe durch ein Space wieder zu löschen. Auf diese Umrandung gelangt man nur mit Control-H (BS). Durch die Zeilenkommandos D, E und X können auch mehrere Zeilen gleichzeitig angesprochen werden, indem das betreffende Kommando in die erste und letzte Zeile des betreffenden Zeilenblocks geschrieben wird (Achtung, Cursor nicht mit RETURN weiterbewegen).
Um einzelne Zeilen in einen anderen Screen zu kopieren ("moven"), ist nicht nur in der Kommandozeile **CC** (copy) oder **MM** (move) mit folgender Screen-Nummer einzugeben, die gewünschten Zeilen sind auch am Zeilenanfang mit X zu kennzeichnen. Nach RETURN wird der angegebene "Ziel-Screen" angezeigt, der Cursor springt direkt auf die linke Umrandung dieses Screens, und es sind nur noch die Cursor-Steuer-Tasten CTL-J (LF), CTL-K (VT) und RETURN (LF) aktiv. Die Ziel-Zeile muß nun mit A oder B bezeichnet werden. ESCAPE löst endgültig die Durchführung des Kommandos aus. Im Ziel-Screen kann danach weiter editiert werden. Sollen alle Änderungen im aktuellen Screen wieder rückgängig gemacht werden, löscht **CAN** (in der Kommando-Zeile) diesen Screen aus dem Pufferspeicher und liest ihn von der Diskette wieder (unverändert) nach. Da, wie erwähnt, vier Pufferspeicher vorhanden sind, lassen sich auch andere, schon vorher editierte Screens wieder "canceln" — man muß sich nur ihre Nummern gemerkt haben.

... näher betrachtet

Dieser Editor führt eine Reihe zum Teil sehr komplexer Funktionen aus, die sich in anderen Anwendungen ebenfalls sinnvoll mitverwenden lassen. Um Screens einzufügen bzw. zu löschen (**I, D** — Befehl in Kommandozeile), ruft EDI letztendlich (**SM** (= Screen-move) auf. Durch Angabe der ersten zu verschiebenden Screens, des letzten zu verschiebenden Screens, der Verschiebeentfernung (= Anzahl der Screens, die "übersprungen" werden sollen) und (SM) lassen sich ganze Blöcke von Screens auch über größere "Entfernung" hin verschieben. Die frei gewordenen Screens sind leer. Aus dem Vorzeichen der Verschiebeentfernung ergibt sich die Richtung, in die diese Screens bewegt werden sollen. Positiv bedeutet in Richtung höherer Screen-Nummern, negativ entgegengesetzt. Der I- oder der D-Befehl verwenden allerdings vorher das Wort **'ES** (empty-screen) mit Übergabe der gerade aktuellen Screen-Nummer. 'ES sucht in aufsteigender Richtung den nächsten leeren Screen (Blanks) und legt dessen Nummer auf dem Stack ab. Wird kein leerer Screen gefunden, bricht das DOS die weitere Verarbeitung mit Fehlerhinweis ab.
PRSCR (print screen) bedarf eigentlich keiner weiteren Erklärung. Es gibt den gerade aktuellen Bildschirminhalt auf dem Drucker aus, erzeugt dabei einen gepunkteten Rand, der die Bildschirmgrenzen anzeigt, und ersetzt nicht druckbare Zeichen ebenfalls durch einen Punkt (Bild 4 wurde so erzeugt). Jede Zeile hat daher normalerweise 82 Zeichen. Andere Bildschirmformate, die das Boot-EPROM kennt, werden aber auch beachtet. Am Ende jeder Zeile wird CR LF ausgegeben (Drucker entsprechend einstellen!).

Tabelle 3.

.CR		gibt Wagenrücklauf (CR) ohne Linefeed aus
CRS	(Anzahl -))	gibt die angegebene Anzahl CR LF aus
PON		schaltet die Ausgabe auf Drucker und Bildschirm
PONLY		schaltet die Ausgabe nur auf den Drucker
POFF		schaltet die Ausgabe wieder nur auf den Bildschirm zurück
.^	(Anzahl -))	schaltet den Drucker ein, gibt die angegebene Anzahl CR LF aus und schaltet den Drucker wieder aus (Papiervorschub)
JSR	(Adresse -))	springt die an der Adresse stehende Maschinenspracheroutine als Unterprogramm an (Rückkehr in das FORTH-Wort)
GO	(Adresse -))	springt in das an der Adresse stehende Maschinensprache-programm ohne Rückkehr nach FORTH (entspricht den DOS-Befehl GO)
CSR		löscht den Bildschirm
BOOT		bootet die in Laufwerk A befindliche Diskette
INH	(Screen from, Screen to -))	listet alle zwischen den angegebenen Nummern liegenden Screens (einschließlich) auf
.S		gibt den aktuellen Daten-Stack-Inhalt aus, ohne ihn zu zerstören
DUMP	(Adresse from, Adresse to -))	gibt formatierten Hex/ASCII-Dump aus
DQ	mit nachfolgender! Eingabe eines Wortnamens	decompiliert dieses Wort und gibt dessen ursprünglichen Source-Code aus. Dieser Decompiler arbeitet, in Gegensatz zu vergleichbaren anderen, besonders schnell
?BREAK		siehe Text

Tabelle 3. Eine Reihe weiterer nützlicher Worte wurde in die neue FORTH-Version eingearbeitet: Hier die wichtigsten.

Möchten Sie eigene Fenster definieren, muß die Größe des Fensters durch Angabe des freien Randes neben, ober- und unterhalb des Fensters festgelegt werden. Die jeweilige Breite des Randes ist in den Variablen **BSL** (linken Rand), **BSR** (rechten Rand), **BSO** (Rand oben) und — raten Sie mal — **BSU** (Rand unten) abzulegen. **CSR FEN** zeichnet das Fenster auf den gelöschten Bildschirm. Welche (Grafik-)Zeichen die Umrandung des Fensters bilden, legen die Variablen (**RW**) (waagerechte Linie) und (**RS**) (senkrechte Linie) fest. Screen 67 auf Disk 19 bzw. Screen 17 auf Disk 20 zeigen ein Beispiel, wie die Cursor-Steuerung für solch ein Fenster aussehen kann. Die dort aufgerufenen Cursor-Routinen stammen alle aus dem Full-Screen-Editor, dessen Source-Code ja auch vorliegt. Diese Routinen übergeben übrigens den aktuellen Standort des Cursors auch an die Video-Software im EPROM, so daß nach Verlassen eines derartigen Programmes der Cursor an definierbarer Stelle steht. Scrollen ist nicht möglich, da das Fenster ja nicht verlassen werden kann.

Das Wort **POS** errechnet aus einer angegebenen Zeile und Spalte die entsprechende Bildschirmadresse (2 20 POS

übergibt z.B. die Adresse des 20. Buchstabens in der 2. Zeile des Bildschirms, nicht des Fensters!). Mit **CUR!** kann der Cursor an dieser Stelle plaziert werden. POS und CUR! lassen auch eine Cursor-positionierung außerhalb des Fensters zu. Jeder Aufruf einer Cursor-Steuer-Routine führt allerdings den Cursor wieder in das Fenster zurück.

Anpassung

Sollte die Verarbeitung der Control-Codes (z.B. Cursor-Steuertasten) nicht mit Ihrer Tastaturbelegung übereinstimmen, läßt sich diese beim Full-Screen-Editor auch nachträglich ändern. Kopieren Sie sich Screen 116 auf Disk 19 bzw. Screen 61 auf Disk 21 in einen leeren Screen. Überschreiben Sie in der letzten Zeile das --> mit ;S. Die jeweils hinter CASE stehende Zahl bezeichnet den von der Tastatur kommenden Control-Code. Ändern Sie diese Codes nach eigenen Wünschen ab, beachten Sie aber, daß keine Doppelbelegungen vorkommen. Ein Code, der nicht mehr verwendet wird, sollte mit

CASE Control-Code CNTL 'ER

belegt werden (Ausgabe einer ERROR-Meldung bei Drücken dieser Taste), sonst wird die entsprechende Funktion auch mit dem alten Control-Code noch ausgeführt. Schließlich verlassen Sie EDI und laden diesen Screen mit TRY, und ab sofort arbeitet EDI mit der neuen Tastaturbelegung. Wenn Sie jetzt in Screen 9 an einer freien Stelle LOAD und die Nummer des gerade geladenen Screens eintragen, wird EDI nach dem Booten automatisch an Ihre Tastaturbelegungen angepaßt. Auf ähnliche Weise können auch zusätzliche, neue Kommandos installiert werden. Bei den Arrays für Kommandozeilen- bzw. Zeilen-Kommandos errechnet sich die Schlüsselzahl auf unterschiedliche Art aus dem Kommando. Eine detaillierte Erklärung würde den Rahmen dieses Artikels vollends sprengen, hier kann nur auf den Source-Code des Editors verwiesen werden.

Weitere Worte

In Tabelle 3 sind noch einige nützliche Worte beschrieben. In Klammern steht das Parameterpassing des jeweiligen Wortes, vor dem Pfeil die Eingabe-, danach die Ausgabe-Werte, die auf dem Stack übergeben werden.

?BREAK bedarf einer näheren Erklärung. Im ursprünglichen FORTH funktionierte der Befehl **?TERMINAL** (Abfrage, ob eine Taste im Moment gedrückt ist) nicht. Das wurde nun dahingehend berichtigt, daß beim Drücken von Control-C das entsprechende Flag gesetzt wird. Bauen Sie nun ?BREAK in ein Wort mit ein (am besten in einer Schleife), so kann dessen Ausführung mit Control-C sofort gestoppt werden. Verwendet wurde es z.B. in NLST (neuer LIST-Befehl), der in Screen 9 ins System eingebunden wird.

NEUER DOS-BEFEHL

Zwei einfache Möglichkeiten, in einem FORTH-Wort DOS-Kommandos zu verwenden, wurden angekündigt. Eine Erleichterung bietet das bereits angesprochene Wort JSR, das den direkten Sprung in die jeweilige DOS-Unterroutine vereinfacht. Als zweite Alternative wurde das Wort !DOS definiert. Hier muß zunächst ein String mit dem Namen DOS$ aufgebaut werden, der das gewünschte DOS-Kommando enthält, und dann mit !DOS zur Ausführung gebracht wird. Ein Beispiel:

: SELC STRING "SE C" DOS$! !DOS;

(Umschaltung auf Laufwerk C).

Im Screen 69/19 der Diskette 19/21 findet sich ein umfangreicheres Beispiel mit dem Wort DI-ALL, das den DOS-Befehl "DI" auf alle Spuren einer (80-Track)-Diskette anwendet. Dieses Wort DI-ALL darf allerdings nicht bei FORTH-Disketten verwendet werden, da sich das DOS "aufhängt". Besitzer von 40-Track-Laufwerken müssen zur Anpassung nur am Anfang der elften Zeile die 80 in 40 umändern.

Programme

Zum Schluß eine Übersicht der nur als Source-Code auf den Disketten befindlichen Programme.

Auf Disk 19 bzw. 20, Screen 58, steht eine Initialisierungs-Routine für neue FORTH-Disketten:

INIT-FORTH-DISK

schreibt ab Track 7 leere Screens auf die Diskette und füllt die Screens 4 und 5 mit den FORTH-ERROR-MESSAGES.

59 LOAD

(Disk 19 oder 20) lädt einen besonderen Maschinensprachedisassembler. Das Wort

DISASS (Adresse von, Adresse bis ->)

disassembliert ein im Speicher stehendes Programm, ähnlich dem im EPROM befindlichen Disassembler. Zusätzlich wird jedoch der Maschinencode als ASCII-Zeichen ausgegeben, um Text auch im Disassembler-Listing leicht erkennen zu können. Etwas besonderes leistet dagegen

INDIS (Anfangs-Track, End-Track, Adress ->)

Es disassembliert direkt von der Diskette. Die angegebenen Tracks müssen als ein Sektor mit acht Pages formatiert sein. Track 0 kann nicht verarbeitet werden. INDIS liest jeweils einen Track in seinen Puffer (ab Adresse 9800) und disassembliert dessen Inhalt unter Verwendung der angegebenen Startadresse des Maschinenprogramms. Beide Worte (DISASS und INDIS) verwenden ein in Screen 1 abgelegtes Maschinensprache-programm, das nach 9000 geladen wird. Wurde mit PON oder PONLY der Ducker aktiviert, verlangt der Disassembler die Eingabe einer Überschrift, die dann am Anfang jeder Druckseite ausgegeben wird. Die Anzahl der Druckzeilen pro Seite läßt sich in der Variablen DRUZEIL nach Wunsch einstellen (Default-Wert ist 72); z.B.

59 LOAD 60 DRUZEIL!

Mit

HEX 9000 9800 PON DISASS POFF

würde die Disassembler-Maschinenspracheroutine ausgeben (vorher wird vom Programm allerdings noch die Eingabe der Überschrift verlangt).

In Screen 63 (Disk 19/20) finden Sie ein kleines Spiel (Start mit **GAME**). Ziel ist, alle neun Spielfelder auf 1 zu setzen. Jeweils nach Eingabe einer Spielfeld-Nummer werden die Felder nach bestimmten Regeln verändert. Den Anfangszustand legt jeweils eine Zufallszahl fest.

Ein "Spiel" mit wissenschaftlichem Hintergrund (von dem englischen Mathematiker John Horton Conway erdacht und "Life" genannt) liegt auf Disk 19, Screen 65 (Disk 21, Screen 15). Mit seiner Hilfe lassen sich dynamisch "Veränderungen von Populationen lebender Organismen simulieren" (9). Gestartet wird mit **SDL**. Unter Verwendung der von EDI bekannten Cursor-Steuertasten muß zunächst eine beliebige Figur mit dem Stern * (Tastencode 2A) in das Fenster gezeichnet werden. Die ESCAPE-Taste beendet die Eingabe. Jetzt kann die Anzeige der Populationsveränderungen mit folgenden Tasten gesteuert werden:

L zeigt den Ablauf von 30 Generationen (schnell)

S zeigt die 30 Generationen langsam

1 führt nur einen Generationswechsel durch

E ermöglicht, in die bestehende Population neue Sterne einzufügen, weiter mit ESCAPE

Control-P druckt den Bildschirm aus

ESCAPE beendet das Programm

Grafik

Disk 19, Screen 71 (Disk 21, Screen 21) enthält einige Hilfsworte.

LOKO lädt die Kolorator-Software nach C000 aus den Screens 72 bis 75 (22 bis 25).

KIO setzt Input und Output auf die Kolorator-Software.

KOEX schaltet auf Text-Mode und setzt den Ouput auf die VDU-Software zurück.

Die Screens 76, 77 (26, 27) ermöglichen die Berechnung von Sinus und Cosinus (10) mit den Worten SIN und COS (Altgrad -> Sinus oder Cosinus). Eine Graphik-Anwendung findet sich in Screen 78 (28) mit dem Wort **3SIN**.

Viel Spaß.

Literaturhinweise:

(1) FORTH, Programmiersprache der vierten Generation, Elektor Verlag
(2) Die Programmiersprache FORTH, Franzis Verlag
(3) (2) Seite 199, Seite 303
(4) (2) Seite 299
(5) (1) Seite 116
(6) (2) Seite 268
(7) c't Heft 1/84, Seite 72
(8) (2) Seite 279
(9) Das Spiel — Naturgesetze steuern den Zufall, Piper Verlag
(10) c't Heft 10/85, Seite 87

Das Betriebssystem UCSD-p

Eine Einführung und Übersicht

Peter Treytl

Als Prof. Kenneth L. Bowles 1974 daran ging, am Institut für Angewandte Physik und Informatik der Universität von Kalifornien in San Diego (University of California San Diego, UCSD) ein Betriebssystem zu schreiben, das den von Prof. Wirth an der ETH in Zürich entwickelten Pascal-p-Code-Compiler auf Mini- und Mikroprozessorsystemen lauffähig machen sollte, rechnete er sicherlich nicht damit, daß dieses System zehn Jahre später zu den meistverkauften Betriebssystemen für Mikrocomputer gehören würde.

Entwickelt wurde das System für die PDP-11 der Firma Digital Equipment. Anfang 1978 gab es bereits Versionen für den 8080 und den Z80 und etwas später auch für die Prozessoren 6502, 6800, 8085, 9900 und andere. Mittlerweile existiert wohl kaum ein Rechner, der auf das UCSD-p-System verzichten müßte. Auch für den EC-65 und den EC-65K ist eine angepaßte Version verfügbar. Man darf gespannt sein, wie sich der Wettbewerb zwischen UCSD und den konkurrierenden Systemen, insbesondere aber dem weit verbreiteten Turbo-Pascal weiter entwickelt.

Das UCSD-p-System

Das UCSD-p-System wurde für die Programmentwicklung auf Mini- und Microcomputersystemen entwickelt, wobei an Peripherie zumindest ein Bildschirmarbeitsplatz (integrales oder vom Rechner abgesetztes Terminal) und ein Diskettenlaufwerk vorausgesetzt werden. Weitere Bedingung für die Programmentwicklung ist ein Programmspeicher mit mindestens 48 KByte.

Um die auf UCSD-p-Systemen entwickelten Programme portabel, d.h. auch auf unterschiedlichen Maschinen lauffähig zu machen, wird der p-Code, das ist der Maschinencode des von Prof. Wirth entwickelten symbolischen Prozessors, als Zwischencode verwendet. Hochsprachen wie Pascal, BASIC, FORTRAN-77 u.s.w. werden zunächst systemunabhängig in den p-Code kompiliert. Ein an den jeweiligen Prozessortyp angepasster p-Code-Interpreter setzt dann die Befehle des symbolischen Pseudo-Prozessors in die Maschinensprache des jeweils verwendeten Mikroprozessors um. Das erklärt einerseits die leichte Anpaßbarkeit des Systems. Erscheint am Markt ein neuer Prozessortyp, brauchen Compiler und Betriebssystem nicht neu geschrieben zu werden. Die Anpassung beschränkt sich auf die Erarbeitung eines neuen p-Code-Interpreters. Andererseits führt die Verwendung eines Zwischencodes natürlich zu längeren Ausführungszeiten, die weiderum nur von schnelleren Rechnern wie z.B. unserem EC-65K-System aufgefangen werden können.

Nun gehört zur praktischen Portabilität, d.h. zur Übertragbarkeit von Programmen von einem Rechner auf einen anderen nicht nur eine für beide Prozessoren gleichermaßen verständliche Sprache, sondern auch kompatible Formate der verwendeten Datanträger, z.B. Disketten. Hier beginnen meist die Schwierigkeiten. Prof. Bowles ging von dem weit verbreiteten Format IBM 3740 aus. Apple benutzt ein damit nicht verträgliches Format (GCR) und das Start-Stop-Verfahren von Ohio Scientific, das man auch beim Junior, EC-65 und EC-65K findet, ist mit keinem der beiden kompatibel. Von der Anwenderseite liegt so der Schluß nahe, die ohnehin nur teilweise vorhandene Portabilität im Sinne einer wesentlich erhöhten Verarbeitungsgeschwindigkeit auf den Quelltext, d.h. das Hochsprachen-Programm zu beschränken. Seit kurzem bietet die für den Vertrieb von UCSD-p-Systemen zuständige Firma PECAN Software Systems direkt kompilierende UCSD-Versionen für MS-DOS, PC-Dos, UNIX, VMS, MSX und viele andere Betriebssysteme an. Im Benutzerinterface unterscheiden sich diese aber praktisch nicht von den interpretierenden Varianten, sodaß die folgenden Ausführungen gleichermaßen für beide gelten.

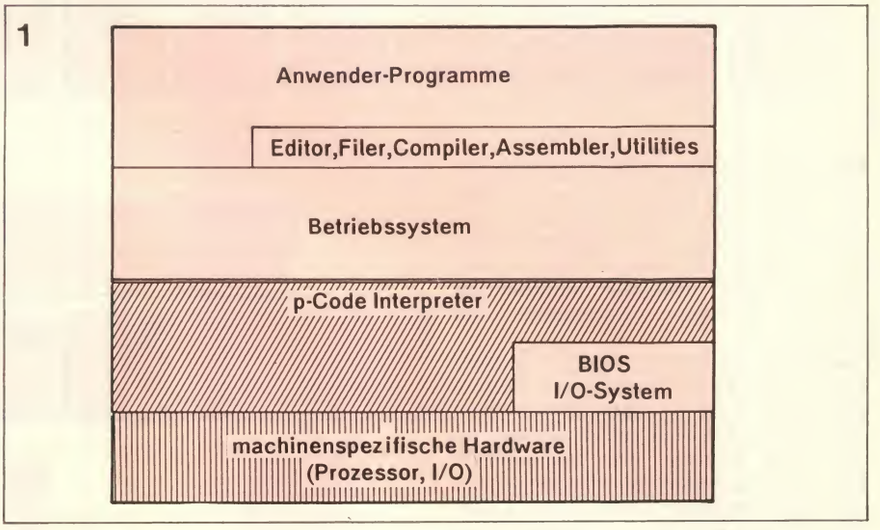


Bild 1. Der hierarchische Aufbau des UCSD-p-Systems.

Die Architektur des UCSD-p-Systems

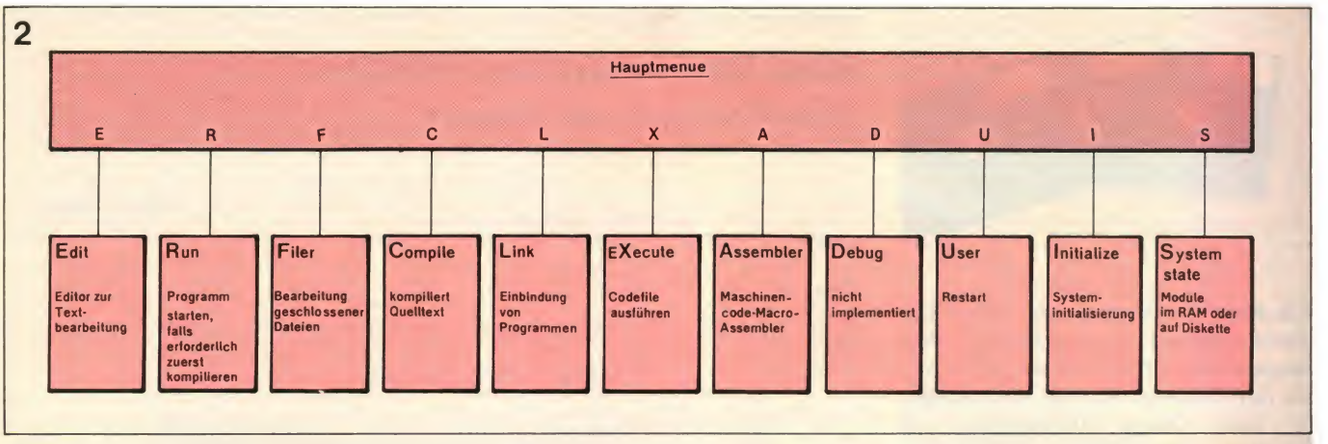
Moderne Betriebssysteme sind modular und strukturiert aufgebaut. Hierarchisch liegen, wie in Bild 1 dargestellt, die einzelnen Programme in Schichten übereinander. Die maschinenspezifischen I/O-Routinen für Bildschirm, Tastatur, Drucker, Echtzeituhr und Massenspeicher bilden das BIOS (Basic Input/Output System), das mit der rechnerspezifischen Hardware und dem p-Code-Interpreter zur Pseudo-p-Maschine (p-Maschinen-Emulator) zusammengefaßt werden kann. Darüber liegen das eigentliche Betriebssystem, die Dienst- und Anwendungsprogramme.

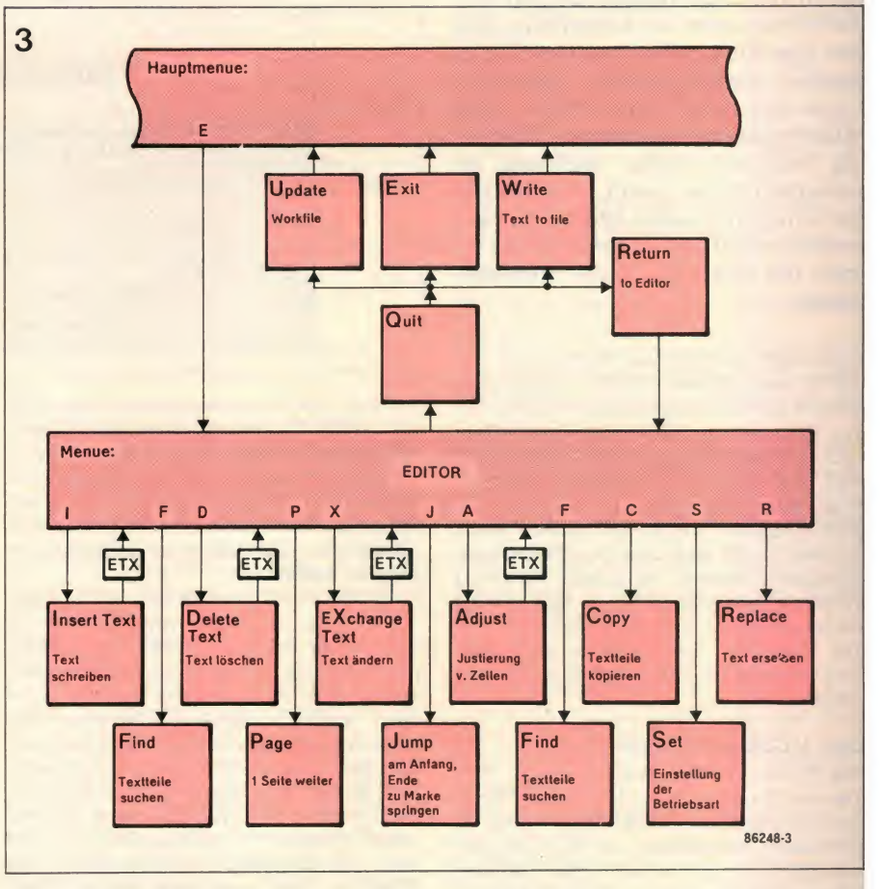
Nach dem Booten meldet sich das System mit einer allgemein gehaltenen Systemnachricht und dem Hauptmenue, das in der Statuszeile, der obersten Zeile des Bildschirms, erscheint. Durch Eingabe des Anfangsbuchstabens können entsprechend Bild 2 bereits vorhandene Programme ausgeführt werden, oder man verzweigt in tiefere Befehlsebenen.

Das Hauptmenue bietet die folgenden Wahlmöglichkeiten:

Edit lädt das Programm für die Bearbeitung von Quelltexten in den Arbeitsspeicher. Damit können Dateien erstellt oder aber auch verändert werden. Befindet sich in der Arbeitsdatei (Workfile) bereits ein Textfile, so wird dieses ebenfalls geladen.

Run führt das der Arbeitsdatei entsprechende Codefile aus. Existiert dieses nicht, so wird die Arbeitsdatei kompiliert und, wenn erforderlich, auch der Linker und die Programmbibliothek aufgerufen. Anschließend wird das Programm ausgeführt.





Filer ruft den Filehandler, das Bearbeitungsprogramm für geschlossene Dateien auf. Damit kann man u.a. das Inhaltsverzeichnis der Diskette lesen oder verändern. Dateien können neu angelegt oder verändert werden, auf andere Disketten übertragen oder gelöscht.

Compile übersetzt die Arbeitsdatei oder andere vom Nutzer bestimmte Quellprogramme. Im Falle von Syntaxfehlern kann der Editor aufgerufen und das Programm verbessert werden. Bei erfolgreicher Kompilierung entsteht ein Code-File, das mit "Run" gestartet werden kann. Dabei hängt es von der jeweils geladenen Datei "SYSTEM.COMPILER" und der zugehörigen Programmbibliothek "SYSTEM.LIBRARY" ab, in welcher Hochsprache (Pascal, FORTRAN-77, BASIC) das System arbeitet.

Link ermöglicht zusätzlich zur Programmbibliothek die Einbindung bereits kompilierter oder assemblierter Programmteile.

Execute startet in kompilierter Form abgespeicherte Programme.

Assembler ruft den maschinenspezifischen Macro-Assembler auf, der im Editor erstellte Quelltexte in absolute oder verschiebbare Maschinenprogramme übersetzt.

Debug ist ein unserer Information nach noch immer nicht implementiertes Programm. Es arbeitet ähnlich wie "Run", ermöglicht aber zusätzlich das Setzen von Breakpoints und erleichtert damit das Austesten syntaktisch richtiger, in der Ausführung aber fehlerhafter Programme.

User restart startet das System neu.

Initialize initialisiert das System neu, wobei, falls vorhanden, zuerst das File "SYSTEM.STARTUP" ausgeführt wird. Man hat dadurch die Möglichkeit, anwenderspezifische Initialisierungsroutinen vorab auszufuhren.

System state ermöglicht es, verschiedene Systemprogramme (File handlers, Screen handlers, Soft buffer handlers) überwiegend speicher- oder diskettenresident zu halten. Damit kann bei kleinen Systemen der für die Programmentwicklung verfügbare Speicherbereich vergrößert werden.

Im Folgenden werden die wesentlichen Module des Hauptmenues kurz beschrieben.

Textbearbeitung mit dem Editor

Das UCSD-System sieht den wahlweisen Einsatz unterschiedlicher Editoren vor. Allgemein wird dem bildschirmorientierten Editor der Vorzug vor dem zeilenorientierten gegeben.

Ergänzt wird der Editor durch ein Ausgabe- (Drucker-) Formatierungsprogramm für die Textverarbeitung. Dieses ist allerdings nicht bei allen UCSD-Varianten vorhanden.

Der Editor wird mit <E> aus dem Hauptmenue aufgerufen. Man befindet sich damit auf der mittleren Kommandoebene mit den in Bild 3 dargestellten Funktionen und noch einigen anderen. Im einzelnen bestehen die folgenden Wahlmöglichkeiten, wobei die Funktionen teilweise mehrfach und im Text sowohl vorwärts als auch rückwärts ausgeführt werden können:

Adjust verschiebt die vom Cursor gekennzeichnete Zeile.

Copy kopiert gekennzeichnete Textteile zur Position des Cursors.

Delete löscht von der Position des Cursors ausgehend alle Zeichen, über die dieser bewegt wird.

Find sucht in der vorgegebenen Richtung nach einem spezifizierten String.

Insert fügt Text ein. Es ist dies der normale Schreibmode.

Jump springt an den Anfang des Textes, an sein Ende, oder an eine vorher markierte Stelle.

Margin justiert Absätze zwischen Leerzeilen.

Page bewegt den Cursor eine Seite weiter.

Quit führt in das Haupmenü zurück. Dabei wird folgendes Untermenü aufgerufen:

— **Update** bringt die Arbeitsdatei auf den neuesten Stand.

Bild 2. Das Hauptmenü mit den erreichbaren Systemkomponenten.

Bild 3. Der Editor mit seinen vielfältigen Wahlmöglichkeiten.

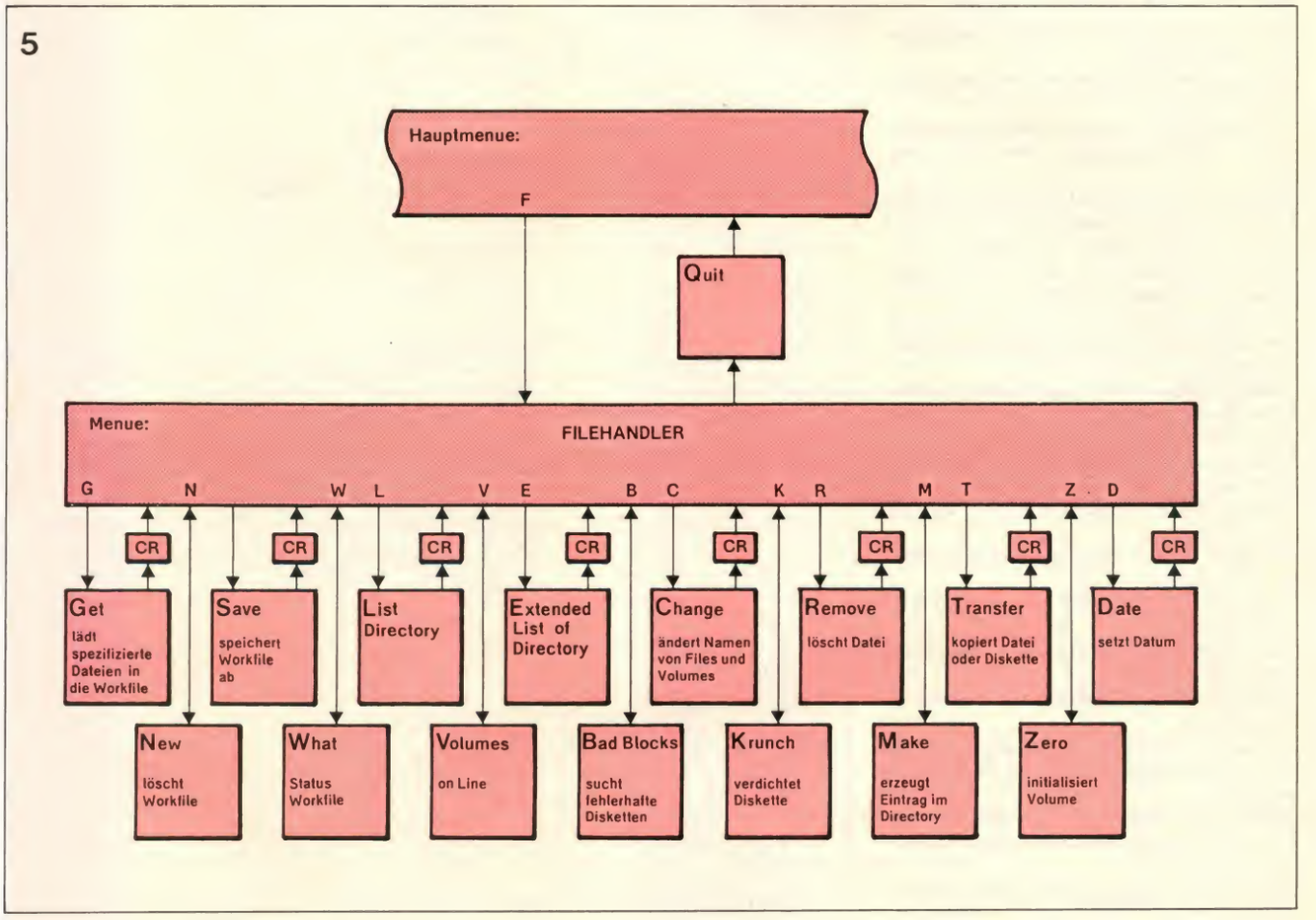
Bild 4. Directory Listing des UCSD-p-Systems.

Bild 5. Der Filehandler umfaßt ein vorbildliches Dateiverwaltungssystem.

4

```
IO-DISK:
BIOS.CODE           8 16-May-80     6   512  Codefile
INTCON.CODE         4 16-May-80    14   512  Codefile
EXTCON.TEXT         4 19-Sep-85    18   512  Textfile
EXTCON.CODE         3 17-Sep-85    22   512  Codefile
PRINT.DRIV.TEXT     4 19-Sep-85    25   512  Textfile
PRINT.DRIV.CODE     3 15-Sep-85    29   512  Codefile
EMPTYREM.CODE       3  5-May-80    32   512  Codefile
EMPTYUSR.CODE       3  2-May-80    35   512  Codefile
SYSTEM.LINKER      22  2-Jun-80    38   512  Codefile
SQUISH.CODE         8 18-Mar-80    60   512  Codefile
INTCOS.CODE         3 23-Nov-85    68   512  Codefile
INTCOS.TEXT         4 23-Nov-85    71   512  Textfile
MYGOTOXY.TEXT       4 23-Nov-85    75   512  Textfile
USR.TEXT            6  6-Feb-86    79   512  Textfile
USR.CODE            3  6-Feb-86    85   512  Codefile
< UNUSED >         64              88
15/15 files<listed/in-dir>, 88 blocks used, 64 unused, 64 in largest
```

5



— **Exit:** der Editor wird verlassen, ohne daß die Arbeitsdatei auf den neuesten Stand gebracht wird,
— **Write** legt den Text in einer spezifizierten Datei ab.
— **Return** kehrt zum Editor zurück.

Replace ersetzt in der vorgegebenen Richtung Textteile.

Set markiert den Text und ermöglicht die Voreinstellung bestimmter Betriebsmodi des Editors.

Verify stellt den Inhalt des Textspeichers auf dem Bildschirm so dar, daß sich der Cursor etwa in der Mitte des Textfensters befindet.

Exchange tauscht Text gegen neu einzugebenden aus.

Zap löscht den Text von der aktuellen Cursorposition bis zu einer Textmarke.

Die Befehle Insert, Delete, Exchange und Adjust werden mit <ETX> oder <CTRL-C> abgeschlossen. Der Text wird daraufhin in den Hauptspeicher übernommen. Durch das Zeichen <ESC> können vorher gemachte Eingaben unwirksam gemacht werden.

Zusammengefaßt steht einem mit dem Editor des UCSD-p-Systems ein vorbildliches Werkzeug zur Erstellung von Quelltexten zur Verfügung. Es gibt nur wenige Textverarbeitungssysteme, die ähnlich durchschaubar aufgebaut und komfortabel zu bedienen sind.

Die Dateiverwaltung (Filehandler oder Filer)

Neben dem Editor verfügt das UCSD-p-System über ein vorbildliches Dateiverwaltungsprogramm. Dieses benutzt die im Inhaltsverzeichnis eines Massenspeichers, z.B. einer Diskette gespeicherten Informationen zur Handhabung von Dateien.

Das UCSD-p-System unterscheidet im wesentlichen folgende Typen von Dateien:

.TEXT lesbarer Text
.CODE vom Rechner ausführbarer Code
.DATA Daten

Ein- und Ausgabegeräte wie Drucker, Tastatur oder Massenspeicher werden als "Volumes" bezeichnet. Volumes, die wie eine Diskette oder ein Festplattenlaufwerk intern blockstrukturiert sind, besitzen ein Inhaltverzeichnis (Directory). Vom Anwender können Volumes ebenso Namen bekommen wie Files. Letztere benötigen noch die Angabe des gespei-

cherten Datentyps. Das in Bild 4 gezeigte ausführliche Inhaltsverzeichnis einer Diskette läßt dies gut erkennen.
Eine Sonderstellung nimmt die Arbeitsdatei "Workfile" ein. In dieser befinden sich die Daten, mit denen gerade gearbeitet wird. Filer, Editor, Assembler und Compiler benutzen diese. Abgespeichert wird diese Datei unter den Namen STSTEM.WRK.TEXT bzw. SYSTEM.WRK.CODE, abhängig vom Datentyp.
Die wesentlichen Funktionen, die vom Filer aufgerufen werden können, zeigt Bild 5:

Get lädt eine spezifizierte Datei in die Arbeitsdatei.
Save speichert die Arbeitsdatei unter einem spezifizierten Namen ab.
New löscht die Arbeitsdatei.
Quit führt ins Hauptmenue zurück.

What informiert über Name und Status der Arbeitsdatei.
Volume listet die Volumes, die gerade on line sind, auf.
List directory listet das Inhaltsverzeichnis eines Massenspeichers oder Teile daraus.
Extended list arbeitet mit einem ausführlicheren Listing als der List-Befehl. Bild 4 ist ein Beispiel dafür. Ausgegeben werden die Dateinamen, der Dateityp, das Datum der letzten Bearbeitung der Datei, die laufenden Blocknummern ab der und bis der das Programm abgespeichert wurde und die Blocklänge in Bytes. Abschließend wird man noch darüber informiert, wieviele Blöcke bereits belegt und wieviele noch frei sind.
Change gibt die Möglichkeit, Namen von Volumes und Dateien zu ändern.
Remove löscht Dateien aus dem Inhaltsverzeichnis.
Transfer kopiert eine spezifizierte Datei in eine andere.
Date ermöglicht die Eingabe des aktuellen Datums.
Bad blocks durchsucht die Diskette auf fehlerhafte Blöcke.
Examine versucht fehlerhafte Blöcke zu reparieren.
Krunch verdichtet eine Diskette, indem noch unbenutzte Blöcke zusammengefaßt werden.
Make trägt einen spezifizierten Dateinamen ins Inhaltsverzeichnis des Massenspeichers ein.
Zero formatiert den spezifizierten Volume neu.
Editor und Filehandler bilden die Werkzeuge für Eingabe, überarbeitung und Verwaltung der Quelltexte. Assembler und Compiler erzeugen daraus lauffähigen Code. Dieser ist im Falle der Kompilierung ein Pseudo-Code, der vom p-Code-Emulator abgearbeitet wird. Der Assembler erzeugt spezifischen Maschinencode, der mittels des Linkers in Hochsprachenprogramm eingebunden werden kann.

UCSD-Pascal

Gegenüber dem im Pascal User Manual and Report von N. Wirth definierten Standard-Pascal ist das UCSD-Pascal in wesentlichen Punkten erweitert. Das betrifft insbesondere File- und Stringbearbeitung sowie die Ein- und Ausgabeverwaltung. Programme können segmentiert werden und **UNIT** ermöglicht den Aufruf unabhängiger Prozeduren, Funktionen und der zugehörigen Datenstrukturen aus der Programmbibliothek. Der Compiler erzeugt die nötigen Informationen, um die Module linken zu können.
Zur Erläuterung der Kompilierung eines Pascal-Programmes diene das kurze Beispielprogramm in Bild 6, das aus der Hypothenuse und einem anliegenden Winkel eines rechtwinkeligen Dreiecks die Länge der Katheten berechnet. Die erste Spalte des wiedergegebenen Compiler-Ausdrucks enthält die fortlau-

6

```
  1    1     1:D      1 (*$L PRINTER: *)
  2    1     1:D      1 PROGRAM DREIECK;
  3    1     1:D      3 (* Berechnet die Laenge der Katheten eines
  4    1     1:D      3 rechtwinkeligen Dreiecks *)
  5    7     1:D      3
  6    7     1:D      3   FUNCTION SIN(X:REAL):REAL;
  7    7     2:D      3   FUNCTION COS(X:REAL):REAL;
  8    7     3:D      3   FUNCTION EXP(X:REAL):REAL;
  9    7     4:D      3   FUNCTION ATAN(X:REAL):REAL;
 10    7     5:D      3   FUNCTION LN(X:REAL):REAL;
 11    7     6:D      3   FUNCTION LOG(X:REAL):REAL;
 12    7     7:D      3   FUNCTION SQRT(X:REAL):REAL;
 13    7     7:D      5
 14    1     1:D      5    USES TRANSCEND;
 15    1     1:D      3    CONST C = 5.23; ALPHA = 32.5;
 16    1     1:D      3    VAR A,B:REAL;
 17    1     1:D      7    P:TEXT;
 18    1     1:0      0 BEGIN
 19    1     1:1      0    REWRITE(P,'PRINTER:');
 20    1     1:1     37    A:=C*SIN(ALPHA*3.14159/180);
 21    1     1:1     72    B:=C*COS(ALPHA*3.14159/180);
 22    1     1:1    106    WRITELN(P,'A =',A:1:4);
 23    1     1:1    139    WRITELN(P,'B =',B:1:4);
 24    1     1:1    172    CLOSE (P)
 25    1     1:0    180 END.



A = 2.8101
B = 4.4109
```

86248-6

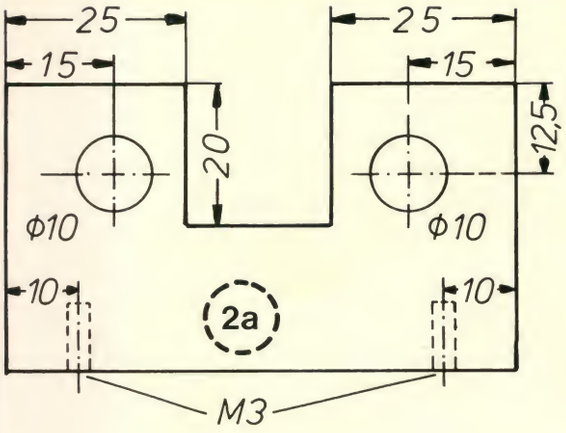
7

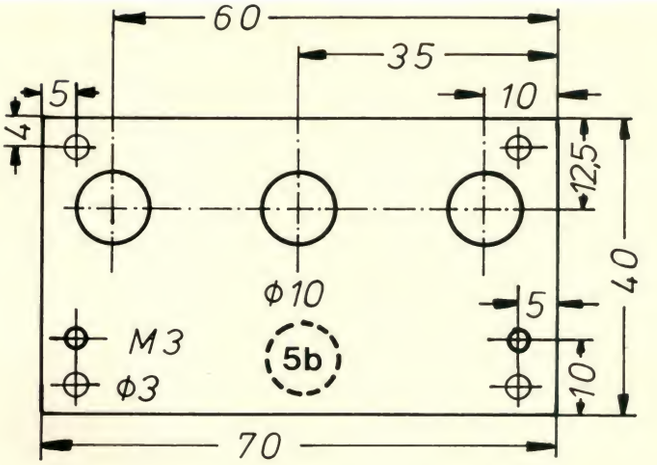
```
FORTRAN Compiler II.0  A1.2sU
    0.     0          PROGRAM RWDR
    1.     0 C        Berechnet die Laenge der Katheten eines
    2.     0 C        rechtwinkeligen Dreiecks
    3.     0 $SWAP
    4.     0 $XREF
    5.     0 1        C=5.23
    6.    10          ALPHA=32.5
    7.    20          A=C*SIN(ALPHA*3.14159/180.0)
    8.    53          B=C*COS(ALPHA*3.14159/180.0)
    9.    87          OPEN(6,FILE='PRINTER:')
   10.   106          WRITE (6,10) A,B
   11.   141 10       FORMAT('A =',F7.4/'B =',F7.4)
   12.   141          STOP
   13.   148          END

A       REAL           7                    7     10
ALPHA   REAL           5                    6      7      8
B       REAL           9                    8     10
C       REAL           3                    5      7      8
COS     INTRINSIC                           8
RWDR    PROGRAM
SIN     INTRINSIC                           7


RWDR    PROGRAM

14 lines.   0 errors.
Smallest available space = 3682 words.



A = 2.8101
B = 4.4109
```

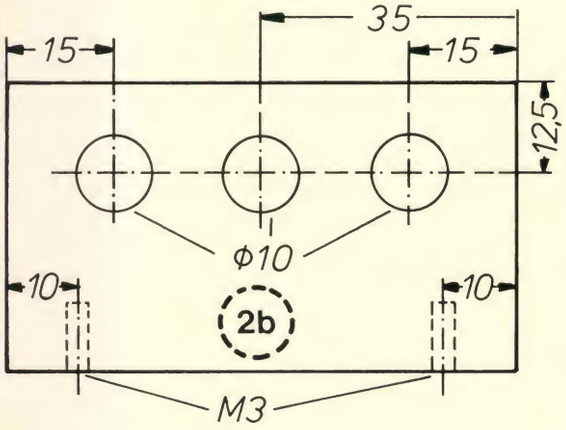
86248-7

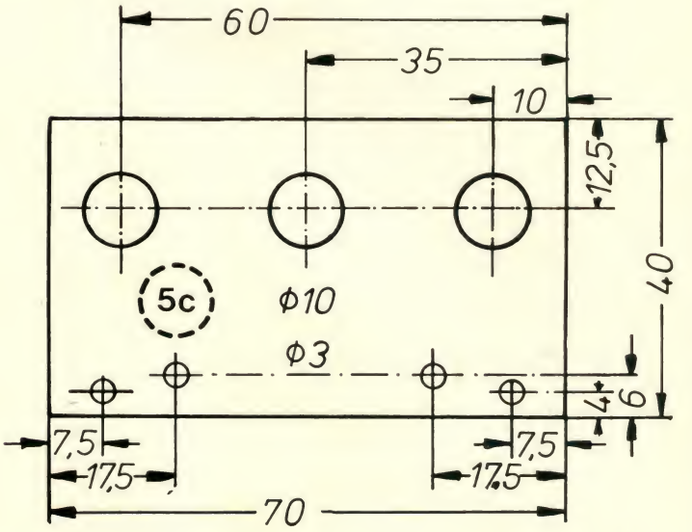
Bild 6. Beispielausdruck für den Pascal-Compiler.

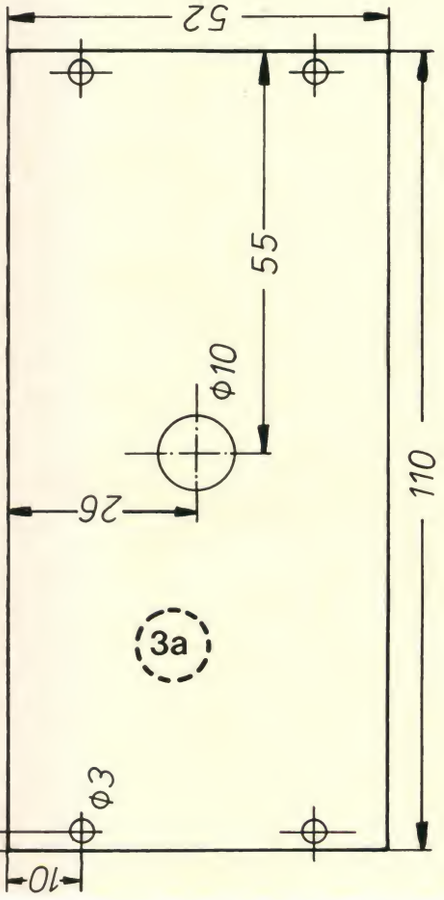
Bild 7. Beispielausdruck für den FORTRAN-Compiler. Das Programm entspricht dem aus Bild 6.

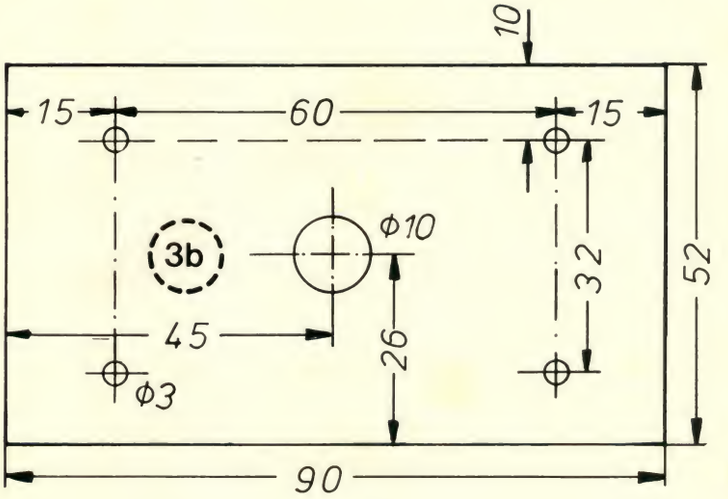

25
25
15
15
20
12,5
Φ10
Φ10
10
10
2a
M3

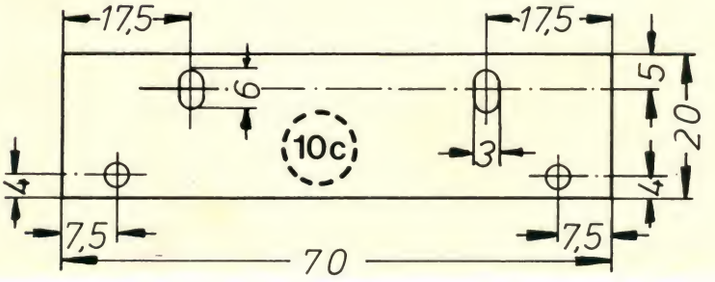

60
35
5
10
4
12,5
40
Φ10
5
M3
Φ3
5b
10
70


35
15
15
12,5
Φ10
10
10
2b
M3

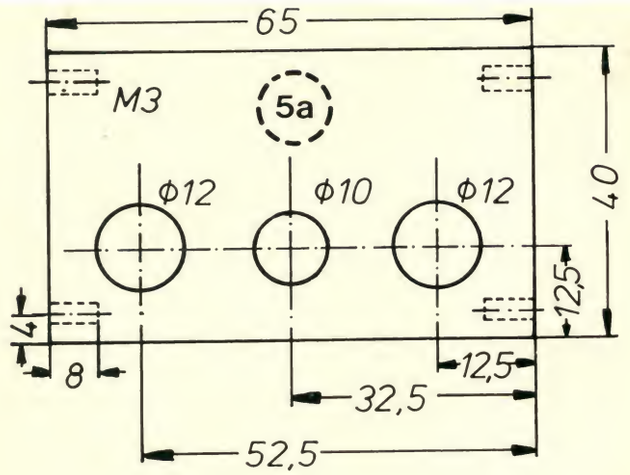

60
35
10
12,5
40
5c
Φ10
Φ3
4
6
7,5
7,5
17,5
17,5
70

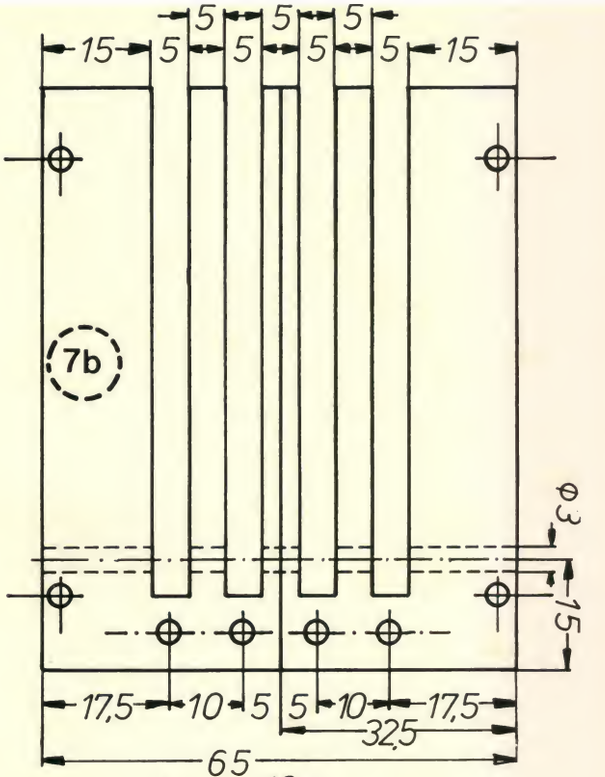

52
55
Φ10
26
110
3a
2,5
Φ3
10

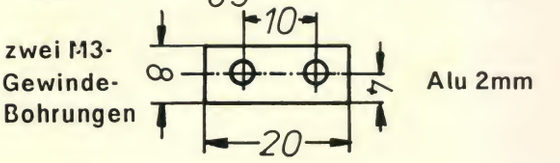

10
15
60
15
Φ10
3b
32
52
45
26
Φ3
90

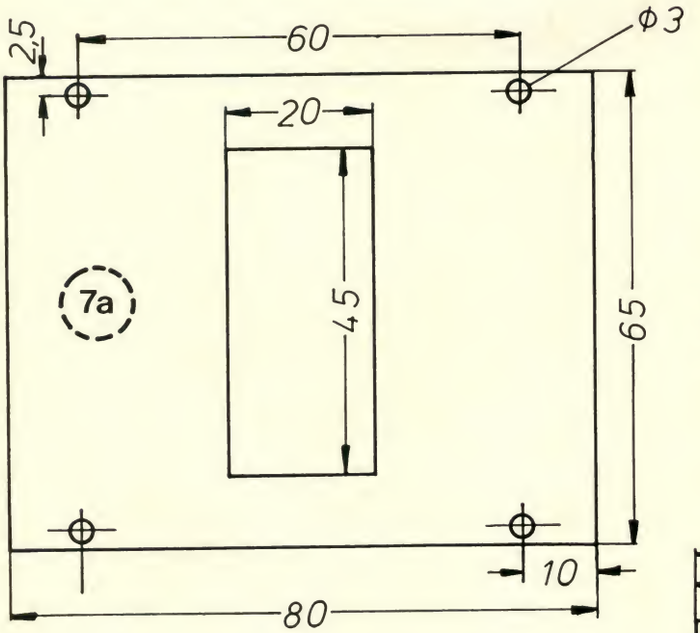

17,5
17,5
6
5
20
10c
3
4
4
7,5
7,5
70

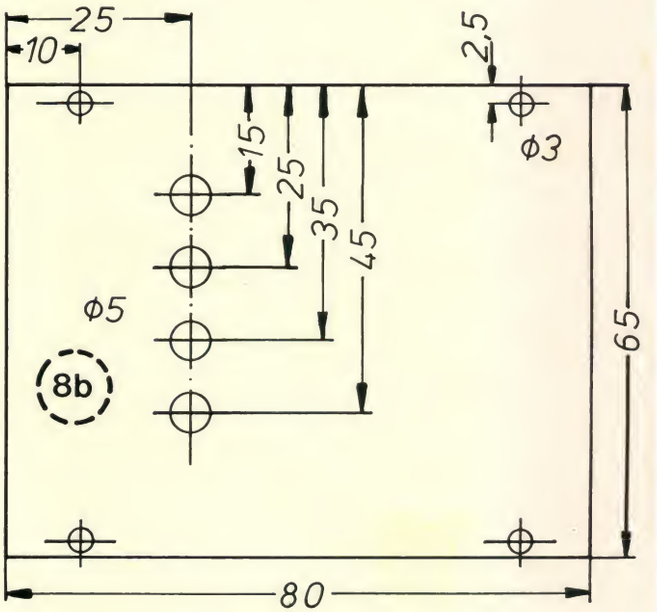
Φ3

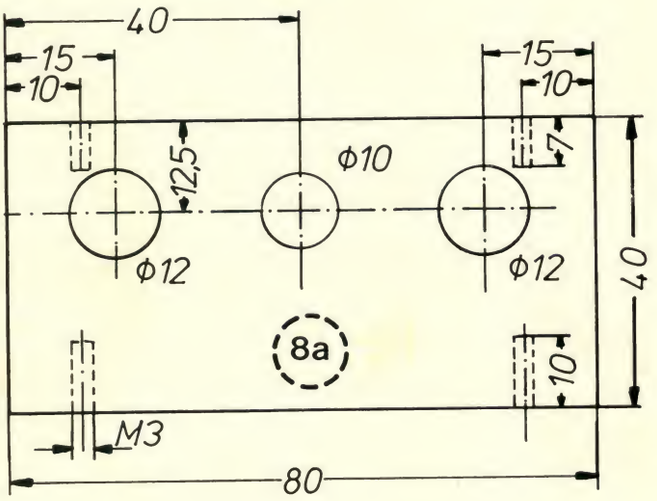


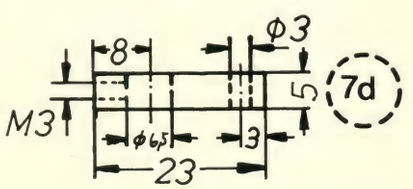


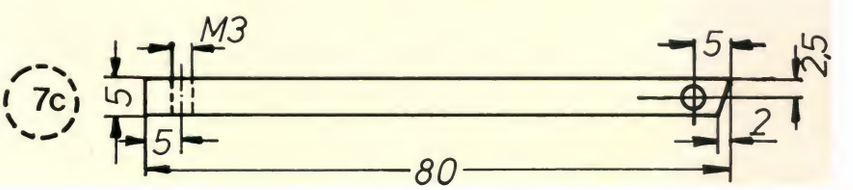












Zum Herausnehmen: Baubogen Plotter. Erläuterungen siehe Textteil

fende Zeilennummer, die zweite die Nummer des Programmsegments. Dem schließt sich durch Doppelpunkt getrennt die Indentifikationsnummer für die einzelnen Programmblöcke (das Hauptprogramm hat immer die Nummer 1) und eine Kennung D für Deklaration bzw. eine Ziffer an, die die Schleifentiefe angibt. Direkt vor dem Programmtext wird vor ausführbaren Statements angegeben, wieviele Bytes Code bisher erzeugt wurden.
Nach der Kompilierung des Quelltextes werden allenfalls im Programm verwendete Bibliotheksprogramme mit dem Linker eingebunden. Abschließend gibt der Compiler noch den Platzbedarf und falls der Rechner über eine Echtzeituhr verfügt, auch die für die Kompilierung benötigte Zeit aus. Das Code-File kann anschließend abgespeichert und mit dem Execute-Befehl gestartet werden.

FORTRAN-77

Der FORTRAN-Compiler des UCSD-p-Systems wurde von der Firma Silicon Valley Software Inc. entwickelt. Es handelt sich um eine fast vollständige Implementierung des ANSI-FORTRAN mit der auch für Pascal geltenden Einschränkung, daß Prozeduren nicht als Parameter übergeben werden können. Zusätzlich zur Standardsprache gibt es hilfreiche Compiler-Direktiven, u.a. die Möglichkeit des Ausdrucks einer Liste mit Kreuzverweisen (Cross Reference).
Um aus einem UCSD-Pascal ein FORTRAN-System zu machen, brauchen im übrigen nur der System-Compiler und die System-Bibliothek auf der Betriebssystemdiskette ausgetauscht zu werden. Editor, Filer und die anderen Hilfsprogramme stehen unverändert weiter zur Verfügung.
Das FORTRAN-Beispielprogramm in Bild 7 entspricht dem Pascal-Programm in Bild 6. Die Spalte neben der fortlaufenden Zeilennummer gibt wieder über den Umfang erzeugten Codes Auskunft. Dem Programm schließt sich ein alphabetisch geordneter Kreuzverweis mit Angabe des Datentyps an.

Der Assembler

Der UCSD-Macro-Assembler ist sowohl für die Entwicklung von Maschinenprogrammen geeignet, die später mit dem Linker in Pascal, FORTRAN oder BASIC-Programme eingebunden werden sollen, als auch für die Assemblierung eigenständiger Routinen. Er ist ebenso wie die Hochsprachencompiler im p-Code geschrieben. Assembliert wird in einem einzigen Durchlauf. Dies und einige andere Eigenheiten machen das Arbeiten mit dem Assembler anfangs nicht gerade leicht. Mit zunehmender Gewöhnung lernt man aber die zahlreichen Möglichkeiten mehr als zu schätzen.
Als sicherlich nützliches Beispiel zeigt Bild 8 eine Routine zur Einbindung der Echtzeituhr auf der PSIO-RTC-Karte des EC-65K. Dieses Modul kann einfach in das UCSD-p-System eingebunden werden. Dem Prozessortyp entsprechend wurde ein 6500-Assembler verwendet. Damit die Prozeduren, im vorliegenden Fall die Nutzerein- und Ausgabe .PROC

```
Page -  0  File:SYSTEM.WRK.TEXT     6500 Assembler   II.0 [e.2]
Page -  1           File:SYSTEM.WRK.TEXT  By  Peter Treytl, Febr. 1986


0000!                       ;
0000!                       .PROC USRIO
0000!                       .DEF USRREAD
0000!                       .DEF USRWRIT
0000!                       .DEF USRINIT
0000!                       .DEF USRSTAT
0000!                       ;
0000!                       .REF UNSTACK
0000!                       ;
0000!                       ;UNSTACK LOADS THE FOLLOWING PARAMETERS TO RAM:
0000!                       ;$A4,A5: BLOCK NUMBER
0000!                       ;$A6,A7: BYTE COUNT
0000!                       ;$A8,A9: POINTER TO BUFFER
0000!                       ;$AA,AB: DEVICE NUMBER
0000!                       ;$AC,AD: CONTROL WORD
0000!                       ;
0000! 20 0000               USRWRIT JSR UNSTACK
0003! A2 09                 USRSTAT LDX #09H
0005! 60                            RTS
0006!                       ;
0006! 20 0000               USRREAD JSR UNSTACK
0009! A2 09                 USRINIT LDX #09H
000B! 60                            RTS
000C!                       ;


Page -  2  USRIO     File:SYSTEM.WRK.TEXT   Symbol Table


AB - Absolute     LB - Label     UD - Undefined     MC - Macro
RF   Ref          DF - Def       PR - Proc          FC - Func
PB - Public       PV - Private   CS - Consts


UNSTACK  RF ----!  USRINIT  DF 0009!  USRIO    PR ----!
USRREAD  DF 0006!  USRSTAT  DF 0003!  USRWRIT  DF 0000!



Page -  3  USRIO     File:SYSTEM.WRK.TEXT  By  Peter Treytl, Febr. 1986


0000!                       .PROC RTC
0000!                       .DEF CLKREAD
0000!                       ;
0000!                       ;ADRESSES
0000!                       ;
0000! E430                  RTCADR .EQU 0E430H
0000! E431                  RTCDAT .EQU 0E431H
0000! 0050                  ASTORE .EQU 50H
0000! 0054                  BSTORE .EQU 54H
0000!                       ;
0000! A2 07                 CLKREAD LDX #7H
0002! 20 ****                       JSR CLEAR
0005! A0 04                 READCLCK LDY #4H
0007! 20 ****               READCL1 JSR READ
```

USRIO und die RTC-Routine .PROC RTC in das Betriebssystem eingebunden werden können, müssen mit .DEF Label definiert werden, über die der Zugriff des Betriebssystems zu den zusätzlichen Routinen sichergestellt wird. Umgekehrt können aber auch Label aus dem Betriebssystem mittels der Direktive **.REF** angesprungen werden. So ist **UNSTACK** ein Unterprogramm des BIOS, das von den Prozeduren USRIO und RTC mitbenutzt wird.

Nach Abschluß der Assemblierung eines jeden Moduls wird eine alphabetische geordnete Symbolliste ausgegeben.

Die zweite Prozedur in Bild 8, .PROC RTC errechnet die seit Mitternacht vergangene Zeit in Vielfachen einer sechzigstel Sekunde. Dazu müssen die Sekunden-, Minuten- und Stundenregister der Echtzeituhr ausgelesen, mit 60 multipliziert und zum neu auszulesenden Wert addiert werden. Das Ergebnis ist als Binärzahl auf dem Stack zu übergeben.

Daß es sich um einen "single pass Assembler" handelt, merkt man spätestens auf Seite 3 des Listings an der Adresse 0002, bei der sich der Unterprogrammsprung sum Label Clear auf eine Adresse bezieht, die erst später, nämlich bei der Adresse 0027 definiert werden wird. Der Assembler setzt daher als vorläufige Adresse **** ein und trägt die tatsächliche Adresse später nach (Adresse 0003* nach Adresse 0027). Ähnlich wird bei den mit .DEF bzw. .REF gekennzeichneten symbolischen Adressen verfahren. Hier wird als vorläufige Adresse 0000 eingesetzt. Die echte Adresse wird beim Linken automatisch berücksichtigt.

Utilities, die Hilfsprogramme des UCSD-p-Systems

Um den Benutzer den Umgang mit der Software zu erleichtern, bietet das UCSD-System eine größere Zahl von Hilfsprogrammen:

LBRARY.CODE ermöglicht es, bis zu 16 separat kompilierte oder assemblierte Programme in die Programmbibliothek aufzunehmen. Sie können dann einfach mit USES. aus einem Programm aufgerufen werden.

SETUP, GOTOXY und **SCREENTEST** helfen, ein spezielles Terminal oder eine besondere Videokarte einschließlich Tastatur an das System anzupassen.

PATCH ist ein komfortabler bildschirmorientierter Hexmonitor zur Überprüfung von Dateien.

Der p-Code Disassembler **DISASM** liest p-Code Files und disassembliert sie. Er hilft auch bei statischen Untersuchungen zur Verbesserung der Architektur des p-Codes.

Darüber hinaus gibt es noch Hilfsprogramme zum Formatieren und Kopieren von Disketten, zur Anpassung des Systems an die vorgegebene Speicherarchitektur, zur Wiederherstellung verlorengegangener Inhaltsverzeichnisse u.s.w.

Zusammenfassung und Ausblick

Das UCSD-System ist sicher eines der komfortabelsten und umfassendsten professionellen Betriebssysteme für Mini- und Microcomputer. In seiner Vielseitig-

8

```
000A! 88                        DEY
000B! 88                        DEY
000C! 10F9                      BPL READCL1
000E! 68                PLSTACK PLA
000F! AA                        TAX
0010! 68                        PLA
0011! 8D ****                   STA TOSTACK1+1
0014! A0 00                     LDY #0
0016! B9 54 00          TOSTACK LDA BSTORE,Y
0019! 48                        PHA
001A! C8                        INY
001B! C0 04                     CPY #4
001D! 30F7                      BMI TOSTACK
0012* 2000
001F! A9 00             TOSTACK1 LDA #0
0021! 48                        PHA
0022! 8A                        TXA
0023! 48                        PHA
0024! A2 00                     LDX #0
0026! 60                        RTS
0027!                           ;
0003* 2700
0027! A9 00             CLEAR   LDA #0
0029! 95 50                     STA ASTORE,X
002B! CA                        DEX
002C! 10F9                      BPL CLEAR
002E! 60                        RTS
002F!                           ;
0008* 2F00
002F! A2 03             READ    LDX #3
0031! 20 2700                   JSR CLEAR
0034! A2 0A             READ1   LDX #0AH
0036! 8E 30E4                   STX RTCADR
0039! AD 31E4                   LDA RTCDAT
003C! 30F6                      BMI READ1
003E! 98                        TYA
003F! 48                        PHA
0040! 8C 30E4                   STY RTCADR
0043! AD 31E4                   LDA RTCDAT
0046! 48                BCDTOBIN PHA

Page -  4  RTC      File:SYSTEM.WRK.TEXT  By  Peter Treytl, Febr. 1986

0047! 4A                        LSR A
0048! 4A                        LSR A
0049! 4A                        LSR A
004A! 4A                        LSR A
004B! AA                        TAX
004C! 68                        PLA
004D! 29 0F                     AND #0FH
004F! 18                        CLC
0050! 7D ****                   ADC TABLE,X
0053! A2 03                     LDX #3
0055! 95 50                     STA ASTORE,X
0057! 18                ADDB    CLC
0058! B5 50             ADDB1   LDA ASTORE,X
005A! 75 54                     ADC BSTORE,X
005C! 95 54                     STA BSTORE,X
005E! 95 50                     STA ASTORE,X
0060! CA                        DEX
```

keit und seinen Programmiermöglichkeiten hebt es sich deutlich von Turbo-Pascal ab, ohne allerdings in der interpretierenden p-Version dessen Geschwindigkeit zu erreichen. Der Einsatz schnellerer Prozessoren und Massenspeicher kann hier einen gewissen Ausgleich schaffen. Man darf aber auch auf erste Testberichte über die direkt kompilerenden UCSD-Versionen gespannt sein, die sich auch geschwindigkeitsmäßig vorteilhaft mit Turbo-Pascal vergleichen. Der dafür genannte Preis von $80,— wird sicherlich mit dazu beitragen, daß das UCSD-System zukünftig noch größere Verbreitung findet.

Literatur:

Kenneth L. Bowles, Beginner's Guide for the UCSD Pascal System, McGraw-Hill Book Company
Kenneth L. Bowles, UCSD Pascal: A (Nearly) Machine Independent Software System, Byte, May 1978, 46 ff.
Kenneth L. Bowles, Microcomputer Problem solving using Pascal, Springer Verlag
UCSD Pascal Users Manual, Softec Microsystems, San Diego, CA.
FORTRAN User Reference Manual, Softec Microsystems, San Diego, CA.
Kathleen Jensen, Niklaus Wirth, Pascal User Manual and Report, Springer Verlag.
N. Wirth, The Programming Language Pascal, Acta Informatica 1, 35-63, 1971
N. Wirth, The Design of a Pascal Compiler, Software-Practice and Experience, Vol. 1, 309-333. 1971.

8

```
0061! 10F5                      BPL ADDB1
0063! A2 05             MULTA64 LDX #5
0065! 06 57             MULTA2  ASL BSTORE+3
0067! 26 56                     ROL BSTORE+2
0069! 26 55                     ROL BSTORE+1
006B! 26 54                     ROL BSTORE
006D! CA                        DEX
006E! 10F5                      BPL MULTA2
0070! A0 03             SUB4B   LDY #3
0072! A2 03             SUB4B1  LDX #3
0074! 38                        SEC
0075! B5 54             SUB4B2  LDA BSTORE,X
0077! F5 50                     SBC ASTORE,X
0079! 95 54                     STA BSTORE,X
007B! CA                        DEX
007C! 10F7                      BPL SUB4B2
007E! 88                        DEY
007F! 10F1                      BPL SUB4B1
0081! 68                        PLA
0082! A8                        TAY
0083! 60                        RTS
0084!                           ;
0051* 8400
0084! 00 0A 14 1E 28 32 TABLE .BYTE 0,0AH,14H,1EH,28H,32H
008A!                           ;
008A!                   .END
```

8

```
Page -  5  RTC       File:SYSTEM.WRK.TEXT   Symbol Table


AB - Absolute     LB - Label     UD - Undefined     MC - Macro
RF   Ref          DF - Def       PR - Proc          FC - Func
PB - Public       PV - Private   CS - Consts


ADDB     LB 0057!  ADDB1    LB 0058!  ASTORE   AB 0050!  BCDTOBIN LB 0046!  BSTORE   AB 0054!  CLEAR    LB 0027!  CLKREAD  DF 0000
MULTA2   LB 0065!  MULTA64  LB 0063!  PLSTACK  LB 000E!  READ     LB 002F!  READ1    LB 0034!  READCL1  LB 0007!  READCLCK LB 0005
RTC      PR ----!  RTCADR   AB E430!  RTCDAT   AB E431!  SUB4B    LB 0070!  SUB4B1   LB 0072!  SUB4B2   LB 0075!  TABLE    LB 0084
TOSTACK  LB 0016!  TOSTACK1 LB 001F!


                                Page -  6  RTC       File:SYSTEM.WRK.TEXT  By  Peter Treytl, Febr. 1986

                              Assembly complete:       117 lines
                                   0   errors flagged on this assembly
```

Bild 8. Hier hat der UCSD-Assembler das Programm zur Einbindung der PSIO-RTC-Karte in das UCSD-Betriebssystem aufgelistet.

Anmerkungen der Redaktion:
Das UCSD-p-Pascal für den EC-65(K) wird mit einer umfassenden Dokumentation geliefert.
Der in der Literaturliste genannte Springer-Verlag ist ein international anerkannter wissenschaftlicher Verlag und nicht mit dem gleichnamigen Zeitungsverlag zu verwechseln.

MSX®
REVUE
DAS MAGAZIN FÜR FREUNDE DER KOMPATIBLEN
DM 5,80/ÖS 49/SFR 5,80

IM TEST: Yamaha CX 5

SERVICE: Welcher Drucker für MSX?

IM TEST: Text-verarbeitung

SERVICE: Basic-Befehle

IM TEST: Joy-sticks

SERVICE: Über 20 Seiten MSX-Listings

Heimcomputer-Bastelkiste

Für alle, die mehr mit ihrem Heimcomputer anfangen wollen und über ein wenig Bastelgeschick und elektronisches Verständnis verfügen, ist dieses Buch eine wahre Fundgrube.
Hier finden sich zahlreiche Anwendungen unterschiedlichen Schwierigkeitsgrades mit Funktionserklärungen, Bauteilbeschreibungen und Verdrahtungslisten — jeweils leicht nachvollziehbar. Neben den Schnittstellen — den "Toren" des Computers zu anderen Bauteilen und Geräten mit ihren vielfältigen Funktionen — werden auch elektronische Bauelemente berücksichtigt, die den menschlichen Bewegungsapparat und die fünf Sinne zu ersetzen vermögen.
Programmbeschreibungen, getestete Listings für die verschiedenen Heimcomputer mit Variablenlisten und Kommentaren sowie eine Einführung in die Maschinensprache für die Mikroprozessoren Z80 und 6502 runden den Band ab.

ISBN 3-8068-4309-0
DM 29,50

Elektor Verlag, Aachen

Zu allen ec 5 Bauprojekten gibt's elektor-Platinen

Bestellen Sie diese mit der Karte am Heftende.

Elektor Verlag, Aachen

Programmieren unter CP/M

Das Know-How zum Programmieren in Assembler unter CP/M

Frank Schmidt

In ec 2 haben wir unter anderem den CP/M-Computer "Big Board II" vorgestellt und bei der Gelegenheit auch gleich erläutert, wie man mit seinem Betriebssystem umgeht (Artikel "CP/M in 90 Minuten" in ec 2). Mittlerweile ist auch der EC 65 CP/M-fähig geworden, und es ist an der Zeit, daß wir unseren Lesern sagen, wie CP/M intern aufgebaut ist, was man bei der Programmierung in Assembler zu beachten hat, und wie man die Ein/Ausgaben zu beliebigen Geräten (z.B. Terminal, Diskettenlaufwerke) mit Unterstützung des BDOS von CP/M realisieren kann. Hierbei wird übrigens auch erklärt, was ein "BDOS" überhaupt ist!

CP/M gliedert sich in drei logische Teile: BIOS, BDOS und CCP. Die unterste Betriebssystem-Ebene bildet das BIOS (Basic Input Output System). In ihm sind die sogenannten "physikalischen" Gerätetreiber abgelegt. Außerdem haben hier verschiedene Tabellen ihren Platz, in denen die Eigenschaften der ansprechbaren Diskettenlaufwerke eingetragen sind (z.B. Kapazität pro Track, Gesamtzahl der Tracks usw.). Aber auch logische Eigenschaften (wie z.B. die Größe des Inhaltsverzeichnisses) sind hier kodiert.

Schale 1: das BIOS

Das BIOS erledigt so grundlegende Sachen wie das Senden von Zeichen an die Konsole, das Lesen eines Sektors von der Diskette oder das Positionieren des Schreib/Lesekopfes eines Laufwerkes über einer bestimmten Spur. Das Interface zu den anderen CP/M-Programmen, vornehmlich dem BDOS, bildet eine Sprungleiste am Anfang des BIOS. In ihr werden bei CP/M 2.2 insgesamt 17 verschiedene Funktionen zur Verfügung gestellt. Sie sind in Bild 1 ausführlich aufgeführt, sollten jedoch im allgemeinen nicht vom Anwender aufgerufen werden, um Kompatibilitätsprobleme mit anderen CP/M-Versionen zu vermeiden.

Nun sind aber all diese BIOS-Funktionen sehr hardwareabhängig. Deshalb muß das BIOS für jeden Computer neu geschrieben werden. Das BIOS ist also bei zwei verschiedenen CP/M-Computern unterschiedlich. Es ist allerdings auch der einzige Teil von CP/M, der hardware-abhängig ist. BDOS und CCP sind auf allen CP/M-Computern identisch, sofern sie mit der gleichen Versions-Nummer arbeiten). Die Implementation von CP/M auf einen Rechner beschränkt sich also im wesentlichen auf das Schreiben eines neuen, der Hardware angepaßten Gerätetreibers, des BIOS. Auf ihm bauen alle anderen Programme auf. In Bild 2 stellt das BIOS deshalb die innerste Schale des Betriebssystems dar.

1

```
BIOS+00H    JMP BOOT      Einsprung Kaltstart
BIOS+03H    JMP WBOOT     Einsprung Warmstart
BIOS+06H    JMP CONST     Hole Konsolen-Status
BIOS+09H    JMP CONIN     Eingabe von Konsole
BIOS+0CH    JMP CONOUT    Ausgabe an Konsole
BIOS+0FH    JMP LIST      Ausgabe an Drucker
BIOS+12H    JMP PUNCH     Ausgabe auf "Lochstreifenstanzer"
BIOS+15H    JMP READER    Eingabe von "Lochstreifenleser"
BIOS+18H    JMP HOME      Laufwerkskopf auf Track 0
BIOS+1BH    JMP SELDSK    Laufwerk anwählen
BIOS+1EH    JMP SETTRK    Tracknummer setzen
BIOS+21H    JMP SETSEC    Sektornummer setzen
BIOS+24H    JMP SETDMA    Sektoradresse setzen
BIOS+27H    JMP READ      angewählten Sektor lesen
BIOS+2AH    JMP WRITE     angewählten Sektor schreiben
BIOS+2DH    JMP LISTST    Hole Drucker-Status
BIOS+30H    JMP SECTRAN   Sektornummer übersetzen
```

Bild 1. Diese Sprungtabelle bildet das Interface zwischen BDOS und BIOS. Für eigene Programme sollte man sie nach Möglichkeit nicht benutzen, um Kompatibilität mit anderen CP/M-Versionen zu wahren: Schon bei CP/M 3.0 führt das normalerweise unweigerlich zum Absturz!

2

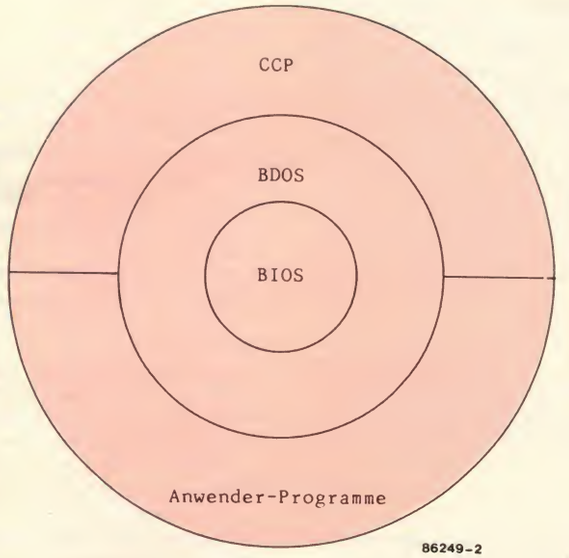


Bild 2. Das Schalenmodell des Betriebssystems. Ein Datenaustausch sollte im allgemeinen nur dort geschehen, wo wirklich Grenzen gezogen sind.

Schale 2: Das BDOS

Aufbauend auf dem BIOS sorgt das BDOS (Basic Disk Operating System) für eine komfortablere Verwaltung der Ein/Ausgabegeräte. Es stellt die eigentliche Schnittstelle der Anwender-Programme zur Hardware dar und wird deshalb noch sehr ausführlich besprochen werden. Auch das BDOS stellt unter anderem Funktionen zum Einlesen von Zeichen von der Konsole zur Verfügung. Aber diese sind zum großen Teil sehr viel leistungsfähiger als die des BIOS. So läßt sich z.B. mit einem einzigen BDOS-Aufruf die Eingabe einer kompletten Zeile vom Terminal realisieren.
Ferner besorgt das BDOS das File-Handling, das auf dieser Betriebssystem-Ebene nicht mehr über die Angabe von Spur und Sektor, sondern ausschließlich über die Angabe von Dateinamen abgewickelt wird. Die entsprechenden Umrechnungen übernimmt das BDOS. Es stellt dabei alle Funktionen zur Verfügung, die zur Verwaltung der Massenspeicher und der darauf befindlichen Inhaltsverzeichnisse nötig sind.

Schale 3: Der CCP und die Anwenderprogramme

Der dritte Betriebssystem-Teil heißt "Console Command Processor" oder kurz CCP. Seine Aufgabe ist es, Befehle vom Benutzer entgegen zu nehmen und diese dann auszuführen. Wenn Sie Ihren Rechner einschalten und es erscheint der Prompt "A>", dann sitzt der CCP schon in den Startlöchern und wartet auf Ihren Startschuß. Geben Sie nun z.B. "DIR" ein, so erkennt er dieses als einen seiner fest eingebauten Befehle, führt ihn aus und meldet sich wieder mit "A>" zurück. Tippen Sie dagegen "STAT *.*" ein, so fühlt sich der CCP allein überfordert und durchblättert hilfesuchend das Inhaltsverzeichnis des momentan angesprochenen Laufwerks nach einer Datei namens "STAT.COM". Ist diese nicht vorhanden, gibt er eine Fehlermeldung aus. Andernfalls wird das File "STAT.COM" geladen und dann gestartet. All das ist Sache des CCP. Mit diesem kurzen Beispiel ist wohl seine Aufgabe für's erste klar genug beleuchtet geworden.
Bleibt noch zu erwähnen, daß der CCP fleißig Gebrauch von der Intelligenz des BDOS macht und deshalb zusammen mit allen anderen Anwender-Programmen seinen Platz in der dritten Betriebssystem-Schale hat.

Lokalisation

Bild 3 zeigt die Speicheraufteilung eines beliebigen CP/M-Systems. Die Adressen 0000h bis 0100h (das "h" am zeigt an, daß es sich um hexadezimale Zahlen handelt) sind für Betriebssystem-Belange reserviert und enthalten in erster Linie Systemparameter und Puffer. Bild 4 zeigt die Belegung dieses Bereiches, auf die wir später zurückkommen.
Der Bereich von 0100h bis CCP-1 ist frei und wird "Transient Program Area", kurz TPA, genannt. Der CCP lädt alle Programme in diesen Bereich und übergibt die Kontrolle dann an Adresse 0100h.
Der obere Adreßbereich wird vom Betriebssystem selbst belegt. Abhängig vom verwendeten System kann die Startadresse des CCP ganz unterschiedliche Werte haben. Entsprechend ändert sich auch die Größe der TPA. Das BDOS schließt an den CCP und das BIOS wiederum an das BDOS an. BDOS und CCP zusammen haben eine Länge von genau 5.5 KBytes (CP/M 2.2). Die Länge des BIOS ist natürlich von Rechner zu Rechner unterschiedlich.
Es gibt allerdings auch CP/M-Versionen, bei denen diese Aufteilung nicht ganz stimmt: Diese Systeme beginnen nicht bei Adresse 0000h, sondern bei einer anderen, BASE genannt. In diesen Systemen muß zu allen genannten Adressen noch der Offset BASE addiert werden. Da solche Systeme aber extrem selten sind, soll dieser Sonderfall hier keine weitere Beachtung finden.
Bemerkenswert ist auch noch, daß CCP, BDOS und BIOS jeweils an einer Seitengrenze beginnen (d.h. an einer Adresse der Form xx00h).

3

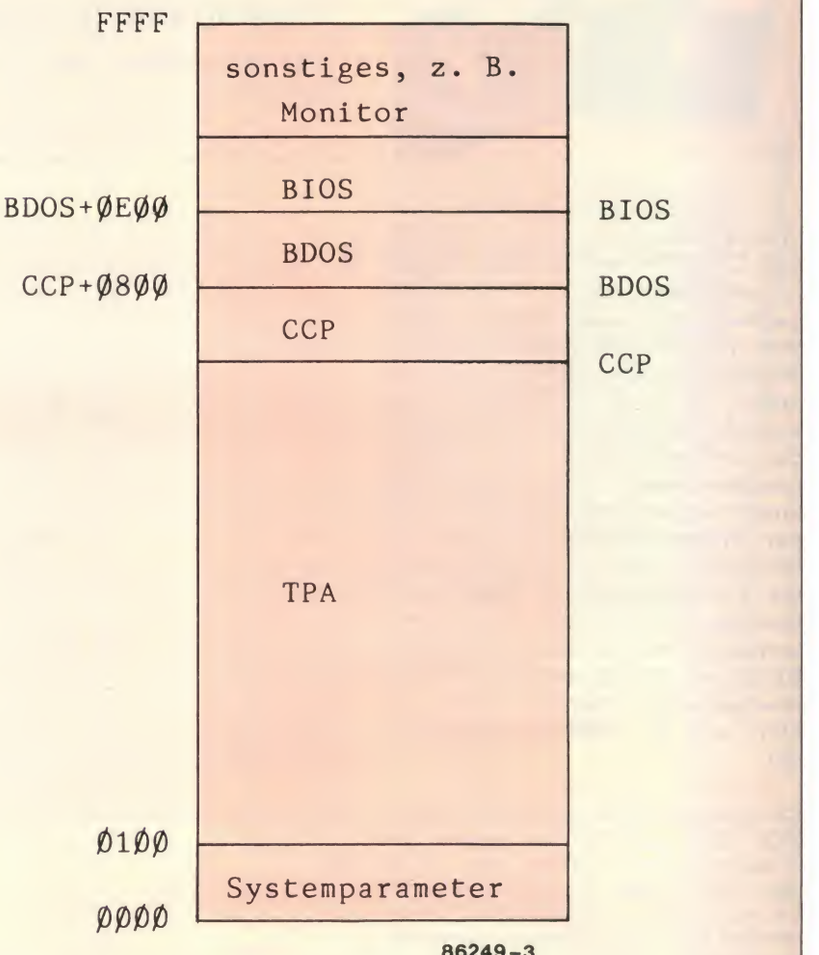


Bild 3. Die Speicheraufteilung eines beliebigen CP/M-Systems. Bei ganz exotischen Implementationen muß jeweils noch ein Offset namens "BASE" zu allen angegebenen Adressen hinzugerechnet werden.

Introduktion

Bevor wir uns nun genauer mit den verschiedenen BDOS-Funktionen beschäftigen, hier noch ein paar ganz allgemeine Regeln zum Schreiben von Assembler-Programmen unter CP/M:

Da der CCP grundsätzlich alle .COM-Files an die Adresse 0100h (Beginn der TPA) lädt und dann dort mit der Ausführung beginnt, muß jedes CP/M-Programm an dieser Adresse einen ausführbaren Befehl enthalten.
Beendet wird ein Programm mit der Anweisung "jmp 0", welche einen Warmstart (wie Control-C) auslöst und sodann den CCP neu startet. Es gibt zwar noch andere Möglichkeiten, die wir später erläutern, doch dies ist die einfachste.
Und nun wollen wir uns endlich den einzelnen BDOS-Funktionen zuwenden. Sie werden alle nach folgendem Schema aufgerufen:

```
mvi c,NUMMER    ; BDOS-Nummer
call 5          ; BDOS-Einsprung
```

Es wird also zuerst das Register C mit der Nummer des gewünschten Aufrufs geladen und dann über die Adresse 5 ins BDOS verzweigt. Alle BDOS-Funktionen erreicht man damit über einen einzigen Einsprung!
An dieser Stelle erkennt man auch schon, daß die absoluten Startadressen der verschiedenen Betriebssystem-Teile (vorerst)

überhaupt nicht wichtig sind, weil über Vektoren zu ihnen verzweigt wird, ähnlich, wie das bei den BIOS-Aufrufen der Fall war. Für den Anwender ist nur die Adresse dieser Vektoren interessant, und die ist in allen CP/M-Systemen identisch!

Low Level BDOS

Da diese die einfacheren sind, beschäftigen wir uns zunächst einmal mit den BDOS-Funktionen, die nichts mit der Verwaltung der Massenspeicher zu tun haben.
Sie betreffen in erster Linie den Datentransport zwischen dem Rechner selbst und der Konsole.
Da gibt es zuerst einmal die beiden noch recht einfachen Funktionen "Konsoleingabe und "Konsolausgabe": Die Funktion 1 liest ein Zeichen von der Konsole ein und liefert es dann im Register A ab. Sie arbeitet mit "Echo", das heißt: Es werden alle darstellbaren ASCII-Zeichen sowie einige Steuerzeichen (z.B. CR, LF, BS) gleich mit an die Konsolen-Ausgabe weitergeleitet. Tabulatoren werden expandiert, die Eingabe von Ctrl-I (Control-I) bewegt also den Cursor an die nächste Tabulatorposition. Ebenso wird auf Ctrl-S sowie Ctrl-P überprüft. Falls nötig, wird als Reaktion darauf die Konsolausgabe bis zur Eingabe des nächsten Zeichens angehalten (Ctrl-S) oder der Drucker zum Protokollieren der Konsolausgabe ein- bzw. ausgeschaltet.
Funktion 2 ist das Pendant zu Funktion 1: Das Zeichen im Register E wird zur Konsole geschickt. Auch hier wird auf Ctrl-P sowie Ctrl-S überprüft. Tabulatoren werden ebenfalls expandiert.
Für den, der es eher spartanisch mag, gibt es die Funktion 6 zur direkten Konsol-Ein/Ausgabe. Will man sie zur Ausgabe benutzen, so lädt man das entsprechende ASCII-Zeichen in das Register E und führt dann ein "CALL BDOS" durch. Bei dieser Form der Ausgabe werden Tabulatoren nicht expandiert und auf Ctrl-P oder Ctrl-S erfolgt keine besondere Reaktion.
Die gleiche Funktion kann aber auch zur Konsoleingabe benutzt werden, indem man das Register E nicht mit einem ASCII-Zeichen, sondern mit dem Wert 0FFh lädt und dann das BDOS aufruft. Ist ein Zeichen vorhanden, so wird dieses im Register A zurückgeliefert. Ist keines vorhanden, so enthält A den Wert 0. Im Gegensatz zur normalen Konsoleingabe wird hier also nicht gewartet, bis wirklich ein Zeichen vorliegt! Bei dieser Form der Eingabe erfolgt kein Echo zur Konsole.
Der Vorteil dieser Funktion gegenüber den beiden zuerst genannten ist, daß sie dem Programmierer mehr Freiheiten läßt. So eignen sich die Funktionen 1 und 2 z.B. denkbar schlecht, um mit ihnen einen Bildschirm-orientierten Editor zu erstellen. Andererseits muß sich der Programmierer diese Freiheit mit einem Mehraufwand an Arbeit erkaufen.

Zeichen an der Kette

Noch komfortabler als die schon besprochenen sind die Funktionen 9 und 10. Hier wird die Ein/Ausgabe nicht mehr Zeichen- sondern Zeichenketten(String)-orientiert durchgeführt. Will man z.B. einen ganzen Satz ausgeben, so braucht man dies nicht mehr Buchstabe für Buchstabe zu erledigen. Man lädt einfach die Adresse, an der der Satz im Speicher abgelegt ist, in das 16-Bit-Register DE und ruft die Funktion 09 auf. Damit das Ende eines Druckstrings erkannt werden kann, muß direkt hinter ihm das Dollar-Zeichen '$' im Speicher stehen. Dieses wird nicht mehr mit zur Konsole übertragen. Leider ergibt sich aus diesem Umstand auch, daß der String selbst dieses Zeichen nicht enthalten darf. Es muß einzeln mit Hilfe der Funktion 2 übergeben werden. Bezüglich der Behandlung der einzelnen Zeichen verhält sich Funktion 9 wie Funktion 2: Tabulatoren werden expandiert und Steuerzeichen ausgewertet.
Kann man Zeichen kettenweise exportieren, so darf man wohl mit Recht erwarten, daß auch der kettenweise Import unterstützt wird. Das erledigt die Funktion 10. Damit läßt sich eine ganze (fix und fertig editierte) Textzeile auf einmal vom Benutzer entgegennehmen. Wiederum im Doppelregister DE wird die Adresse des Puffers angegeben, in den die Zeichen eingelesen werden sollen. Damit dieser nicht während der Eingabe überläuft, muß man ihn mit einer Längenangabe versehen. Diese belegt das erste Byte im Puffer. Man kann sie im Bereich von 1 bis 255 wählen. (Ein "Länge 0" macht ja schließlich auch keinen Sinn!). Übergeben wird die Eingabezeile an das aufrufende Programm, wenn der Benutzer entweder CR oder LF eingegeben hat oder wenn der Puffer voll ist. Auch hier steht man wieder vor dem Problem, das Ende der Zeichenkette erkennen zu müssen. Es wurde in diesem Fall so gelöst, daß der zweite Eintrag im Puffer die wirkliche Länge des Strings enthält und der String selbst erst ab der dritten Adresse im Pufferbereich abgelegt ist.
Das Schöne an dieser Funktion ist, daß sie alle CP/M-spezifischen Steuerzeichen auswertet, z.B. Ctrl-C, Ctrl-H, Ctrl-X, Ctrl-P, Ctrl-S usw. (siehe auch "CP/M in 90 Minuten" in ec 2). Auch ein Echo zur Konsole wird sofort vorgenommen. Diese Funktion stellt den Editor zur Verfügung, mit dessen Hilfe der CCP Ihre Kommandos entgegennimmt!

4

```
0000H - 0002H   Enthält Sprungbefehl zur BIOS-Warmstartroutine
0003H           Intel Standard IOBYTE zum Verteilen der I/O
0004H           Voreingestellte Disk/User Nummer:
                z. B.: 00 bedeutet User 0, Laufwerk A oder
                       2F bedeutet User 2, Laufwerk P
0005H - 0007H   Sprung zum BDOS
                Mit 'LHLD 0006H' erhält man die niedrigste, vom
                CP/M belegte Speicheradresse (wenn man den CCP
                nicht mitrechnet). Diese Grenze dürfen Anwender-
                Programme nicht überschreiten!
0008H - 0027H   Interrupt-Einsprünge 1 bis 5, nicht benutzt
0030H - 0037H   Interrupt-Einsprung 6, von CP/M 2.2 nicht benutzt,
                aber für später reserviert
0038H - 003AH   Restart 7 - wird von Debuggern wie DDT oder SID als
                Einsprung bei der Benutzung von Breakpoints genutzt,
                wird aber sonst nicht verwendet
003BH - 003FH   von CP/M 2.2 nicht genutzt, für später reserviert
0040H - 004FH   16 Byte Arbeitsbereich für ein CBIOS, wird aber von
                der Handelsversion des CP/M nicht benutzt
0050H - 005BH   von CP/M 2.2 nicht benutzt, für später reserviert
005CH - 007CH   Adresse des voreingestellten File Control Blocks
                (FCB). Hier legt z. B. der CCP seinen ersten FCB ab
007DH - 007FH   Optionale FCB-Erweiterung für wahlfreien Zugriff
0080H - 00FFH   voreingestellter Diskettenpuffer für einen 128-Byte
                Sektor. Hier wird auch die Kommandozeile abgelegt,
                wenn ein Anwender-Programm gestartet wird.
```

Bild 4. So werden die ersten 256 Bytes (die sogenannte Page Zero) vom Betriebssystem belegt. Viel Platz bleibt da nicht mehr...

Und sonst noch?

An diesem Punkt angelangt, sollte man mal einen Blick auf Bild 5 werfen, wo ein kleines Programm den Umgang mit den bisher vorgestellten Funktionen erläutert: Nach so viel grauer Theorie mal ein bischen (rosa?) Praxis! Sie hilft sicher, das bisher Gesagte besser zu verdauen.
Das Listing zeigt auch gleich eine weitere Möglichkeit, Programme zu beenden. So findet man hinter dem Label "exit" nicht, wie weiter oben empfohlen, die Anweisung "jmp 0", sondern einen Aufruf der BDOS-Funktion 0, die exakt das Gleiche bewirkt.
Wer fleißig mitgezählt hat, der hat auch bemerkt, daß wir von den ersten 13 BDOS-Funktionen gerade 6 kennengelernt haben: Da waren's nur noch 7. Mit der Funktion 11 kann man den Konsolenstatus prüfen, ohne (wie bei Nr. 6) bei Vorhandensein gleich ein Zeichen einle-

Bild 5. Dieses kleine Demonstrations-Programm zeigt den Umgang mit den im Text besprochenen Funktionen.

5

```
                        ;*****************************************
                        ;*                                       *
                        ;*  Demo-Programm zur Zeichen-Ein/Ausgabe  *
                        ;*                                       *
                        ;*****************************************

0100                    org 0100h       ; Programmstart immer hier
                    ;
                    ;
0005 =              bdos    equ 5           ; Eintrittsadresse des BDOS
                    ;
000D =              cr      equ 0dh
000A =              lf      equ 0ah
0024 =              eot     equ '$'
                    ;
                    ;
                    start:
0100 CD8E01             call clrscrn    ; Bildschirm löschen
0103 11B401             lxi d,string$1  ; Adresse des Strings holen
0106 CDA801             call outline    ; String ausgeben
                    ;
0109 0627               mvi b,len$1+2   ; Länge der ausgegebenen Zeile
                    ;
                    underlp:
010B C5                   push b        ; Zähler retten
010C 1E3D                 mvi e,'='     ; Zeichen zum Unterstreichen
010E CD7D01               call conout   ; Ausgabe des Zeichens
0111 C1                   pop b         ; Zähler wiederherstellen
0112 05                   dcr b         ; und erniedrigen
0113 C20B01             jnz underlp     ; Schleife bis Ende
                                        ; Zeile fertig unterstrichen
0116 CD8301             call crlf       ; und hochschieben das Ganze
0119 CD8301             call crlf
                    ;
011C 11DD01             lxi d,string$2  ; Adresse 2er String
011F CDA801             call outline    ; ausgeben
                    ;
                    ascii:
0122 CD7701               call conin    ; Zeichen einlesen
0125 FE20                 cpi ' '       ; darstellbares Zeichen?
0127 DA2201               jc ascii      ; falls nicht, dann neu
012A FE7F                 cpi 07fh
012C D22201             jnc ascii
012F F5                 push a          ; Zeichen retten
0130 CD8301             call crlf       ; Zeilenvorschub
0133 F1                 pop a           ; Zeichen wiederherstellen
0134 5F                 mov e,a         ; Zeichen in E
                    ;
                    outlp:
0135 D5                   push d        ; Zeichen retten
0136 CDA201               call dirout   ; direkt ausgeben
0139 CD9A01               call dirin    ; direkte Eingabe
013C D1                   pop d         ; Ausgabe-Zeichen holen
013D FE03                 cpi 3         ; Ctrl-C?
013F C23501             jnz outlp       ; Weiter, falls kein Zeichen
                                        ; oder nicht Ctrl-C
0142 CD8301             call crlf
0145 CD8301             call crlf
0148 115002             lxi d,string$3
014B CDA801             call outline
014E 117C02             lxi d,linbuf    ; Adresse Zeilenpuffer
0151 CDAE01             call inline     ; Zeile entgegennehmen
                    ;
                    :
```

```
0154 CD8301             call crlf
0157 3A7D02             lda linbuf+1    ; Zeilenlänge holen
015A B7                 ora a           ; Statusbits setzen
015B CA7201             jz exit         ; Schluß, wenn Länge=0
015E 47                 mov b,a         ; Zähler in b
015F 217E02             lxi h,linbuf+2  ; Zeiger auf Zeichen
                ;
                dnlp:
0162 5E                   mov e,m       ; Zeichen holen
0163 C5                   push b        ; Zähler retten
0164 E5                   push h        ; Zeiger retten
0165 CD7D01               call conout   ; Zeichen ausgeben
0168 CD8301               call crlf     ; Zeilenvorschub
016B E1                   pop h         ; Zeiger wiederherstellen
016C C1                   pop b         ; Zähler wiederherstellen
016D 23                   inx h         ; Zeiger erhöhen
016E 05                   dcr b         ; Zähler erniedrigen
016F C26201             jnz dnlp        ; Schleife bis Ende
                ;
                ;
                exit:
0172 0E00               mvi c,0         ; Ende des Programms,
0174 CD0500             call bdos       ; Rückkehr zum CCP
                ;
                ;
                ; *** Unterprogrammsammlung
                ;
                ;
                conin:                  ; Konsoleingabe mit Echo
0177 0E01               mvi c,1         ; BDOS-Nr ist 1
0179 CD0500             call bdos       ; ausführen, Zeichen ist in A
017C C9                 ret             ; und Ende
                ;
                ;
                conout:                 ; Konsolausgabe mit Auswertung
                                        ; der Steuerzeichen, Zeichen
                                        ; befindet sich im Register E
017D 0E02               mvi c,2         ; BDOS-Nr ist 2
017F CD0500             call bdos
0182 C9                 ret
                ;
                ;
                crlf:                   ; Wagenrücklauf durchführen
0183 1E0D               mvi e,cr        ; CR ausgeben
0185 CD7D01             call conout
0188 1E0A               mvi e,lf        ; LF hinterher
018A CD7D01             call conout
018D C9                 ret
                ;
                ;
                clrscrn:                ; Bildschirm löschen
018E 0618               mvi b,24        ; Anzahl der darstellbaren Zeilen
                ;
                clrlp:
0190 C5                 push b          ; Schleifenzähler retten
0191 CD8301             call crlf       ; Bildschirm eine Zeile nach oben
0194 C1                 pop b           ; Schleifenzähler wiederherstellen
0195 05                 dcr b           ;        "        erniedrigen
0196 C29001             jnz clrlp       ; und wiederholen bis Ende
0199 C9                 ret
                ;
                ;
                dirin:                  ; Direkte Konsoleingabe
```

```
019A 0E06               mvi c,6
019C 1EFF               mvi e,0ffh
019E CD0500             call bdos       ; A enthält Zeichen oder Status
01A1 C9                 ret             ; 0 in A bedeutet: Kein Zeichen
                ;
                ;
                dirout:                 ; Direkte Konsolausgabe
                                        ; ASCII-Zeichen zur Ausgabe
                                        ; befindet sich im Register E
01A2 0E06               mvi c,6
01A4 CD0500             call bdos
01A7 C9                 ret
                ;
                ;
                outline:                ; Zeichenketten-Ausgabe
                                        ; DE enthält Adresse des Strings
                                        ; '$' schließt String ab
01A8 0E09               mvi c,9
01AA CD0500             call bdos
01AD C9                 ret
                ;
                ;
                inline:                 ; Eine Zeile von der Konsole
                                        ; entgegennehmen:
                                        ; Pufferadresse in DE, erster
                                        ; Wert in Puffer ist maximale
                                        ; Länge
01AE 0E0A               mvi c,10
01B0 CD0500             call bdos
01B3 C9                 ret
                ;
                ;
                ; *** Strings und Puffer
                ;
                ;
                string$1:
01B4 204465D6F          db ' Demo-Programm zur Zeichen-Ein/Ausgabe'
01DA 0D0A               db cr,lf        ; Zeilenvorschub
01DC 24                 db eot          ; Ende des Strings
                ;
0025 =          len$1   equ 37          ; Anzahl der ASCII-Zeichen
                                        ; in string$1
                ;
                ;
                string$2:
01DD 4269747465         db 'Bitte ein beliebiges Zeichen eingeben! Das'
0207 0D0A               db cr,lf
0209 7769726420         db 'wird dann so lange ausgegeben,  bis Ctrl-C'
0233 0D0A               db cr,lf
0235 65696E6765         db 'eingegeben wird. Zeichen: '
024F 24                 db eot
                ;
                ;
                string$3:
0250 4A65747A74         db 'Jetzt bitte eine ganze Zeile eingeben!'
0276 0D0A               db cr,lf
0278 203E20             db ' > '
027B 24                 db eot
                linbuf:
027C 50                 db 80           ; maximale Länge der Eingabezeile
027D 00                 db 0            ; Platz für wirkliche Länge
027E                    ds 80           ; Platz für den String
                ;
                ;
02CE                    end
```

6

Nummer	Parameter	Resultat	Bemerkung
0	-	-	Warmstart des Betriebssystems
1	-	A	Konsoleinsgabe, Zeichen in A
2	E	-	Konsolausgabe, Zeichen in E
3	-	A	Reader-Eingabe, Zeichen in A
4	E	-	Punch-Ausgabe, Zeichen in E
5	E	-	List-Ausgabe, Zeichen in E
6	E=0ffh	A	Direkte Konsol-Eingabe, Zeichen oder Status in A
6	E	-	Direkte Konsol-Ausgabe, Zeichen in E, E<>0ffh
7	-	A	IOBYTE nach A holen
8	E	-	IOBYTE setzen, neuer Wert in E
9	DE	-	String drucken: Anfangsadresse in DE, abgeschlossen mit '$'
10	DE	-	String über Editor einlesen, Puffer-Anfangsadresse in DE, erstes Byte in Puffer enthält die maximale, zweites die wirkliche Stringlänge
11	-	A	Konsolenstatus in A, 0 bedeutet "kein Zeichen vorhanden"
12	-	HL	Versionsnummer holen: H=1: MP/M, H=0: CP/M, L enthält Versionsnummer in BCD-Kodierung

Bild 6 gibt eine Zusammenstellung aller "Non-Disk"-BDOS-Funktionen. Genauere Informationen entnimmt man dem Text.

sen zu müssen. Aussagekräftig ist der Inhalt des Registers A. Ist er gleich 0, so liegt kein Zeichen an, andernfalls doch.
Mit der Funktion 12 kann man zwischen den verschiedenen CP/M-Versionen unterscheiden. Sie liefert in HL die Versionsnummer. Bei CP/M 2.2 ist z.B. H=0 und L=22h. Bei MP/M x.y dagegen H=1, L=xyh.

Distribution

Nun kann man unter CP/M nicht nur mit der "normalen" Konsole als Ein/Ausgabe-Gerät arbeiten, sondern z.B. auch mit einem Drucker, Plotter, Grafik-Terminal usw. Schauen wir uns doch mal an, wie man das arrangiert.
Da gibt es zunächst einmal die Funktionen 4 (Ausgabe auf Lochstreifenstanzer) und 5 (Ausgabe auf Drucker). Sie verhalten sich logisch genau wie die Funktion 2, nur daß die Ausgabe halt auf andere Geräte umgelenkt wird. Damit Sie Ihre Lochstreifen auch wieder einlesen können, gibt es die Funktion 3, die ein Zeichen vom Lochstreifenleser einliest. Hierbei erfolgt übrigens kein Echo!
Und wenn Sie nun aber ganz und gar keinen Lochstreifen-Wirrwarr auf Ihrem Schreibtisch entdecken können? Nicht traurig werden! Diese Bezeichnungsweise hat sich nun mal bei CP/M so eingebürgert, auch wenn sie sehr anachronistisch anmutet. Das Betriebssystem hat ja auch schon ein paar Jahre auf dem Buckel! Es handelt sich hierbei einfach um irgendwelche sonstigen Geräte, die Sie über diese Software-Schnittstelle bedienen können!
Mit den bis jetzt zur Verfügung stehenden Hilfsmitteln kann man schon vier verschiedene Geräte bedienen. Diese nennt man auch die "physikalischen" Geräte. Mit Hilfe des "IOBYTE" ist es nun zusätzlich möglich, jedem dieser physikalischenbis zu vier "logische" Geräte zu zuordnen. Wie das im Einzelnen geht, darauf wollen wir hier nicht eingehen. Es soll an dieser Stelle nur erwähnt werden, daß die BDOS-Funktionen 7 und 8 für die Änderung bzw. für die Abfrage des IOBYTEs zuständig sind: 7 holt es in das Register A, mit 8 wird es auf den Wert im Register E gesetzt.
Damit wären alle "Non-Disk"-BDOS-Funktionen besprochen. Weiter geht's im nächsten Heft. Dort werden wir uns eingehend mit dem "File Control Block" (FCB) beschäftigen, der für die Parameterübergabe zwischen Anwender-Programmen und Disk-orientierten BDOS-Funktionen eine Schlüsselrolle spielt. Außerdem geht es um das Öffnen und Schließen von Dateien, sequentielles sowie wahlfreies Schreiben und Lesen, das Kreieren (Einrichten), Umbenennen und Löchen von Dateien und vieles anderes mehr.

CP/M auf dem EC-65 (Teil 2)

Die Systemsoftware zur Coprozessor - Karte für den EC-65

Frank Schmidt

Ohne Disketten-Software "geht nichts" beim EC-65! Vor allem dann, wenn der in Kontakt mit der in diesem Heft vorgestellten Z80-Karte treten will. Das ist aber noch lange kein Grund zum Verzagen: Denn nach guter alter Elektor-Manier gibt's die mal wieder vollkommen kostenlos beim Software-Verteiler. Daß dabei nicht mit Komfort gegeizt wird, sei hier gleich vorweggenommen. Wer Genaueres wissen will, der sollte mal einen Blick in diesen Artikel werfen!

Bei den meisten Computern, die auch ohne Diskettenlaufwerk arbeiten können und nicht gleich BASIC als Betriebssystem verwenden, findet man sich nach dem Einschalten des Rechners in einem Monitor-Programm wieder, von dem aus man weitere Programme laden und damit das System "aufblasen" kann. Das war schon bei unserem "guten alten" Junior so, und auch der EC-65, der EC-68 und das Big Board II bauen auf diesem Prinzip auf. Im EPROM der Z80-Karte steckt zwar auch ein Monitor-Programm, trotzdem kann man ohne Disketten-Software damit nicht arbeiten. Das hat natürlich seinen Grund: Um nämlich das System überhaupt zum Laufen zu bringen, müssen sowohl die 6502- als auch die Z80-CPU mit Software versorgt werden. Da aber der EC-65 nach dem Einschalten von einer eingesteckten Z80-Karte "gar nichts weiß", muß er sowieso per Programm von Diskette schlau gemacht werden! Da kann man dann auch gleich ein komplettes Betriebssystem laden. Zwischen diesem Operating-System und dem Monitor-Programm kann man dann beliebig hin- und herschalten. Näheres dazu im Artikel "EC-65: CP/M-Software" in diesem Heft.

An welches Betriebssystem denkt man bei einer Z80-CPU wohl zuerst? An CP/M doch natürlich.

Nur, daß das noch teurer wäre als die komplette Z80-Hardware (das gilt für CP/M 2.2, für CP/M 3.x muß man sogar ein Vielfaches davon hinblättern!). Wer ec regelmäßig gelesen hat, der weiß vielleicht, daß man so viel Geld nicht unbedingt ausgeben muß: Haben doch findige Computer-Freaks in den USA ein Betriebssystem ausgetüftelt, das nahezu 100% kompatibel zu CP/M 2.2 ist und außerdem viel, viel komfortabler! Und das ist noch nicht alles: Das Wunderwerk namens ZCPR2 ist "Public Domain"! Das bedeutet, daß Sie mit dieser Programm-Sammlung vollkommen kostenlos arbeiten können! Dabei gilt allerdings die Einschränkung, daß Sie sie privat und nicht zu geschäftlichen Zwecken nutzen.

Dieses Betriebssystem haben wir schon auf dem Big Board II und jetzt auch auf dem EC-65/Z80 implementiert, und es bietet die Gewähr dafür, daß Sie Zugriff haben auf die riesige Menge von Software, die für CP/M geschrieben wurde!

Booten

Die ZCPR2-Systemdiskette ist beim Software-Verteiler sowohl im 80-Track- als auch im 40-Track-Format erhältlich (siehe auch Artikel "EC-65: CP/M-Software" in diesem Heft). Nachdem Sie sie bekommen haben, sollten Sie davon zuerst einmal eine Sicherheitskopie erstellen. Und dann ab in den Safe mit dem Original! Die Kopien kann man übrigens wie gewohnt mit dem Menüpunkt 8 (Single or dual disk drive copier) von BEXEC* machen. Also mit Hilfe des OHIO-Betriebssystems. Wer das nicht besitzt und mit dem EC-65 nur als CP/M-Maschine arbeiten will (auch das ist möglich!), der muß sich mit dem Kopieren noch etwas gedulden und weiter lesen.

Gebootet wird die Diskette wie eine normale OSI-Scheibe: Rechner anschalten (bzw. RESET betätigen), Diskette in Laufwerk A einlegen und ".b" eintippen. Jetzt sollte auf dem Bildschirm eine Meldung wie in Bild 1 erfolgen. Ist das wider alles Erwarten nicht der Fall, obwohl die Z80-Karte fehlerfrei funktioniert (Testprogramm!) und auch Ihr EC-65 bis jetzt noch keinen Anlass zur Beschwerde gegeben hat, so kann das an einer fehlerhaften Interrupt-Verarbeitung in Ihrem System liegen. Da nämlich die "normale" Software des EC-65 vollkommen ohne Interrupts läuft (abgesehen vom Tracer, und der arbeitet mit dem nicht maskierbaren Interrupt NMI), ist es durchaus möglich, daß auf der IRQ-Leitung Ihres Computers "regelwidrige" Dinge ablaufen, ohne daß dies einen Einfluß auf die Zuverlässigkeit des Systems hat. Die CP/M-Software macht dagegen fleißig Gebrauch von den Interrupt-Möglichkeiten des EC-65! Hier macht sich ein solcher, bisher unentdeckter Fehler schlagartig bemerkbar.

Sollte also Ihr System nicht auf Anhieb korrekt arbeiten, sondern "hängen", so überprüfen Sie nach dem Booten zuerst einmal die IRQ-Leitung. Sie liegt auf dem Bus auf Anschluß 12a. Wenn diese dauernd aktiv (d. h. LOW) ist, so liegt mit großer Wahrscheinlichkeit der hier beschriebene Fehler vor. Die Software gestattet es nur der VIA A (im Schaltbild (ec 1, S. 22f) IC 2), Interrupts auszulösen. Wahrscheinlich existiert in einem nicht korrekt arbeitenden System noch eine weitere Interrupt-Quelle, die vollkommen unmotiviert "ihren Senf dazugibt".

Suchen und eliminieren Sie die! Im einfachsten Fall geht das schon mit der Methode "Raus aus dem Sockel". Wenn das System dann nacher läuft, können Sie sich immer noch überlegen, wie Sie den Übeltäter regelgerecht "maultot" machen, ohne ihn gleich ganz entfernen zu müssen.

Hängt Ihr Rechner, Sie können aber trotzdem keine zusätzliche Interruptquelle finden, so geht es Ihnen vieleicht so, wie uns mit einem EC-65 in der Redaktion: Da lag der Fehler im Monitor-EPROM. Das war zwar vorschriftsmäßig programmiert, jedoch nur bis auf das letzte (!) Byte. Das ist "leer" geblieben. Etwas, was beim Programmieren leicht passieren kann ("muß man denn nun die letzte Adresse oder die letzte Adresse +1 eingeben?"). Leider liegt jedoch genau an dieser Stelle der Vektor, der letztendlich zur ISR (Interrupt Service Routine) führt. Die hat unser EC-65 deshalb erst gar nicht gefunden! Gleich beim ersten IRQ hat er sich verabschiedet!

1

```
   == EC 65  CP/M system ==

6502 speed: 2 MHz

CBIOS65 vers 3.3
CBIOS80 vers 3.4
  (C) Frank Schmidt

59k ZCPR2.3
  (C) Richard Conn


A>
```

Bild 1. So meldet sich der Rechner, wenn Sie die ZCPR2-Diskette booten. Die Taktfrequenz der 6502-CPU wird dabei automatisch festgestellt und das Timing für die Disketten-Routinen entsprechend angepaßt. Sie benötigen also für zwei verschiedene Taktfrequenzen nur eine einzige Betriebssystem-Version!

Taktisch

Wenn Sie geglaubt haben, die Meldung "6502 speed: 2 MHz" (Bild 1) sei nur schmückendes Beiwerk, so haben Sie sich getäuscht (dazu wäre auch der Speicherplatz viel zu schade gewesen). Vielmehr ist es so, daß während des Boot-Vorganges die Taktfrequenz des Prozessors "gemessen" wird. Abhängig davon erscheint die Meldung "1 MHz", "2 MHz" oder "No standard 6502 speed!", wenn keine der beiden Taktfrequenzen zutrifft. Der tiefere Sinn des Ganzen wird Ihnen vielleicht klar, wenn Sie Ihren EC-65 auf

die höhere Taktfrequenz umgerüstet haben: Bei der Bedienung der Diskettenlaufwerke müssen bestimmte Zeitabläufe eingehalten werden. Dazu dienen programmierte Verzögerungsschleifen. Erhöht man die Taktfrequenz, so werden diese Schleifen schneller durchlaufen und das Resultat ist ein "Kuddelmuddel" im Floppy-Timing. Langer Rede kurzer Sinn: Will man mit zwei verschiedenen Taktfrequenzen arbeiten, so benötigt man dazu zwei leicht verschiedene Versionen des Betriebssystems!

Nicht so bei unserer CP/M-Implementation: Der CPU-Takt wird automatisch erkannt und das Timing entsprechend angepaßt!

Diskettenwechsel

Bevor Sie sich nun auf das neue Betriebssystem stürzen, sollten Sie wenigstens noch diesen Absatz lesen. Das kann Ihnen viel Ärger ersparen!
Wie alle Dinge hat leider auch unser CP/M-BIOS seine Schattenseiten. In unserem Fall betrifft das den Diskettenwechsel. Ist der bei CP/M-Systemen im allgemeinen schon problematisch, bei uns ist er sogar gefährlich! Allerdings nur, wenn man nicht die folgende kleine Regel beachtet:

Vor und nach jedem Diskettenwechsel ist ein Warmstart des Systems (Control-C) durchzuführen!

(Das gilt nicht, wenn Sie von einem Programm ausdrücklich zum Diskettenwechsel aufgefordert werden!) Ein Nichtbeachten dieser Regel kann zum Todesstoß für Ihre Software werden! Der Grund dafür soll hier nicht weiter erläutert werden (das ist u. a. Inhalt von Teil 3 der Beschreibung des Z80-Systems). Nur so viel: Er liegt im verwendeten Blocking/Deblocking-Allgorithmus, der nämlich nicht den Empfehlungen von Digital Research folgt, dafür aber die Anzahl der Diskettenzugriffe erheblich vermindert und deshalb einen Geschwindigkeitsgewinn bedeutet. Manchmal muß man halt Kompromisse eingehen...

Disketten formatieren

Konnten Sie das Kopieren von CP/M-Disketten ohne Schwierigkeiten mit Hilfe des OHIO-DOS erledigen, so ist dies beim Initialisieren leider nicht mehr möglich. Das liegt daran, daß die vom OS65D vorbereiteten Disketten zwar beschreibbar, aber noch nicht lesbar sind (sie können das gern einmal ausprobieren). "INIDSK.COM" ist die Utility, die hier weiter hilft. Bild 2 zeigt, wie's geht: "INIDSK" eintippen, Laufwerksnummer angeben und dann die Anzahl der zu initialisierenden Tracks mitteilen. Vorerst wird das immer Auswahlpunkt 3 (80 Tracks) oder 1 (40 Tracks) sein, später kann sich das ändern (wenn Sie nämlich auch andere Diskettenformate bearbeiten können...). Aber Vorsicht: Im gleichen Moment, in dem Sie die Ziffer eintippen, geht's auch schon los mit dem Initialisieren! Also unbedingt folgende Vorgehensweise angewöhnen: Zuerst INIDSK aufrufen. Dann leere Diskette in das gewünschte Laufwerk einlegen (hier kann man einen Diskettenwechsel ungestraft vornehmen). Laufwerksnummer eingeben. Haben Sie sich vertippt, so können Sie nach Eingabe von <CR> neu wählen. Ist alles in Ordnung, tippen Sie auch noch die 3 ein. Dann geschieht folgendes:
Die Nummer des Tracks, der gerade bearbeitet wird, erscheint auf dem Bildschirm. Kann der aus irgendeinem Grunde nicht initialisert werden, so erscheint dahinter die Meldung "Cannot init track" und in der folgenden Zeile wird die nächste Track-Nummer ausgegeben. Ist alles glatt gegangen, so wird einfach die alte Nummer von der neuen überschrieben. Ist die Diskette fertig vorbereitet, so gibt das Programm eine entsprechende Meldung aus und startet sich neu.
Abgebrochen werden kann es mit Control-C oder einfach mit <CR>.

2

```
A>inidsk


Disk Initialisation Utility
Created 02-86 by F. Schmidt

Init disk on which drive (A..D) ? c
Init how many tracks ?
35 (0), 40 (1), 77 (2), 80 (3) or 160 (4)
Enter number (0..4): 3

The whole disk is now initialised

Init disk on which drive (A..D) ? <CR>

A>
```

3

```
A>sysgen


CP/M - SYSGEN for JUNIOR
Version 1.1  -  05-01-86
Copyright (C) 1986 by Frank Schmidt


Enter source drive name (or RETURN to skip) a
Source on drive A, then press RETURN <CR>

Enter destination drive name (or RETURN to skip) c
Destination on drive C, then press RETURN <CR>


 - System generated -

Enter destination drive name (or RETURN to skip) <CR>

A>
```

Bild 2. So einfach kann man Disketten initialisieren (alle Eingaben sind unterstrichen). Aber Vorsicht: sowieSie die Ziffer für die Anzahl der zu initialisierenden Tracks eingeben, geht's auch schon los. Also unbedingt vorher die leere Diskette einlegen!

Bild 3. So kopiert man die Systemspuren, in denen das Betriebssystem "versteckt" ist. Wenn man die Disketten nach dem Lesevorgang wechselt, kann man das auch mit nur einem einzigen Laufwerk erledigen.

Systemdisketten-Datendisketten

Das Betriebssystem selbst ist bei CP/M auf den äußeren Tracks der Disketten untergebracht (bei uns belegt es fünf Spuren). Ist es dort auch wirklich vorhanden, so sprechen wir von einer "Systemdiskette", ganz gleich was auf der Diskette sonst noch zu finden ist. Im anderen Fall handelt es sich um eine reine "Datendiskette".
Bei jedem Warmstart wird nun ein Teil des CP/M (BDOS und CCP) neu geladen. Bei einem "normalen" CP/M-System muß deshalb im Laufwerk A **immer** eine Systemdiskette liegen. Bei uns ist das wieder einmal anders: eine Systemdiskette muß nur beim Booten in Laufwerk A liegen. Während dieses Vorganges wird das Betriebssystem dann ein einziges Mal geladen und im 6502-Memory festgehalten. Bei jedem darauf folgenden Warm-

start wird es einfach vom 6502- in den Z80-Speicher kopiert! Das spart Zeit und erleichtert den Umgang mit den Disketten.
Trotzdem wird man nicht umhin können, auch mal eine neue Systemdiskette zu erstellen. Dazu gibt es die Utility "SYSGEN.COM". Sie wurde zwar für den EC-65/Z80 vollkommen neu geschrieben, arbeitet jedoch genauso wie die Original-Version von Digital Research. Bild 3 zeigt an einem einfachen, aber sehr praxisnahen Beispiel den Umgang damit. Man kann übrigens unter anderem auch die Systemspuren von Laufwerk A nach Laufwerk A kopieren, indem man zwischendurch die Diskette wechselt. Wieder einer der seltenen Fälle, in denen man das "einfach so" ungestraft tun darf.

Möchten Sie nicht die Systemspuren, sondern ganz normale Dateien kopieren, so steht Ihnen dazu die ZCPR2-Utility "MCOPY.COM" zur Verfügung. Wollten wir die in allen Details besprechen, so würde das rasch unseren Rahmen sprengen. Für's erste können Sie damit genauso umgehen, wie mit der CP/M-Utility "PIP.COM" (siehe Artikel "CP/M in 90 Minuten" in ec 2). Für's zweite können Sie mit dem Befehl "MCOPY //" Hilfsinformationen (in Englisch) über das Programm abrufen.
Das gilt generell für alle ZCPR2-Utilities, sofern sie überhaupt Hilfsinformationen bereitstellen. Das Kommando "DIFF //" wird Ihnen also beispielsweise Informationen über das Programm "DIFF.COM" beschaffen.
Und nun zur dritten Möglichkeit, an ZCPR2-Dokumentation heranzukommen: Besorgen Sie sich das "Software-Handbuch" (in Englisch)zum Big Board II! Das können Sie kaufen oder selbst kopieren (wenn Sie jemanden kennen, der es besitzt). Beachten Sie bei Ihrer Entscheidung aber eines: Es handelt sich dabei um ein Werk von etwa 300 Seiten... (Anm. d. Redaktion: Wir werden uns um eine Bezugsquelle bemühen und sie dann u.a. den Softwareverteilern mitteilen).

Es ist übrigens auch allen sonstigen CP/M-Anwendern ein Blick in dieses Buch zu empfehlen! Es beschreibt ausführlich die Möglichkeiten des ZCPR2-Systems und dessen Installation. Auch der Umgang mit diesem schönen Software-Paket wird darin ausführlich erläutert. Voraussetzung für eine eigene Installation sind allerdings ein Z80-Prozessor sowie fundierte Kenntnisse in Maschinensprache. Auch einen Macro-Assembler sollte man besitzen. Die gesamte sonst noch benötigte Software sowie das angeführten Handbuch sind Public Domain und auch auf Diskette von der SIG/M-Usergroup sowie einigen Software-Vertreibern zu erhalten.

Schnelle Scheiben

Nun zu einem weiteren hübschen Ausstattungsmerkmal unserer CP/M-Implementation: Der RAM-Floppy.
CP/M ist standardmäßig nur zur Verwaltung von maximal 64 KByte Halbleiter-Speicher in der Lage. Die im Hardwareteil dieses Artikels vorgestellte CPU-Karte kann aber bis zu einem Megabyte (!) adressieren. In der Grundversion bringt sie es sogar schon auf stattliche 256 KByte.
Es empfiehlt sich dringend, die Grundversion mit 256er DRAMs zu bestücken. Dann kann man nämlich die "überflüssigen" drei Speicherbänke als RAM-Floppy benutzen. Sie werden in diesem Fall vom Betriebssystem aus wie Diskettenlaufwerke verwaltet. Für den Anwender ergeben sich allerdings zwei (gravierende) Unterschiede zu diesen:
— Nach dem Aus- und Einschalten des Rechners ist die "Diskette" leer
— Die Schreib/Lese Zugriffe sind erheblich schneller als bei der magnetischen Scheibe
Natürlich ist es gerade der zweite Punkt, der für uns von Interesse ist. Doch dazu später mehr.
Die Software ist so aufgebaut, daß sie automatisch erkennt, wie viel Speicher zur Verfügung steht und alles, was über die 64K-Grenze hinausgeht, als RAM-Floppy verwaltet. Bild 4 zeigt den "STAT DSK:"-Ausdruck des Laufwerks E - der RAM-Floppy - bei einer 256K-Speicher-Konfiguration.
Hat man nur 64K zur Verfügung, so erscheint beim Ansprechen des Laufwerks E eine BDOS-Fehlermeldung.
Aber selbst dann kann man noch zu einer RAM-Floppy kommen. Gibt es doch auch im Bereich der 6502-CPU Speicher (immerhin 40 KBytes), der bis jetzt noch nicht genutzt ist! Die Software zur Verwaltung dieses Reservoirs als flinke Floppy ist schon auf der System-Diskette vorhanden. Sie muß nur noch aktiviert werden. Normalerweise wird nämlich dieser Speicherbereich als Printer-Spooler verwaltet. Der Printer-Spooler muß deshalb vorher "lahmgelegt" werden. Wie man das tut, werden wir an anderer Stelle erläutern. Man kann also nur entweder mit der (kleinen) RAM-Floppy oder dem Printer-Spooler arbeiten.
Wozu benötigt man denn nun so eine elektronische Diskette?
Die meisten unter CP/M arbeitenden Compiler, Assembler und sonstige Übersetzungs-Programme arbeiten nach folgender Methode: Sie lesen ein File ein, übersetzen es und liefern als Output ein zweites File (oder gar mehrere). Dabei werden die Files aber nicht "auf einen Rutsch" gelesen oder geschrieben, sondern schön so, wie's kommt: einen Sektor File A lesen, einen File B schreiben usw. (CP/M-Sektoren sind immer 128 Bytes groß!). Da die verschiedenen Files an ganz verschiedenen Orten auf der Diskette stehen können, muß der Schreib/Lesekopf des Laufwerks ständig hin- und her fahren. Das kostet - im Verhältnis zum eigentlichen Disk-Zugriff - sehr viel Zeit und fördert außerdem weder die Lebenszeit der Disketten noch die der Laufwerke.
Einer RAM-Floppy dagegen ist esganz egal, von wo sie die Daten liefern muß. Bei ihr gibt es kein langsames (weil mechanisches) Hin- und Herfahren - elektronisch geht's halt schneller!
Die schnelle Scheibe sollte man deshalb in erster Linie dazu benutzen, Programme dort übersetzen bzw. anderweitig bearbeiten zu lassen.

4

```
A>stat e:dsk:

     E: Drive Characteristics
  1512: 128 Byte Record Capacity
   189: Kilobyte Drive  Capacity
    64: 32  Byte Directory Entries
    64: Checked  Directory Entries
   128: Records/ Extent
     8: Records/ Block
    24: Sectors/ Track
     0: Reserved Tracks

A>
```

Bild 4. Als Laufwerk E ist eine RAM-Floppy implementiert, die das Z80-Memory benutzt, das über die 64K-Grenze hinausgeht. Der Ausdruck zeigt die Diskettenparameter, die sich bei einer 256 KByte-Konfiguration ergeben. Dabei erstellt die Software selber diese Parameter in Abhängigkeit der zur Verfügung stehenden Speicherbänke. Sind nur 64 KByte auf der Platine vorhanden, so erscheint eine BDOS-Fehlermeldung beim Versuch, Laufwerk E anzusprechen.

Auch ist es zu empfehlen, dort oft benutzte Programme abzulegen, um sie flink laden zu können. Und noch ein drittes Anwendungsgebiet gibt es: Wenn Sie Dateien von einer Diskette auf eine andere mit nur einem einzigen Laufwerk kopieren wollen, so stehen Sie bei CP/M ziemlich dumm da! Das ist nämlich einfach nicht vorgesehen! Die Lösung: Dateien erst in RAM-Floppy kopieren, Ctrl-C, Disketten wechseln, Ctrl-C, Dateien von RAM-Floppy auf zweite Diskette kopieren - fertig.
Um mal ein paar Zahlen zu nennen: Das Assemblieren eines Programmes, daß auf der konventionellen Scheibe etwa 42 Sekunden in Anspruch nahm, wurde in der "kleinen" RAM-Floppy in 11 und in der "großen" in nur neun Sekunden abgetan!

Ein/Ausgabe

Wir haben es eben schon angesprochen: In der Software zum EC-65/Z80 steckt auch ein Programm, daß die 40 KByte, die sonst nicht verwendet werden - sofern man nämlich auf die Mini-RAM-Disk verzichtet - als Printer-Spooler verwaltet (was ein Spooler ist, das können Sie übrigens in ec 4 nachlesen). Der arbeitet aber ausschließlich mit der Centronics-Schnittstelle zusammen. Die wird auch von CP/M aus als LST:-Device angesprochen. Die V24-Schnittstelle dagegen wird nicht gepuffert. Sie wird als PUN: und RDR: bedient.
Wer zur Konsol-Ausgabe nicht die VDU-Karte nutzen möchte, sondern viel lieber die Grafik-Karte, dem kann geholfen werden. Dazu muß aber die Software etwas abgewandelt werden: Wie, ist auch dem dritten Teil dieses Artikels zu entnehmen. Vor dem Booten von ZCPR2 muß dann allerdings zuerst die Grafik-Software geladen werden. Ferner muß dafür gesorgt werden, daß sie dann nicht mehr überschrieben werden kann (etwa durch den

Printer-Spooler). In einer späteren ec-Ausgabe werden wir wahrscheinlich ein sogenanntes "Patch"-Programm vorstellen, das all diese Aufgaben automatisch durchführt.
Auch die Tastatur-Eingabe ist gepuffert (256 Bytes). Sie können deshalb ruhig schon ein neues Kommando eingeben, während der Rechner noch mit dem alten beschäftigt ist. Eine Ausnahme gibt es dabei: Während Disketten-Zugriffen werden - aus Zeitgründen - Interrupts verboten, und deshalb kann in dieser Zeit auch die Tastatur nicht bedient werden.
Und noch eine hübsche Kleinigkeit haben wir in die Software eingebaut: Die meisten bildschirmorientierten Textverarbeitungsprogramme (z. B. der bekannte "WordStar", aber auch der Editor zum "Turbo Pascal") benötigen eine Terminal-Funktion, die eine direkte Cursor-Adressierung erlaubt. So etwas wie "PRINT AT" in BASIC. Diese Funktion ist von uns gleich mit integriert worden: Die Zeichenfolge $27, $11, x, y (x=0..79, y=0..23) bringt den Cursor in die xte Spalte in der yten Zeile.

Kompatibilität: Schöne Aussichten...

Um das traurige Ende gleich vorweg zu nehmen: Es gibt zur Zeit nicht ein einziges CP/M-Format, daß Ihr EC-65 lesen könnte, selbst wenn wir ihm noch so viele Software-Tricks beibringen würden!
Es handelt sich hier schlichtweg um eine Hardware-Inkompatibilität.
Das bedeutet für Sie als EC-65/Z80-Anwender zuerst einmal, daß Sie auf die vom Software-Verteiler angebotenen Programme angewiesen sind, anstatt den Rahm vom riesigen CP/M-Softwaretopf abschöpfen zu können!
Aber...

Es gibt Möglichkeiten... Und voraussichtlich ab dem nächsten ec-Heft wird das ganz anders! Da wollen wir einen zweiten Floppy-Controller für den EC-65 vorstellen! Und für die Z80-Karte wird dann wohl auch ein "VARBIOS" zur Verfügung stehen, das es erlaubt, wenigstens die gängigen Formate zu bearbeiten. Und das Beste: Auch ein Harddisk-Controller sitzt auf dieser Karte!!!
Aber erst mal schön eins nach dem anderen:
Bis zum nächsten ec-Heft ist noch viel Zeit, und wir möchten ja nicht, daß Sie in dieser Zeit Ihre Z80-Karte brach liegen lassen. Als "Übergangslösung" bieten wir ein paar ganz besonders schnuckelige Programme aus dem Public-Domain Software-Pool an (werfen Sie ruhig mal einen Blick in den Artikel "EC-65: CP/M-Software" in diesem Heft - da erfahren Sie näheres).
Oder Sie kennen vielleicht jemanden, der einen "normalen" CP/M-Rechner sein Eigen nennt und an dem außerdem eine V24-Schnittstelle zur Verfügung steht? Nehmen Sie Ihren Computer und Ihre frisch erstandenen Programme, koppeln Sie die beiden Rechner über die serielle Schnittstelle, und kopieren Sie so die Software vom einen auf den anderen Computer. Vielleicht etwas umständlich, aber im großen CP/M-Diskettenformat-Wirrwarr schon längst gang und gebe.
Haben Sie keine Lust, Ihren Rechner spazieren zu tragen, so ist natürlich auch eine Rechnerkopplung per Modem möglich!
Wenn die angesprochene Controller-Karte erst einmal veröffentlicht ist, dann haben Sie das wohl nicht mehr nötig: Wir werden uns bemühen, Leser zu finden, die diese Karte besitzen und nach Art des Software-Verteilers die Disketten für Sie kopieren. Oder noch besser: Sie bauen sich den Controller selbst auf und können dann anderen den Gefallen tun!
Nach so viel Theorie und Zukunftsvisionen sollten Sie sich aber erst einmal ins ZCPR2-Abenteuer stürzen und fleißig damit herumexperimentieren.
Wir wünschen Ihnen viel Spaß dabei!

Anzeigen

FORTH-SYSTEME
Angelika Flesch

Programmieren in Forth	
Einführungsbuch, dt.	48,— DM
Thinking FORTH, L. Brodie	65,— DM
A PRELUDE für Novix 4000	98,— DM
System Guide to fig-FORTH	98,— DM
Mastering FORTH	78,— DM
Einführung in F83	
FORTH Encyclopedia	98,— DM
Pocket Guide to Forth	30,— DM
Understanding FORTH	15,— DM
FORTH Fundamentals I	56,50 DM
FORTH Fundamentals II	45,90 DM
Dokumentationen der FORTH Konferenzen	
FORML Proceedings 1980	98,— DM
FORML Proceedings 1981 660 Seiten	156,80 DM
FORML Proceedings 1982	98,— DM
FORML Proceedings 1983	98,— DM
FORML Proceedings 1984	98,— DM
Rochester Forth Proceedings	
81, 82, 83, 84, 85	je 98,— DM
Byte Reprints 8/80 - 4/81	20,— DM
Dr. Dobb's Forth Ausgaben	
9/81 bis 9/85	je 20,— DM

In unserem Katalog finden Sie eine Zusammenfassung über den Inhalt dieser Bücher.

FORTH-SYSTEME Angelika Flesch
Postfach 1226, 7820 Tit.-Neustadt, 07651/1665 od. 3304

RATEV AKTUELL

Winchester-Kits für IBM-PC/XT und komp. Systeme von Western Digital
Komplettlösung, bestehend aus: 3,5" Harddisk Fab. Tandon Controller WD 1002-SWX 2, Kabelsatz und Montagematerial zum Einbau in einen 5¼"-Ausschnitt
WD 10I, Speicherkapazität 10 MB formatiert ... DM 1899,—
WD 20I, Speicherkapazität 20 MB formatiert ... DM 2269,—

Winchester-Laufwerke
Fabrikat Seagate, Serie ST 200, 5¼" Slimline
ST 225, 20 MB formatiert, 65 mS mittlere Zugriffszeit ... DM 1330,—
Fabrikat Seagate, Serie ST 4000, 5¼" normale Bauhöhe
ST 4038, 30 MB formatiert, 40 mS mittlere Zugriffszeit ... DM 3200,—
ST 4051, 40 MB formatiert, 40 mS mittlere Zugriffszeit ... DM 3950,—
Fabrikat Maxtor, Serie XT-2000, 5¼" normale Bauhöhe
XT 2085, 70,2 MB formatiert, 30 mS mittlere Zugriffszeit ... DM 5890,—
XT 2140, 110,3 MB formatiert, 30 mS mittlere Zugriffszeit ... DM 10980,—
XT 2190, 150,4 MB formatiert, 30 mS mittlere Zugriffszeit ... DM 13290,—

Winchester-Controller für IBM-PC/XT und komp. Systeme
WD 1002 SWX 2, Fab. Western Digital ... DM 495,—
Ebenfalls erhältlich sind Controller für größere Speicherkapazitäten mit SCSI-Bus. Fordern Sie Unterlagen an.

Monitore
Hantarex, Boxer 12, monochrom, BAS-Eingang, schwarzes Gehäuse, Bandbreite 20 MHz DM 398,—
Hantarex, Boxer 12, graues Gehäuse, mit TTL-Eingang für IBM-PC/XT, ansonsten wie oben DM 446,—
Mitsubishi, XC 1402, Color Monitor, Eingang RGB (TTL-Level), Auflösung horizontal, 640 dots, vertikal 200 lines ... DM 1250,—

EPSON Floppy Laufwerke

3.5" SMD 160, 500 KB, 2x40 Spur	DM 420,—	5,25" SD 540, 1 MB, 2x80 Spur	DM 379,—
3.5" SMD 180B, 1 MB, 2x80 Spur	DM 360,—	5,25" SD 560, 1,6 MB, 2x77 Spur	DM 435,—
5,25" SD 521, 500 KB, 2x40 Spur	DM 330,—	5,25" SD 580, umschaltbar 77/80 Spur	DM 479,—

Riteman Drucker
Riteman C+, der Drucker für den C64 ... DM 849,—
Riteman F+, der EPSON FX-80 kompatible ... DM 945,—

* IBM Warenzeichen der International Business Mashines Corp.

RATEV Elektronik Vertriebs GmbH
Postfach 1601, Gothaerstr. 15, 4030 Ratingen 1
Tel.: 02102-42051-52 Telex: 8585180 Mailbox: 02102-475400
Besuchen Sie auch unser Ladenlokal!

Die 68000er-Serie

NEU

Der Mikroprozessor, das Wunderding der 70er Jahre, ist gewachsen. Die 8-Bit-Generation ist tot, es lebe die 16-Bit-Generation. Als 16-Bit-System schlechthin hat sich der 68000er mit seiner Familie durchgesetzt.

Eine ganze Reihe moderner Mikro- und Personalcomputer sind mit dem 68000er ausgerüstet. Um nur einige zu nennen: der Sinclair QL, der Amiga von Commodore und der Atari 520 ST sowie der 1040 STF. Wer nach wie vor den Selbstbau praktiziert, kann mit dem EC-68k (Elektor-Computing Nr. 3) sein eigenes System aufbauen, natürlich mit dem 68000er.

Um alle Vorteile des 68000er und seiner Familie optimal zu nutzen, sind ausführliche Informationen unerläßlich; hier sind sie.

Buch 1: Anatomie eines 16-Bitters und seiner Bausteine

Das einführende Kapitel befaßt sich mit der CPU, deren Anschlüsse und dem Programmier-Modell.

Es folgt ein Kapitel zum Thema Hardware mit Zeitdiagrammen für alle relevanten Signale.

Die Peripheriechips zum 68000er beschreibt das dritte Kapitel, das auch die 8-Bit-ICs der 65XX/68XX-Serie berücksichtigt. Schließlich das wichtigste in Kapitel 4: die sehr kräftigen Adressierungsarten. Viele Beispiele zeigen die vielfältigen Möglichkeiten.

Der Anhang informiert über den 68008er, einem Interface-Prozessor zwischen 8-Bit- und 16-Bit-Systemen: 8-Bit-Hard- mit 16-Bit-Software!

230 Seiten, 14 × 21 cm
DM/sFr 42,50 öS 340,— ISBN 3-921608-42-2

Buch 2: Alle Instruktionen auf einen Blick

Einhundertsiebenundzwanig (127!) Instruktionen sind natürlich nicht alle auf einen Blick zu übersehen, sondern auf 127 Einzelblicke. Dabei ist das gesamte Instruktionsset, alphabetisch geordnet von ABCD bis UNLK, übersichtlich und detailliert beschrieben. Bit für Bit. Das ist noch nicht alles. In der Einleitung lernen Sie alle Basis-Mnemonics und die wichtigsten Abweichungen hiervon kennen.
Für die Praxis unentbehrlich ist nicht nur der Aufbau der einzelnen Instruktionen, sondern auch der Umgang mit dem Stack. Klarer Fall: Es gibt dazu genügend Hinweise.
Eine zusammenfassende Instruktionsübersicht und die tabellarische Auflistung der Ausführungszeiten machen das Buch zu einem 100%igem Software-Nachschlagewerk.

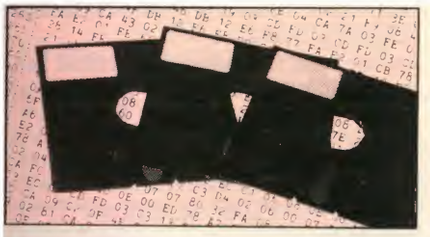
256 Seiten 14 × 21 cm
DM/sFr 44,— öS 350,— ISBN 3-921608-43-0

Kombinationspreis Buch 1 und Buch 2 DM/sFr 75,— öS 600,—

Die 68000er Handbücher sind eine Pflichtlektüre für jeden, der sich mit 16-Bit-Systemen befaßt; sei es beruflich, als Hobby oder nur aus reinem Informationsdrang heraus.

Elektor Verlag GmbH
Süsterfeldstr. 25 5100 Aachen

Bestellen Sie mit der Karte am Heftende!



CP/M auf dem EC 65 (Teil 3)

Das Software Know-How zur Z80-Karte

Frank Schmidt

Im zweiten Teil dieses Artikels haben wir schon ein bißchen umrissen, was so drinsteckt in der Software zur Z80-Karte. An dieser Stelle werfen wir einen genaueren Blick darauf und schauen uns an, wie das alles realisiert wird. Dabei geht es vor allem um den 6502-Teil des Systems.

Der "Grundstock" der Software für die Z80-Karte besteht, grob gesehen, aus folgenden sieben Teilen:

— EC-65-EPROM-Software (I/O-Treiber, Boot-Routine)
— BIOS65 Teil 1: I/O-Handler
— BIOS65 Teil 2: Diskettenroutinen für OHIO-kompatiblen Controller
— BIOS65 Teil 3: RAM-Floppy-, Printer-Spooler- Routinen (später auch die Software für den zweiten Disk-Controller)
— Z80-EPROM-Software
— BIOS80
— Betriebssystem (ZCPR2, CP/M, ...)

Alle diese Teile greifen nahtlos ineinander. Das soll ein Beispiel erläutern: Gesetzt den Fall, wir haben ZCPR2 gestartet und das System will jetzt den Prompt "A" auf dem Bildschirm ausgeben. Dazu werden die Zeichen zuerst einmal dem BIOS80 übergeben. Von dort gelangen Sie über die Z80-EPROM-Routinen zum I/O-Handler des BIOS65. Dieser entschlüsselt das von der Z80 empfangene Kommando "Zeichen an Konsole senden" und leitet die empfangenen Daten wiederum weiter: In diesem Fall an den Video-Handler im EC-65-EPROM. Und der erzeugt dann das entsprechende Ergebnis.

Der I/O-Handler im BIOS65 ist also praktisch der "Abteilungsleiter" des 6502-Computers. Er empfängt Befehle von "oben" und sorgt dann dafür, daß die anfallenden Arbeiten entsprechend verteilt werden. Ihm kommt deshalb eine zentrale Bedeutung zu. Wir werden ihn aus diesem Grund auch gleich noch ein bißchen genauer unter die Lupe nehmen.

Im nächsten ec-Heft drucken wir dann die beiden anderen Teile des BIOS65 ab: Die Diskettentreiber und den Printer-Spooler. Wahrscheinlich sind da dann auch schon die Treiber für den neuen Floppy-Controller und vielleicht sogar schon für die Harddisk dabei.

Die Z80-Software wird hier nicht abgedruckt.

Software-Schaltstelle

Werfen wir einen Blick auf diesen ersten Programmteil. Die Speicheraufteilung zeigt Bild 1. Das Programm selbst wurde zweigeteilt, um den Speicher möglichst optimal nutzen zu könen. Das Problem dabei ist, daß die Seite $23xx nicht vollständig genutzt werden kann: In ihr liegt die Speicherzelle AHOLD, die von verschiedenen I/O-Treibern benutzt wird. Ganz am Anfang des Programmes sind Vektoren zu den I/O-Routinen eingetragen. Hier muß man ansetzen, wenn man andere Geräte ins System "einschleusen" will, z.B. die Grafikkarte. Etwas Vorsicht ist bei den Einsprüngen CI und CO geboten:

CO führt normalerweise direkt zum Video-Handler des EC-65. Dieser Sprung wird aber im Laufe des Bootvorganges "gepatcht", d.h. überschrieben. Das geschieht in Zeile 2160ff. Wenn der zweite Track von Diskette gelesen wurde, steht ein etwas erweiterter VDU-Treiber zur Verfügung, der auch eine absolute Cursor-Adressierung erlaubt. Logischerweise wird die Konsolausgabe dann dorthin umgelenkt.

Der Einsprung CI führt nicht zu einer ROM-Routine. Hier wird einfach ein Zeichen aus dem Tastaturpuffer gelesen. Ist dort noch keins abgelegt, so wird halt erst darauf gewartet. Damit bei diesem Warten nicht gleich das ganze System lahm gelegt wird, wird gleichzeitig auch noch der Printer-Spooler bedient (Zeile 6320ff). Und der Tastaturpuffer wird eh über eine Interrupt-Routine gefüllt (Zeile 6550ff).

Ändert man den CI-Vekor, so wird man im allgemeinen auch die Routine ändern müssen, die den Konsolen-Status an die Z80 übergibt (Zeile 4680ff).

Es ist aus diesem Grund zu empfehlen, den CI-Vektor möglichst unberührt zu lassen.

Ab dem Label "START" geht's dann richtig los. Zuerst kommen eine ganze Reihe Initialisierungen: Stackpointer, PIA, verschiedene Pointer, Zustandsvariablen, Timer und Tastaturport. Das Inhaltsverzeichnis der RAM-Disk wird gelöscht. Dann erscheint eine erste Systemmeldung. Der CPU-Takt wird festgestellt, notfalls das Timing für 2 MHz angepaßt. Es erscheint die zweite Meldung, diesmal eine Information über die Taktfrequenz. Sodann wird die zweite Hälfte des BIOS65 von Track geladen und der CO-Vektor "verbogen". Der jetzt vorhandene Spooler wird initialisiert und schließlich werden die Tracks 2 bis 4 in den CP/M-Puffer geladen (Zeile 2270). Zum Schluß wird ein Kommando an die Z80 "abgeschickt": Kaltstart durchführen. Das kommt einer Kommando-Übergabe gleich. Als Abschiedsgruß erscheint noch die BIOS65-Versionsnummer auf dem Bildschirm, und schließlich springt die 6502 in die Endlosschleife bei Label "MAINLP" (Zeile 4460). Hier hat sie nur noch "niedere Dienste" zu tun: Sie führt die schon im letzten ec-Heft angeführten von der Z80 kommenden Befehle aus. Die meisten dieser Befehle sind recht einfach zu erledigen und auch entsprechend einfach zu verstehen. Komplizierter wird's eigentlich nur bei der Disketten-Bedienung, auf die wir jetzt mal einen genaueren Blick werfen wollen.

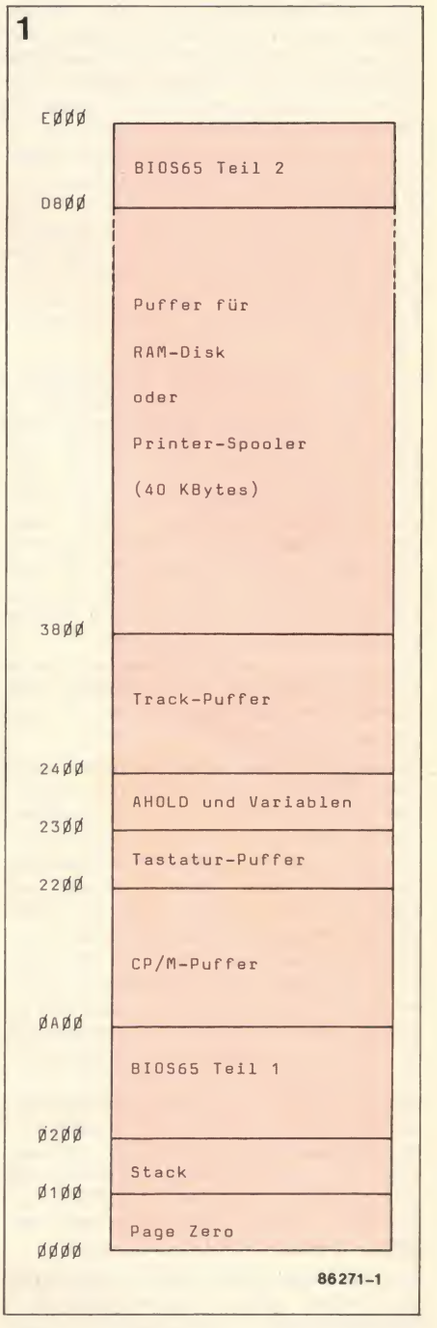


Bild 1. So sieht die Speicheraufteilung beim Arbeiten mit CP/M auf dem EC-65 aus. Es werden fast alle Ressourcen ausgeschöpft.

Disk-Handling: Im Prinzip...

Das eigentliche Problem bei der Bedienung der Diskettenlaufwerke ist, daß CP/M das Beschreiben und Lesen der Disketten grundsätzlich in "Portiönchen" von jeweils 128 Bytes abwickelt. Andererseits ist es aber empfehlenswert, die Anzahl der Sektoren pro Track auf einer Diskette möglichst klein zu halten. Denn sonst benötigt man zu viel Platz für Formatierungs-Informationen, der dann von der Gesamtkapazität der Diskette abgeht. Aus diesem Grunde arbeiten wir mit der beim OHIO-Format maximalen Sektorgröße von 2 KBytes - Sektor- und Trackgröße sind in diesem Fall identisch. Da mußten wir uns schon etwas einfallen lassen, um die Inkompatibilitäten bezüglich der Sektorgrößen zu überbrücken. Gleichzeitig galt es aber, darauf zu achten, daß der dabei verwendete Umrechnungs-Algorithmus möglichst noch eine hohe Geschwindigkeit der Diskettenzugriffe unterstützt.
Befassen wir uns zuerst einmal mit einem Lese-Zugriff:

Will das Betriebssystem Informationen von der Diskette lesen, so übergibt es dazu drei Parameter: Das Laufwerk, den Track und den (128 Byte-) Sektor. Das BIOS65 adressiert dann zuerst das gewünschte Laufwerk, positioniert dessen Schreib/Lese-Kopf über der gewünschten Spur und lädt sodann den gesamten Track in einen Puffer. Der ist 5 KByte groß, statt - wie zu erwarten - nur 2 KByte. Es sind nämlich schon weitere Diskettenformate berücksichtigt.
Nun wird die Sektornummer als Index benutzt, um aus dem Bit-Pool die gewünschten 128 Bytes abzuzweigen. Sie werden dann zur Z80 übertragen.

Wünscht jetzt das Betriebssystem noch einen (lesenden) Diskettenzugriff, so werden seine Laufwerks- und Tracknummer mit denen des vorherigen verglichen. Sind sie nicht identisch, so geschieht das Gleiche wie oben. Sind sie es aber, so macht es sich bezahlt, daß wir zuvor gleich den ganzen Track auf einmal eingelesen haben. Es brauchen jetzt nur noch die gewünschen Bytes aus dem Puffer ausgelesen und dann verschickt zu werden. Ein wirklicher Diskettenzugriff ist nicht mehr nötig! Das steigert die Geschwindigkeit der Diskettenroutinen erheblich. Allerdings nur, wenn aufeinanderfolgende Sektoren gelesen werden, wie z.B. beim Einlesen oder Abspeichern eines Programmes. Beim Arbeiten mit Compilern und Assemblern sieht das leider nicht immer so aus. Die sollten sich lieber in einer RAM-Disk austoben.

Um das Ganze noch etwas komplizierter zu machen, gibt es leider auch noch Schreib-Zugriffe auf die Diskette. Sie funktionieren im Prinzip genauso: Als erstes wird der selektierte Track in den Puffer geladen, sofern das nicht schon geschehen ist. Sodann wird aus der Sektor-Nummer die Startadresse des Sektors im Puffer berechnet und dann die Z80-Daten ab der indizierten Stelle abgelegt.
Nun sollten die aber eigentlich nicht in den Puffer, sondern auf die Diskette. Das könnte man nachholen, indem man sofort den Puffer dorthin kopiert. Dagegen spricht aber wieder das Zeit-Argument. Wollte man mit dieser Methode 16 aufeinanderfolgende Sektoren auf die Diskette schreiben, so müßte man auch ganze 16 mal den Puffer auf die Diskette kopieren, was eine halbe Ewigkeit in Anspruch nehmen würde. Wir lassen also vorläufig die Daten dort wo sie sind, merken uns aber, daß der im Puffer stehende Track noch auf die Diskette muß.
Dort kommt er hin, wenn ein neuer Track in den Puffer geladen werden muß, oder wenn es vom Betriebssystem ausdrücklich befohlen wird - das ist zum Beispiel während eines Betriebssystem-Warmstarts der Fall.
Jetzt wird auch die Warnung im zweiten Teil dieser Artikelserie klar:
Nehmen wir einmal an, daß Sie den Befehl "ERA *.*" eingeben. Dann wird das Betriebssystem zuerst die Sektoren des Tracks laden, auf dem das Inhaltsverzeichnis abgelegt ist. Sodann werden sämtliche Einträge gelöscht und das Ganze wieder zurückgeschickt. Da bei diesen Disketten-Zugriffen immer der gleiche Track angesprochen wird, geht das alles ohne wirkliches Ansprechen der Laufwerke vor sich. Ist der Befehl ausgeführt, so steht das geänderte Inhaltsverzeichnis zwar im Puffer, aber noch nicht auf Diskette! Tauschen Sie nun sofort die Diskette aus und arbeiten mit einer neuen weiter, so wird zuerst einmal das Inhaltsverzeichnis der alten Diskette weiterverwendet. Aber sowie ein Zugriff auf einen anderen Track oder ein anderes Laufwerk geschieht, wird das "alte" Inhaltsverzeichnis auf die "neue" Diskette kopiert und damit das dort vorhandene ein für allemal zerstört — die Diskette ist schlichtweg unbrauchbar geworden.
Der Warmstart vor dem Diskettenwechsel soll dafür sorgen, daß ein eventuell als geändert markierter Track auf die "alte" Scheibe findet. Der nach dem Wechsel hat die Aufgabe, das wirklich aktuelle Directory in den Puffer zu holen.

...und im Detail

Für die Beschreibung des Puffer-Inhalts benötigt man folgende Angaben: seine Adresse auf die Diskette (Laufwerk, Track), die Information, ob der Inhalt überhaupt einen Sinn macht oder ob der Puffer vielleicht leer ist und schließlich, ob der Inhalt des Puffers geändert wurde und deshalb wieder auf die Diskette zurückgeschrieben werden muß. Die Speicherstellen, die diese Informationen enthalten, haben die Namen DRIVE, TRACK, TRFLAG und WRFLAG.
Die gesamte Bedienung des Track-Puffers erledigt das Unterprogramm "SET" (Zeile 3000ff). Es übernimmt Laufwerks-, Track- und Sektor-Code von der Z80 und schreibt sie in Zwischenspeicher. Sodann wird überprüft, ob sich ein "vernünftiger" Inhalt im Puffer befindet. Wenn nicht, so wird zum Label "SETB" (Zeile 3320) verzweigt. Dort wird dann der Track in den Puffer geladen und die entsprechenden Flags gesetzt. Die Routine kann jetzt verlassen werden.
Enthält der Puffer einen Track, der aber nicht der gewünschte ist, so wird zum Label "SETA" in Zeile 3220 verzweigt. Dort wird vor dem Neueinlesen des richtigen Tracks die Variable "WRFLAG" überprüft. Enthält sie nicht den Wert Null, so muß zuerst der alte Track zurückgeschrieben werden.
CP/M übergibt alle Daten als binäre Zahlen. Das OHIO-DOS benutzt dagegen zur Angabe der Tracknummern BCD-Zahlen. Deshalb wird in Zeile 3140f nachgeschaut, ob ein OHIO-kompatibles Laufwerk (eines der ersten vier) angesprochen wurde. Ist das der Fall, so wird die Track-Nummer in den BCD-Code umgewandelt.
Sei dem wie will, eines ist immer gleich: Wird die Routine "SET" verlassen, so befindet sich der gewünschte Track im Puffer. Oder auch nicht: Es kann ja z.B. sein, daß sich im angesprochenen Laufwerk keine Diskette befand oder aber ein Track beim besten Willen nicht zu lesen war. Dann wird "SET" mit zurückgesetztem Z-Bit verlassen und das übergeordnete Programm leitet eine Fehlermeldung an CP/M.
Das Umrechnen von der Sektornummer zur effektiven Adresse, an der der Sektor zu finden ist, übernimmt "CALCTR" (Zeile 3500ff). "MOVSEC" und "RECSEC" sorgen dann für das Hin- und Herschaufeln der Bytes.

Der Rest des Programms enthält keine großartigen Besonderheiten mehr: Wer firm ist in der 6502-Maschinensprache und auch ein bißchen im Englischunterricht aufgepaßt hat, dem müßte ein Nachvollziehen ohne allzu große Schwierigkeiten möglich sein.

Ein Wort noch zu Erweiterungen der hier vorgestellten Software: Natürlich wird als nächstes auch unsere Grafikkarte "Kolorator" in das CP/M-System eingebunden werden. Ebenso Treiber für den schon öfters angesprochenen zweiten Diskettencontroller, für eine Harddisk und, und, und ...
Diese gesamte neue Software wird dann in die bestehende "eingepatcht". Das heißt, die bestehende Software wird nicht vollkommen neu überarbeitet, sondern es werden einzelne Stellen darin geändert und neue Teile hinzugefügt. Das mag sich im ersten Augenblick etwas erschreckend anhören, ist aber gar nicht so schlimm: Alle Vorbereitungen für solche Erweiterungen sind schon geschehen, sodaß das Ganze nachher nicht als "Flickwerk", sondern als homogenes Ganzes dasteht. Dies sei auch als Warnung für solche gedacht, die gern selbst am System "herumschustern", trotzdem aber mit den von uns vorgenommenen Änderungen kompatibel bleiben wollen.
Etwas anders sieht es da mit einer Anpassung der Software an den EC-65K aus. Da der sowohl Soft- als auch Hardwaremäßig recht weit vom EC-65 abweicht (z.B. keine Timer usw.), muß hier ein Teil des BIOS65 vollkommen überarbeitet werden. Es wird also in Zukunft zwei (und nur zwei) Grundversionen des BIOS65 geben.
Vom BIOS80 wird es dagegen nur eine einzige Grundversion geben.

Entwicklungskosten . . .

Regelmäßige ec-Leser wissen: Fast alle Software zum EC-65 war bisher Free-Soft. Doch nun kommen wir langsam in Bereiche, wo wir passen müssen: Die Entwickler (=Autoren) unserer Großprojekte erhalten neben dem selbstverständlichen Artikelhonorar auch ein Entwicklungshonorar als Aufwandsentschädigung für die oft immense Zeit, die hinter solchen Entwicklungen wie dem EC-65K oder diesem Z80-Zusatz steckt. Die Redaktion findet dies richtig und wichtig: Sie sollen als Leser auch weiterhin viele interessante Projekte in ec finden!

Nun war es eigentlich schon immer nicht ganz gerecht, diese Kosten über den Heftpreis auf alle ec-Leser "umzuverteilen". Und bei den jetzt eingeleiteten Projekten ist dies nicht mehr möglich. Andererseits wollen wir die praxis beibehalten, daß Sie als Leser über alle notwendigen Informationen (sprich: kommentierte Source-Listings) verfügen. Die Quadratur des Kreises. . .?

Wir bauen einfach auf das Eigeninteresse (an interessanten Projekten) und die Ehrlichkeit unserer Leser: Das Source-Listing des BIOS80 gibt es wie gewohnt und zusammen mit einem CP/M-kompatiblen Betriebssystem und einer Menge CP/M-Free-Soft auf Diskette (über die Software-Verteiler und die Fa. Rieger). Das fertig programmierte EPROM mit dem BIOS80 erhalten Sie beim elektor-Verlag. Sie brauchen in diesem Sonderfall kein Leer-EPROM an den Elektor-Software-Service (ESS) einzusenden, sondern können das fertig programmierte EPROM beim Verlag bestellen (siehe EPS-Anzeige weiter hinten in diesem Heft). Der Preis wird zwischen 50,— und 60,— DM liegen — Sie können sich also im Vergleich zu den normalen ESS-Preisen selbst ausrechnen: Der "Entwicklungskosten-Zuschlag" ist sehr, sehr maßvoll.

Und nun bitten wir Sie: Beziehen Sie Ihren BIOS80 nicht aus "dunklen Quellen", kopieren Sie das EPROM auch nicht selber weiter (wir halten nichts von Kopierschutz, es gibt keinen), sondern beziehen Sie es direkt vom Elektor-Verlag.

Ihre ec-Redaktion

```
10              ; ****************************************
20              ; *                                      *
30              ; *  CP/M BIOS for EC 65  /  6502 part   *
40              ; *       File Nr. 0                     *
50              ; *                                      *
60              ; *  Main I/O handler for the Z80 CPU    *
70              ; *       1 / 2 MHz auto select          *
80              ; *       program version 3.4            *
90              ; *       last revision date: 22/03/86   *
100             ; *                                      *
110             ; *  Written 1985/86 by  Frank Schmidt   *
120             ; *                                      *
130             ; ****************************************
140             ;
150             ;
160             ;
170             ;
180 0200         *      = $0200
190             ;
200             ;
210             ;  I/O Handling Addresses
220             ;
230 E101=        VAPAD  = $E101  ; Port A data (keyb.)
240 E104=        VATACL = $E104  ; timer 1, counter low
250 E105=        VATACH = $E105  ;    "     counter high
260 E10B=        VAACR  = $E10B  ; auxiliary control register
270 E10C=        VAPCR  = $E10C  ; peripheral control register
280 E10D=        VAIFR  = $E10D  ; Int. Flag Reg.
290 E10E=        VAIER  = $E10E  ; Int. enable Reg.
300             ;
310             ;
320             ;  Floppy Interface Addresses
330             ;
340 E002=        FDRB   = $E002  ; data/data dir reg B
350             ;
360             ;
370             ;  DOS Locations
380             ;
390 2363=        AHOLD  = $2363  ; buffer for character
400 235E=        SECTOR = $235E  ; sector number
410 235C=        DRIVE  = $235C  ; drive number
420 2370=        VECLO  = $2370  ; buffer for page 0 load vector
430 2371=        VECHI  = $2371
440 235D=        TRACK  = $235D  ; track number
450             ;
460 0A00=        CPMA   = **$0800 ; buffer for CP/M (6Kbytes)
470 2200=        KEYBUF = **$2000 ; buffer for keyboard - input
480 2400=        BLOCK  = **$2200 ; buffer for actual track (5Kbytes
490 3800=        RAMDSK = **$3600 ; buffer for RAM - disk
500 D800=        PARTWO = $D800   ; 2nd part of program
510             ;
520 E7CB=        IROL   = $E7CB
530 E7CC=        IROH   = IROL+1
540             ;
550             ;
560             ;  Z80 Handshaking Addresses
570             ;
580 E300=        PAD    = $E300  ; port A data/data direction
590 E301=        PAC    = PAD+1  ; port A control register
600 E302=        PBD    = PAD+2  ; port B data/data direction
610 E303=        PBC    = PAD+3  ; port B control register
620             ;
630             ;
640             ; Page Zero Addresses
650             ;
660 0000=        POINTL = 0      ; universal pointer
670 0001=        POINTH = 1
680 0002=        MEMNTL = 2      ; memory pointer
690 0003=        MEMNTH = 3
700 0004=        WRFLAG = 4      ; 0: data allready on disk
710 0005=        TRFLAG = 5      ; 0: data valid
720 0006=        IODRV  = 6      ; next access drive number
730 0007=        IOTRK  = 7      ; next access track number
740 0008=        CIFLAG = 8      ; 0: data in keyboard buffer
750 0009=        KEYPNT = 9      ; pointer into KEYBUF
760 000A=        COUNTR = 10     ; counter for UNLDHD - timeout
770 00FE=        MEMLO  = $FE
780 00FF=        MEMHI  = $FF
790             ;
800             ;
810 0200 4C2302          JMP START
820             ;
830             ;
840             ; I/O Vector Table
850             ;
860 0203 201206  CI      JSR GETCHA ; console input
870 0206 AD6323          LDA AHOLD
880 0209 60              RTS
890             ;
900 020A 8D6323  CO      STA AHOLD  ; video output
910 020D 4C00F0          JMP $F000
920             ;
930 0210 2070F7  READER  JSR $F770  ; serial input
940 0213 AD6323          LDA AHOLD
950 0216 60              RTS
960             ;
970 0217 8D6323  PUNCH   STA AHOLD  ; serial output
980 021A 4C7EF7          JMP $F77E
990             ;
1000 021D 8D6323 LIST    STA AHOLD  ; Centronics output
1010 0220 4C15D8         JMP INSCHR ; change into $F7B0, if no
1020                                ; printer-spooler used
1030            ;
1040            ;
1050            ;  Start of Program
1060            ;
1070 0223 A2FF   START   LDX #$FF   ; init stack pointer
1080 0225 9A             TXS
1090            ;
1100            ;
1110            ;  Init PIO
1120            ;
```

```
1130 0226 A928          LDA #%00101000
1140 0228 8D01E3        STA PAC
1150 022B A900          LDA #0    ; all inputs
1160 022D 8D00E3        STA PAD
1170 0230 A92C          LDA #%00101100
1180 0232 8D01E3        STA PAC
1190            ;
1200 0235 A928          LDA #%00101000
1210 0237 8D03E3        STA PBC
1220 023A A9FF          LDA #$FF    ; all outputs
1230 023C 8D02E3        STA PBD
1240 023F A92C          LDA #%00101100
1250 0241 8D03E3        STA PBC
1260 0244 A9FF          LDA #$FF    ; initialize handshaking
1270 0246 8D02E3        STA PBD
1280 0249 A214          LDX #20
1290            ;
1300 024B AD00E3  HNDSHK LDA PAD
1310 024E CA            DEX
1320 024F D0FA          BNE HNDSHK
1330            ;
1340            ;
1350            ; Other Initialisations
1360            ;
1370 0251 A2FF          LDX #$FF
1380 0253 8608          STX CIFLAG  ; no data in KEYBUF
1390 0255 E8            INX
1400 0256 8609          STX KEYPNT
1410 0258 8604          STX WRFLAG  ; all data on disk
1420 025A E8            INX
1430 025B 8605          STX TRFLAG  ; data in BLOCK not valid
1440 025D A935          LDA #GETISR ; init IRQ vector
1450 025F 8DCBE7        STA IRQL
1460 0262 A906          LDA #GETISR/256
1470 0264 8DCCE7        STA IRQH
1480 0267 AD01E1        LDA VAPAD   ; clear PA1 flag
1490 026A AD04E1        LDA VATACL  ; clear T1 flag
1500 026D AD0BE1        LDA VAACR   ; get auxiliary control reg
1510 0270 293F          AND #$3F    ; reset bits 6 & 7:
1520 0272 8D0BE1        STA VAACR   ; timer 1 one shot mode
1530 0275 A9C2          LDA #%11000010
1540 0277 8D0EE1        STA VAIER   ; enable timer & keyb IRQ
1550 027A 58            CLI
1560 027B A200          LDX #0      ; after booting, head
1570 027D 8E5D23        STX TRACK   ; is on track zero,
1580 0280 8E5C23        STX DRIVE   ; drive A
1590 0283 A900          LDA #RAMDSK
1600 0285 8500          STA POINTL  ; clear RAM - Directory
1610 0287 A938          LDA #RAMDSK/256
1620 0289 8501          STA POINTH
1630 028B A208          LDX #2048/256
1640 028D A000          LDY #0
1650 028F A9E5          LDA #$E5
1660            ;
1670 0291 9100    CLRMDI STA (POINTL),Y
1680 0293 C8            INY
1690 0294 D0FB          BNE CLRMDI
1700 0296 E601          INC POINTH
1710 0298 CA            DEX
1720 0299 D0F6          BNE CLRMDI
1730 029B 200204        JSR PRMES
1740 029E 20C909        JSR COUNT   ; get the CPU frequency
1750 02A1 18            CLC
1760 02A2 6907          ADC #7
1770 02A4 29F0          AND #$F0    ; mask off last bits
1780 02A6 C940          CMP #$40    ; this is the 1 MHz count
1790 02A8 D00B          BNE TWOMHZ  ; if not, 2 MHz?
1800 02AA A931          LDA #'1
1810 02AC 8D5D04        STA FREQ
1820 02AF 201B04        JSR FRQMES  ; print message
1830 02B2 4CD402        JMP MAIN    ; continue
1840            ;
1850 02B5 C980    TWOMHZ CMP #$80   ; 2 MHz count
1860 02B7 D00E          BNE NOSTND  ; no standart frequency
1870 02B9 20E009        JSR ADJTIM  ; adjust timing for 2 MHz
1880 02BC A932          LDA #'2
1890 02BE 8D5D04        STA FREQ
1900 02C1 201B04        JSR FRQMES  ; print message
1910 02C4 4CD402        JMP MAIN    ; continue
1920            ;
1930 02C7 1008    NOSTND BPL ONEADP
1940 02C9 20E009        JSR ADJTIM  ; adjust timing for 2 MHz
1950 02CC A932          LDA #'2     ; real frequency is higher
1960 02CE 8D7E04        STA FREQAD  ; than 2 MHz
1970            ;
1980 02D1 202204  ONEADP JSR PRADAP ; print a message
1990            ;
2000            ;
2010 02D4 A900    MAIN  LDA #0      ; select drive A
2020 02D6 203007        JSR SETDRV  ; if error occurs
2030 02D9 D02D          BNE ERROR   ; RESET the system
2040 02DB A901          LDA #1      ; move head on track 1
2050 02DD 20EF06        JSR SETTK
2060 02E0 D026          BNE ERROR
2070 02E2 A900          LDA #PARTWO ; init MEPNT to store
2080 02E4 8502          STA MEPNTL  ; 2nd part of program
2090 02E6 A9D8          LDA #PARTWO/256
2100 02EB 8503          STA MEPNTH
2110 02EA 78            SEI         ; disable IRQ
2120 02EB 205C07        JSR READ    ; read track into BLOCK
2130 02EE 58            CLI         ; enable IRQ again
2140 02EF D017          BNE ERROR
2150 02F1 205B03        JSR SAVEBU  ; move data to PARTWO
2160 02F4 A90F          LDA #EXTVDU ; install extended VDU routines
2170 02F6 8D0E02        STA CO+4    ; change this into NOPs,
2180 02F9 A9D8          LDA #EXTVDU/256 ; if not used
2190 02FB 8D0F02        STA CO+5
2200 02FE 201D08        JSR INSPLR  ; init spooler
2210 0301 A902          LDA #2      ; put head on track 2
2220 0303 20EF06        JSR SETTK
2230 0306 F003          BEQ MAINA   ; branch if no error
2240            ;
2250 0308 6CFCFF  ERROR JMP ($FFFC) ; JMP via RESET vector
2260            ;
2270 030B A903    MAINA LDA #3      ; read next 3 tracks
2280 030D 8500          STA POINTL
2290 030F A900          LDA #CPMA   ; store data from CPMA on
2300 0311 8502          STA MEPNTL
2310 0313 A90A          LDA #CPMA/256
2320 0315 8503          STA MEPNTH
2330            ;
2340 0317 78      LDCPM SEI
2350 031B 205C07        JSR READ    ; read one track
2360 031B 58            CLI
2370 031C D0EA          BNE ERROR
2380 031E 205B03        JSR SAVEBU
2390 0321 C600          DEC POINTL  ; next track
2400 0323 F00C          BEQ MAINB   ; was it last track?
2410 0325 AE5D23        LDX TRACK
2420 0328 E8            INX
2430 0329 8A            TXA
2440 032A 20EF06        JSR SETTK
2450 032D D0D9          BNE ERROR
2460 032F F0E6          BEQ LDCPM
2470            ;
2480 0331 A902    MAINB LDA #2      ; Z80: start program
2490 0333 204F03        JSR OUT
2500            ;
2510 0336 A9F8          LDA #$F8    ; at $F80F: COLDBOOT
2520 0338 204F03        JSR OUT
2530 033B A90F          LDA #$0F
2540 033D 204F03        JSR OUT
2550            ;
2560 0340 202904        JSR BVMES   ; print bios version message
2570            ;
2580 0343 4CAD04        JMP MAINLP  ; wait for Z80-command
2590            ;
2600            ;
2610            ;
2620            ; Subroutines
2630            ;
2640            ; IN : receive one byte from Z80
2650            ;
2660 0346 2C01E3  IN    BIT PAC     ; wait until PIA contains
2670 0349 10FB          BPL IN      ; new data
2680 034B AD00E3        LDA PAD     ; get data
2690 034E 60            RTS
2700            ;
2710            ;
2720            ; OUT : transmit one byte to Z80
2730            ;
2740 034F 2C03E3  OUT   BIT PBC     ; wait until last byte
2750 0352 10FB          BPL OUT     ; accepted
2760 0354 CD02E3        CMP PBD     ; reset PBC flag
```

```
2770 0357 8D02E3        STA PBD     ; store data in PIA
2780 035A 60            RTS
2790            ;
2800            ; SAVEBU : move BLOCK to (MEPNTL)
2810            ;
2820 035B A900    SAVEBU LDA #BLOCK
2830 035D 85FE          STA MEMLO
2840 035F A924          LDA #BLOCK/256
2850 0361 85FF          STA MEMHI
2860 0363 A208          LDX #8      ; there are 8 pages
2870 0365 A000          LDY #0
2880            ;
2890 0367 B1FE    SAVELP LDA (MEMLO),Y
2900 0369 9102          STA (MEPNTL),Y
2910 036B C8            INY
2920 036C D0F9          BNE SAVELP
2930 036E E6FF          INC MEMHI
2940 0370 E603          INC MEPNTH
2950 0372 CA            DEX
2960 0373 D0F2          BNE SAVELP
2970 0375 60            RTS
2980            ;
2990            ;
3000            ; SET : set drive, track
3010            ;
3020 0376 204603  SET   JSR IN      ; drive number
3030 0379 8506          STA IODRV   ; save it
3040 037B 204603        JSR IN      ; track number
3050 037E 8507          STA IOTRK
3060 0380 204603        JSR IN      ; sector number
3070 0383 8D5E23        STA SECTOR
3080 0386 A505          LDA TRFLAG  ; valid data in BLOCK?
3090 0388 D02B          BNE SETB    ; if not
3100 038A A506          LDA IODRV   ; drive already selected?
3110 038C CD5C23        CMP DRIVE
3120 038F D012          BNE SETA    ; no
3130 0391 A507          LDA IOTRK   ; track already selected?
3140 0393 AE5C23        LDX DRIVE   ; if OHIO format, use BCD-code
3150 0396 E004          CPX #4      ; for track number
3160 0398 B003          BCS SETAA
3170 039A 20FE05        JSR HEXBCD
3180 039D CD5D23  SETAA CMP TRACK
3190 03A0 D001          BNE SETA    ; no
3200 03A2 60            RTS         ; all right, ret Z=1
3210            ;
3220 03A3 A504    SETA  LDA WRFLAG  ; data in BLOCK must be changed
3230                                ; is it allready on disk?
3240 03A5 F00B          BEQ SETB    ; yes
3250 03A7 A900          LDA #0      ; data in BLOCK exists on disk
3260 03A9 8504          STA WRFLAG
3270 03AB 78            SEI
3280 03AC 20CC08        JSR WRITE   ; put it on disk
3290 03AF 58            CLI
3300 03B0 D01D          BNE SETEND  ; error
3310            ;
3320 03B2 A901    SETB  LDA #1      ; no valid data in BLOCK
3330 03B4 8505          STA TRFLAG
3340 03B6 A506          LDA IODRV
3350 03B8 203007        JSR SETDRV
3360 03BB D012          BNE SETEND  ; error
3370 03BD A507          LDA IOTRK
3380 03BF 20EF06        JSR SETTK
3390 03C2 D00B          BNE SETEND
3400 03C4 78            SEI
3410 03C5 205C07        JSR READ    ; load new track
3420 03C8 58            CLI
3430 03C9 D004          BNE SETEND
3440 03CB A900          LDA #0      ; valid data in BLOCK
3450 03CD 8505          STA TRFLAG
3460            ;
3470 03CF 60      SETEND RTS        ; Z=1: ok   Z=0: error
3480            ;
3490            ;
3500            ; CALCTR : calculate sector start address
3510            ;
3520 03D0 AD5E23  CALCTR LDA SECTOR
3530 03D3 4A            LSR A
3540 03D4 08            PHP
3550 03D5 18            CLC
3560 03D6 65FF          ADC MEMHI
3570 03D8 85FF          STA MEMHI
3580 03DA 28            PLP
3590 03DB 9006          BCC CALCA
3600 03DD A97F          LDA #$7F
3610 03DF 65FE          ADC MEMLO
3620 03E1 85FE          STA MEMLO
3630            ;
3640 03E3 A000    CALCA LDY #0
3650 03E5 60            RTS
3660            ;
3670            ;
3680            ; MOVSEC : move sector to Z80
3690            ;
3700 03E6 20D003  MOVSEC JSR CALCTR
3710            ;
3720 03E9 B1FE    MVSCLP LDA (MEMLO),Y
3730 03EB 204F03        JSR OUT
3740 03EE C8            INY
3750 03EF C080          CPY #$80
3760 03F1 D0F6          BNE MVSCLP
3770 03F3 60            RTS
3780            ;
3790            ;
3800            ; RECSEC : receive sector from Z80
3810            ;
3820 03F4 20D003  RECSEC JSR CALCTR
3830            ;
3840 03F7 204603  RCSCLP JSR IN
3850 03FA 91FE          STA (MEMLO),Y
3860 03FC C8            INY
3870 03FD C080          CPY #$80
3880 03FF D0F6          BNE RCSCLP
3890 0401 60            RTS
3900            ;
3910            ;
3920            ; PRMES : print message
3930            ;
3940 0402 A930    PRMES  LDA #MESS
3950 0404 A204          LDX #MESS/256
3960            ;
3970 0406 8500    PRINT  STA POINTL
3980 0408 8601          STX POINTH
3990 040A A000          LDY #0
4000            ;
4010 040C B100    PRMSLP LDA (POINTL),Y
4020 040E F00A          BEQ RETPRM
4030 0410 8402          STY MEPNTL
4040 0412 200A02        JSR CO
4050 0415 A402          LDY MEPNTL
4060 0417 C8            INY
4070 0418 D0F2          BNE PRMSLP
4080            ;
4090 041A 60      RETPRM RTS
4100            ;
4110            ;
4120 041B A951    FRQMES LDA #FMESS
4130 041D A204          LDX #FMESS/256
4140 041F 4C0604        JMP PRINT
4150            ;
4160            ;
4170 0422 A966    PRADAP LDA #AMESS
4180 0424 A204          LDX #AMESS/256
4190 0426 4C0604        JMP PRINT
4200            ;
4210            ;
4220 0429 A99A    BVMES  LDA #BVMESS
4230 042B A204          LDX #BVMESS/256
4240 042D 4C0604        JMP PRINT
4250            ;
4260            ;
4270 0430 1B      MESS   .BYTE $1B,'1',$D
4270 0431 31
4270 0432 0D
4280 0433 20            .BYTE '  == EC 65  CP/M system ==',$D,$A,$A
4280 0434 20
4280 0435 3D
4280 0436 3D
4280 0437 20
4280 0438 45
4280 0439 43
4280 043A 20
4280 043B 36
4280 043C 35
4280 043D 20
4280 043E 20
4280 043F 43
4280 0440 50
4280 0441 2F
4280 0442 4D
```

```
4280 0443 20
4280 0444 73                            4280 044A 20
4280 0445 79                            4280 044B 3D
4280 0446 73   .                        4280 044C 3B
4280 0447 74                            4280 044D 0D
4280 0448 65                            4280 044E 0A
4280 0449 6D                            4280 044F 0A
4290 0450 00            .BYTE 0
4300            ;
4310            ;
4320 0451 36    FMESS   .BYTE '6502 speed:
4320 0452 35
4320 0453 30
4320 0454 32
4320 0455 20
4320 0456 73
4320 0457 70
4320 0458 65
4320 0459 65
4320 045A 64
4320 045B 3A
4320 045C 20
4330 045D 78    FREQ    .BYTE 'x MHz',$D,$A,$A
4330 045E 20
4330 045F 4D
4330 0460 48
4330 0461 7A
4330 0462 0D
4330 0463 0A
4330 0464 0A
4340 0465 00            .BYTE 0
4350            ;
4360            ;
4370 0466 4E    AMESS   .BYTE 'No standard 6502 speed!
4370 0467 6F
4370 0468 20
4370 0469 73
4370 046A 74
4370 046B 61                            4370 0475 32
4370 046C 6E                            4370 0476 20
4370 046D 64                            4370 0477 73
4370 046E 61                            4370 0478 70
4370 046F 72                            4370 0479 65
4370 0470 64                            4370 047A 65
4370 0471 20                            4370 047B 64
4370 0472 36                            4370 047C 21
4370 0473 35                            4370 047D 20
4370 0474 30
4380 047E 31    FREQAD .BYTE '1 MHz delays installed!',$D,$A,$A,$A
4380 047F 20
4380 0480 4D
4380 0481 48                            4380 048E 74
4380 0482 7A                            4380 048F 61
4380 0483 20                            4380 0490 6C
4380 0484 64                            4380 0491 6C
4380 0485 65                            4380 0492 65
4380 0486 6C                            4380 0493 64
4380 0487 61                            4380 0494 21
4380 0488 79                            4380 0495 0D
4380 0489 73                            4380 0496 0A
4380 048A 20                            4380 0497 0A
4380 048B 69                            4380 0498 0A
4380 048C 6E
4380 048D 73
4390 0499 00            .BYTE 0
4400            ;
4410            ;
4420 049A 43    BVMESS .BYTE 'CBIOS85 vers 3.4',$D,$A
4420 049B 42
4420 049C 49
4420 049D 4F
4420 049E 53
4420 049F 38
4420 04A0 35
4420 04A1 20
4420 04A2 76
4420 04A3 65
4420 04A4 72
4420 04A5 73
4420 04A6 20
4420 04A7 33
4420 04A8 2E
4420 04A9 34
4420 04AA 0D
4420 04AB 0A
4430 04AC 00            .BYTE 0
4440            ;
4450            ;
4460 04AD 2018D8 MAINLP JSR PREADY  ; serve printer spooler
4470 04B0 2C01E3        BIT PAC     ; loop till PIA contains
4480 04B3 10F8          BPL MAINLP  ; new Z80 command
4490 04B5 AD00E3        LDA PAD     ; get it
4500 04B8 C900          CMP #0      ; console input
4510 04BA D009          BNE CONOUT
4520 04BC 200302        JSR CI
4530 04BF 204F03        JSR OUT
4540 04C2 4CAD04        JMP MAINLP
4550            ;
4560 04C5 C901  CONOUT  CMP #1      ; console output
4570 04C7 D009          BNE LISTO
4580 04C9 204603        JSR IN
4590 04CC 200A02        JSR CO
4600 04CF 4CAD04        JMP MAINLP
4610            ;
4620 04D2 C902  LISTO   CMP #2      ; list output
4630 04D4 D009          BNE CONST
4640 04D6 204603        JSR IN
4650 04D9 201D02        JSR LIST
4660 04DC 4CAD04        JMP MAINLP
4670            ;
4680 04DF C903  CONST   CMP #3      ; console status
4690 04E1 D00A          BNE PUNCHO
4700 04E3 A508          LDA CIFLAG
4710 04E5 49FF          EOR #$FF
4720 04E7 204F03        JSR OUT
4730 04EA 4CAD04        JMP MAINLP
4740            ;
4750 04ED C904  PUNCHO  CMP #4      ; punch output
4760 04EF D009          BNE READRI
4770 04F1 204603        JSR IN
4780 04F4 201702        JSR PUNCH
4790 04F7 4CAD04        JMP MAINLP
4800            ;
4810 04FA C905  READRI  CMP #5      ; reader input
4820 04FC D009          BNE READDK
4830 04FE 201002        JSR READER
4840 0501 204F03        JSR OUT
4850 0504 4CAD04        JMP MAINLP
4860            ;
4870 0507 C906  READDK  CMP #6      ; read sector
4880 0509 D018          BNE WRITDK
4890 050B 207603        JSR SET     ; load track into BLOCK
4900 050E D033          BNE ERROUT  ; error
4910 0510 A900          LDA #0      ; all right, track in
4920 0512 204F03        JSR OUT     ; buffer
4930 0515 A900          LDA #BLOCK  ; read data from BLOCK
4940 0517 85FE          STA MEMLO
4950 0519 A924          LDA #BLOCK/256
4960 051B 85FF          STA MEMHI
4970 051D 20E603        JSR MOVSEC  ; move sector to Z80
4980 0520 4CAD04        JMP MAINLP
4990            ;
5000 0523 C907  WRITDK  CMP #7      ; write sector
5010 0525 D026          BNE HANCPM
5020 0527 207603        JSR SET     ; load track into BLOCK
5030 052A D017          BNE ERROUT  ; error
5040 052C A900          LDA #0      ; all right, track in
5050 052E 204F03        JSR OUT     ; buffer
5060 0531 A900          LDA #BLOCK  ; write data into BLOCK
5070 0533 85FE          STA MEMLO
5080 0535 A924          LDA #BLOCK/256
5090 0537 85FF          STA MEMHI
5100 0539 20F403        JSR RECSEC  ; receive sector from Z80
5110 053C A9FF          LDA #$FF    ; data not yet on disk
5120 053E 8504          STA WRFLAG
5130 0540 4CAD04        JMP MAINLP
5140            ;
5150            ;
5160 0543 A901  ERROUT  LDA #1      ; read/write error
5170 0545 8505          STA TRFLAG  ; no valid data
5180 0547 204F03        JSR OUT
5190 054A 4CAD04        JMP MAINLP
5200            ;
5210 054D C908  HANCPM  CMP #8      ; hand CP/M to Z80
```

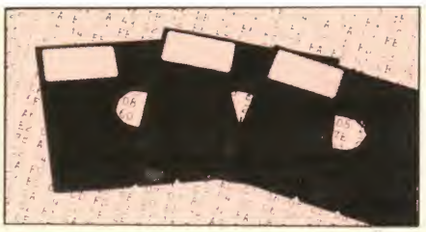
```
5220 054F D02A          BNE CLRBUF
5230 0551 AD0B12        LDA CPMA+$800 ; get CP/M start addr
5240 0554 38            SEC
5250 0555 E908          SBC #8
5260 0557 204F03        JSR OUT
5270 055A A900          LDA #0
5280 055C 204F03        JSR OUT
5290 055F A900          LDA #CPMA
5300 0561 85FE          STA MEMLO
5310 0563 A90A          LDA #CPMA/256
5320 0565 85FF          STA MEMHI
5330 0567 A218          LDX #24     ; handle 24*256=6K bytes
5340 0569 A000          LDY #0
5350            ;
5360 056B B1FE    HANLPB LDA (MEMLO),Y
5370 056D 204F03        JSR OUT
5380 0570 C8            INY
5390 0571 D0F8          BNE HANLPB
5400 0573 E6FF          INC MEMHI
5410 0575 CA            DEX
5420 0576 D0F3          BNE HANLPB
5430 0578 4CAD04        JMP MAINLP
5440            ;
5450            ;
5460 057B C909    CLRBUF CMP #9     ; clear track buffer
5470 057D D017          BNE INIDSK
5480 057F A505          LDA TRFLAG
5490 0581 D009          BNE ENDCLR  ; if no valid data in BLOCK
5500 0583 A504          LDA WRFLAG
5510 0585 F005          BEQ ENDCLR  ; if data allready on disk
5520 0587 78            SEI
5530 0588 20DC08        JSR WRITE   ; put track on disk
5540 058B 58            CLI
5550            ;
5560 058C A200    ENDCLR LDX #0
5570 058E 8604          STX WRFLAG  ; track on disk,
5580 0590 E8            INX
5590 0591 8605          STX TRFLAG  ; track not valid
5600 0593 4CAD04        JMP MAINLP
5610            ;
5620            ;
5630 0596 C90A    INIDSK CMP #$A    ; init track on disk
5640 0598 D02F          BNE STODAT
5650 059A 204603        JSR IN      ; get drive nr
5660 059D 8506          STA IODRV
5670 059F 204603        JSR IN      ; get track nr
5680 05A2 8507          STA IOTRK
5690 05A4 A505          LDA TRFLAG  ; valid data in track buffer?
5700 05A6 F01D          BEQ INIERR  ; then error
5710 05A8 A506          LDA IODRV
5720 05AA C904          CMP #4      ; one of drives a..d?
5730 05AC B017          BCS INIERR  ; error if not
5740 05AE CD5C22        CMP DRIVE   ; drive allready selected?
5750 05B1 F005          BEQ INIDA
5760 05B3 203007        JSR SETDRV  ; select drive
5770 05B6 D00D          BNE INIERR
5780            ;
5790 05B8 206309  INIDA JSR INITRK  ; initialize the track
5800 05BB D008          BNE INIERR
5810 05BD A900          LDA #0
5820 05BF 204F03        JSR OUT     ; o.k.: track initialized
5830 05C2 4CAD04        JMP MAINLP
5840            ;
5850 05C5 A901    INIERR LDA #1
5860 05C7 D0F6          BNE INIDA+7
5870            ;
5880 05C9 C90B    STODAT CMP #$B    ; store data in address
5890 05CB D015          BNE LODAT
5900 05CD 204603        JSR IN      ; get address upper part
5910 05D0 8DDE05        STA STORE+2 ; in code modification
5920 05D3 204603        JSR IN      ; address lower part
5930 05D6 8DDD05        STA STORE+1
5940 05D9 204603        JSR IN      ; fetch data
5950 05DC 8DFFFF  STORE STA $FFFF   ; store data in address
5960 05DF 4CAD04        JMP MAINLP
5970            ;
5980 05E2 C90C    LODAT CMP #$C     ; load data from address
5990 05E4 D015          BNE EXIT    ; ERROR: leave CP/M system
6000 05E6 204603        JSR IN      ; get upper address byte
6010 05E9 8DF405        STA LOAD+2  ; in code modification
6020 05EC 204603        JSR IN      ; lower part of address
6030 05EF 8DF305        STA LOAD+1
6040 05F2 ADFFFF  LOAD  LDA $FFFF   ; load data
6050 05F5 204F03        JSR OUT     ; and hand it to Z80
6060 05F8 4CAD04        JMP MAINLP
6070            ;
6080            ;
6090 05FB 4C0B03  EXIT  JMP ERROR   ; JMP via reset vector
6100            ;
6110            ;
6120            ;
6130            ; HEXBCD : convert HEX byte in acc into BCD
6140            ;
6150 05FE 38      HEXBCD SEC
6160 05FF A2FF          LDX #$FF
6170            ;
6180 0601 E8      HEXLP  INX
6190 0602 E90A          SBC #10
6200 0604 B0FB          BCS HEXLP
6210 0606 690A          ADC #10
6220 0608 8510          STA $10
6230 060A 8A            TXA
6240 060B 0A            ASL A
6250 060C 0A            ASL A
6260 060D 0A            ASL A
6270 060E 0A            ASL A
6280 060F 0510          ORA $10
6290 0611 60            RTS
6300            ;
6310            ;
6320            ; GETCHA : get character from keyboard buffer
6330            ;
6340 0612 201B08  GETCHA JSR PREADY ; serve printer spooler
6350 0615 A508          LDA CIFLAG  ; character in buffer?
6360 0617 D0F9          BNE GETCHA  ; if not then wait
6370 0619 AD0022        LDA KEYBUF  ; get character
6380 061C 8D6323        STA AHOLD   ; and save it
6390 061F A201          LDX #1
6400 0621 78            SEI
6410            ;                   ; and update buffer
6420 0622 BD0022  GETLP  LDA KEYBUF,X
6430 0625 9DFF21        STA KEYBUF-1,X
6440 0628 E8            INX
6450 0629 D0F7          BNE GETLP
6460 062B C609          DEC KEYPNT
6470 062D D004          BNE RETGET  ; if this was last character,
6480 062F A2FF          LDX #$FF    ; then set flag: buffer empty
6490 0631 8608          STX CIFLAG
6500            ;
6510 0633 58      RETGET CLI
6520 0634 60            RTS
6530            ;
6540            ;
6550            ; GETISR : interrupt service routine
6560            ;
6570 0635 48      GETISR PHA        ; save A, X
6580 0636 8A            TXA
6590 0637 48            PHA
6600 0638 AD0DE1        LDA VAIFR   ; get the flag register
6610 063B 2902          AND #2      ; mask off PA1 bit (keyb)
6620 063D F015          BEQ NOGET   ; branch if no keyb IRQ
6630 063F AD01E1        LDA VAP4D   ; get data
6640 0642 297F          AND #$7F
6650 0644 A609          LDX KEYPNT
6660 0646 E0FF          CPX #$FF    ; KEYBUF full?
6670 0648 F00A          BEQ NOGET   ; then forget character
6680 064A 9D0022        STA KEYBUF,X
6690 064D E8            INX
6700 064E 8609          STX KEYPNT
6710 0650 A200          LDX #0
6720 0652 8608          STX CIFLAG  ; update flag
6730            ;
6740 0654 AD0DE1  NOGET  LDA VAIFR  ; get the flag reg. again
6750 0657 2940          AND #$40    ; mask off T1 bit
6760 0659 F017          BEQ RETISR ; branch if no timer IRQ
6770 065B AD04E1        LDA VATACL ; clear IRQ flag
6780 065E C60A          DEC COUNTR
6790 0660 D00B          BNE LDCNTR
6800 0662 A980          LDA #$80    ; unload drive head
6810 0664 0D02E0        ORA FDRB    ; set PB7
6820 0667 8D02E0        STA FDRB
6830 066A 4C7206        JMP RETISR
6840            ;
6850 066D A9FF    LDCNTR LDA #$FF   ; load the counter again
```

```
6860 066F BD05E1        STA VATACH ; and start it
6870            ;
6880 0672 68    RETISR PLA
6890 0673 AA            TAX
6900 0674 68            PLA
6910 0675 58            CLI
6920 0676 40            RTI
6930            ;
6940            ;
6950            ;
6960            ; disk i/o routines
6970            ;
6980 06EF=      SETTK  = $6EF
6990 0730=      SETDRV = $730
7000 075C=      READ   = $75C
7010 0B3C=      WRITE  = $B3C
7020 0963=      INITRK = $963
7030            ;
7040            ; routines for timing adjustment
7050            ;
7060 09C9=      COUNT  = $9C9
7070 09E0=      ADJTIM = $9E0
7080            ;
7090            ; extended VDU & spooler routines
7100            ;
7110 DB0F=      EXTVDU = $DB0F
7120 DB12=      INSPLR = $DB12
7130 DB15=      INSCHR = $DB15
7140 DB18=      PREADY = $DB18

10              ; ****************************************
20              ; *                                      *
30              ; *  CP/M BIOS for EC 65 / 6502 part     *
40              ; *       File Nr. 1                     *
50              ; *                                      *
60              ; *  Disk I/O Routines for OHIO kompatible *
70              ; *          Floppy Interface Board      *
80              ; *          using timer at $E100        *
90              ; *          35/40/80 track version      *
100             ; *          1/2 MHz auto select         *
110             ; *          program version 3.4         *
120             ; *          last revision date: 22.03.86 *
130             ; *                                      *
140             ; *  Written  1985/86 by  Frank Schmidt  *
150             ; *                                      *
160             ; ****************************************
170             ;
180             ;
190 0680                * = $680
200             ;
210             ;
220             ; Addresses of the main program
230             ;
240 05FE=       HEXBCD = $5FE
250             ;
260             ;
270             ; Floppy Interface Addresses
280             ;
290 E000=       FDRA   = $E000  ; data/data dir. reg A
300 E001=       FCRA   = $E001  ; control register A
310 E002=       FDRB   = $E002  ; data/data dir. reg B
320 E003=       FCRB   = $E003  ; control register B
330             ;
340 E010=       DACIA  = $E010  ; ACIA control/status reg
350 E011=       DACIA  = $E011  ; ACIA data register(s)
360             ;
370             ;
380             ;
390             ; page zero addresses
400             ;
410 0000=       POINTL = $0000  ; universal pointer
420 0001=       POINTH = $0001
430 0002=       MEPNTL = $0002
440 0003=       MEPNTH = $0003
450 0004=       WRFLAG = $0004  ; 0: data allready on disk
460 0005=       TRFLAG = $0005  ; 0: data valid
470 0006=       IODRV  = $0006  ; next access drive number
480 0007=       IOTRK  = $0007  ; next access track number
490 0008=       CIFLAG = $0008  ; 0: data in keyboard buffer
500 0009=       KEYPNT = $0009  ; pointer for keybuf
510 000A=       COUNTR = $000A  ; counter for UNLDHD - timeout
520 00F9=       PAGES  = $00F9  ; nr of pages to be stored
530 00FA=       TRBUF  = $00FA  ; temporary buffer for track nr
540 00FB=       IOSTPS = $00FB  ; max I/Osteps within r/w attempt
550 00FD=       ATTMPT = $00FD  ; max number of attempts
560 00FE=       MEMLO  = $00FE  ; read/write memory pointer
570 00FF=       MEMHI  = $00FF
580             ;
590             ;
600             ; DOS Locations
610             ;
620 235E=       SECTOR = $235E  ; sector number
630 235C=       DRIVE  = $235C  ; drive number
640 2370=       VECLO  = $2370  ; buffer for page 0 load vector
650 2371=       VECHI  = $2371
660 235D=       TRACK  = $235D  ; current track number
670             ;
680 0200=       START  = $0200  ; start of 6502 program
690             ;
700 0A00=       CPMA   = START+$0800 ; buffer for CP/M (6Kbytes)
710 2200=       KEYBUF = START+$2000 ; buffer for keyboard input
720 2400=       BLOCK  = START+$2200 ; buffer for track (5Kbytes)
730 3800=       RAMDSK = START+$3600 ; buffer for RAM - disk
740             ;
750             ;
760             ; Addresses of the bootstrap loader
770             ;
780 F50A=       DUMMY  = $F50A  ; a JSR needs 12 us @ 1 MHz
790 F4F6=       FDELAY = $F4F6  ; a JSR needs X*1248+14 us @ 1 MHz
800 F501=       RBYTE  = $F501  ; read one byte from disk
810             ;
820             ;
830             ; Timer addresses
840             ;
850 E104=       VATACL = $E104  ; timer 1 counter - low
860 E105=       VATACH = $E105  ; timer 1 counter - high
870 E10E=       VAIER  = $E10E  ; interrupt enable register
880             ;
890             ;
900             ;
910             ; HOME : put head on track 0
920             ;
930 0680 A954   HOME   LDA #84     ; do max 84 STPINs
940 0682 85FD          STA ATTMPT
950 0684 20CB06        JSR LDHEAD  ; do a head load
960 0687 20AE06        JSR STPOUT
970 068A 20C606        JSR STPDEL  ; wait & return with X=0
980 068D 8E5D23        STX TRACK   ; update TRACK
990             ;
1000 0690 20A706 HOMELP JSR STPIN
1010 0693 A902          LDA #2      ; PA1 is TR0 input
1020 0695 2C00E0        BIT FDRA    ; is it low?
1030 0698 F007          BEQ RETHOM  ; if, then return
1040 069A C6FD          DEC ATTMPT  ; was it last attempt?
1050 069C 10F2          BPL HOMELP  ; if not, then next one
1060 069E CE5D23        DEC TRACK   ; mark TRACK as not valid
1070            ;
1080 06A1 08    RETHOM PHP          ; save Z-flag
1090 06A2 20DE06        JSR UNLDHD  ; unload the head again
1100 06A5 28            PLP         ; restore Z-flag
1110 06A6 60            RTS         ; Z=1: ok   Z=0: error
1120            ;
1125            ;
1130            ; STPIN / STPOUT : move head one step in / out
1140            ;
1150 06A7 AD02E0 STPIN  LDA FDRB    ; set DIR bit
1160 06AA 0904          ORA #4
1170 06AC D005          BNE STPOUT+5 ; continue
1180            ;
1190 06AE AD02E0 STPOUT LDA FDRB    ; reset DIR bit
1200 06B1 29FE          AND #$FB
1210 06B3 8D02E0        STA FDRB
1220 06B6 200AF5        JSR DUMMY   ; delay 12 us @ 1 MHz
1230 06B9 29F7          AND #$F7    ; reset bit 3 = STEP
1240 06BB 8D02E0        STA FDRB
1250 06BE 200AF5        JSR DUMMY   ; delay 12 us @ 1 MHz
1260 06C1 0908          ORA #8      ; set STEP bit
1270 06C3 8D02E0        STA FDRB
1280            ;
1285            ;
1290            ; STPDEL : delay approx 10 ms
1300            ;
1310 06C6 A210  STPDEL LDX #16     ; (34)
1320 06C8 4CF6F4        JMP FDELAY
```

```
1330            ;
1340            ;
1350            ; LDHEAD : do a head-load
1360            ;
1370 06CB A9BF  LDHEAD LDA #%10111111
1380 06CD 2D0EE1       AND VAIER  ; disable timer IRQ
1390 06D0 8D0EE1       STA VAIER
1400 06D3 A97F         LDA #$7F   ; reset PB7
1410 06D5 2D02E0       AND PDRB
1420 06D8 8D02E0       STA PDRB
1430 06DB 4CC606       JMP STPDEL ; wait & return
1440            ;
1450            ;
1460            ; UNLDHD : unload the head
1470            ;
1480 06DE A2FF  UNLDHD LDX #$FF
1490 06E0 A940         LDA #$40
1500 06E2 850A         STA COUNTR ; wait for $40 timeouts
1510 06E4 0D0EE1       ORA VAIER  ; allow timer IRQ
1520 06E7 8D0EE1       STA VAIER
1530 06EA 8E05E1       STX VATACH ; start timer
1540 06ED E8           INX        ; set Z flag
1550 06EE 60           RTS
1560            ;
1565            ;
1570            ; SETTK : set head on track in acc
1580            ;
1590 06EF AA    SETTK  TAX        ; save track number
1600 06F0 AD5C23       LDA DRIVE
1610 06F3 C904         CMP #4
1620 06F5 8A           TXA        ; restore track number
1630 06F6 9003         BCC SETTK0
1640 06F8 4C00D8       JMP FSETTK
1650            ;
1660 06FB F083  SETTK0 BEQ HOME
1670 06FD 20FE05       JSR HEXBCD
1680 0700 85FA         STA TRBUF
1690 0702 C980         CMP #$80   ; track nr < 80?
1700 0704 B027         BCS RETSET ; error exit
1710 0706 20CB06       JSR LDHEAD ; load the head
1720            ;
1730 0709 A5FA  SETLP  LDA TRBUF  ; track reached?
1740 070B CD5D23       CMP TRACK
1750 070E F01A         BEQ RESET  ; unload head & ret
1760 0710 B007         BCS SETTKA ; if track nr too big
1770 0712 20A706       JSR STPIN  ; then step in
1780 0715 A999         LDA #$99   ; = -1
1790 0717 D005         BNE ADJTRK ; branch always
1800            ;
1810 0719 20AE06 SETTKA JSR STPOUT ; else step out
1820 071C A901         LDA #1
1830            ;
1835            ;
1840 071E 18    ADJTRK CLC
1850 071F F8           SED
1860 0720 6D5D23       ADC TRACK
1870 0723 8D5D23       STA TRACK
1880 0726 D8           CLD
1890 0727 4C0507       JMP SETLP
1900            ;
1910 072A A900  RESET  LDA #0
1920 072C 60           RTS
1930            ;
1940 072D 0901  RETSET ORA #1     ; clear Z-bit
1950 072F 60           RTS        ; Z=1: ok   Z=0: error
1960            ;
1970            ;
1980            ; SETDRV : select drive
1990            ;
2000 0730 C905  SETDRV CMP #5     ; drivenr < 5 ?
2010 0732 B0F9         BCS RETSET ; else error exit
2020 0734 CD5C23       CMP DRIVE  ; drive already selected?
2030 0737 F0F6         BEQ RETSET+2 ; return with Z=1
2040 0739 8D5C23       STA DRIVE  ; new drive nr
2050 073C C904         CMP #4
2060 073E 9003         BCC STDRV0
2070 0740 4C03D9       JMP FSTDRV
2080            ;
2090 0743 0A    STDRV0 ASL A      ; X=0,2,4,6
2100 0744 AA           TAX
2110 0745 BD5407       LDA DRVTAB,X
2120 0748 8D00E0       STA PDRA
2130 074B BD5507       LDA DRVTAB+1,X
2140 074E 8D02E0       STA PDRB
2150 0751 4C8006       JMP HOME   ; put head on tr 0 & ret
2160            ;
2170 0754 40FF  DRVTAB .DBYTE $40FF
2180 0756 00FF         .DBYTE $00FF
2190 0758 40DF         .DBYTE $40DF
2200 075A 00DF         .DBYTE $00DF
2210            ;
2220            ;
2230            ; READ : read  memory into BUFFER
2240            ;
2250 075C AD5C23 READ  LDA DRIVE
2260 075F C904         CMP #4
2270 0761 9003         BCC READ0
2280 0763 4C06D8       JMP FREAD
2290            ;
2300 0766 A903  READ0  LDA #3     ; max 3 attempts with STPIN/OUT
2310 0768 85FB         STA IOSTPS
2320 076A A907  READA  LDA #7     ; max 7 simple read attempts
2330 076C 85FD         STA ATTMPT
2340 076E A900  READB  LDA #BLOCK ; store data start address
2350 0770 85FE         STA MEMLO
2360 0772 A924         LDA #BLOCK/256
2370 0774 85FF         STA MEMHI
2380 0776 AD5D23       LDA TRACK  ; read track zero?
2390 0779 D020         BNE READC
2400            ;
2410 077B A901         LDA #1     ; no STPIN/OUT
2420 077D 85FB         STA IOSTPS
2430 077F 20DA07       JSR TRKBEG ; init ACIA, load head, wait in
2440 0782 202908       JSR RBYTET ; read byte & test index
2450 0785 D03C         BNE NXREAD ; error, next read attempt
2460 0787 8D7123       STA VECHI  ; save the load vector
2470 078A 202908       JSR RBYTET
2480 078D D034         BNE NXREAD
2490 078F 8D7023       STA VECLO
2500 0792 2001F5       JSR RBYTE  ; sector length
2510 0795 C908         CMP #8
2520 0797 D02A         BNE NXREAD ; not the right format
2530 0799 F008         BEQ READD
2540            ;
2550 079B 20F207 READC JSR RWBEG  ; load head, wait until index
2560                              ; pulse over and read header
2570 079E 201508       JSR HEADC  ; read sector header
2580 07A1 D020         BNE NXREAD
2590            ;
2600 07A3 A208  READD  LDX #8     ; there are 8 pages to load
2610 07A5 A000         LDY #0
2620            ;
2630 07A7 AD10E0 READLP LDA DACIA  ; receive data reg full?
2640 07AA 4A           LSR A
2650 07AB 90FA         BCC READLP ; if not, then wait
2660 07AD AD11E0       LDA DACIA  ; else load data
2670 07B0 2C10E0       BIT CACIA  ; and test if parity error
2680 07B3 700E         BVS NXREAD ; try again
2690 07B5 91FE         STA (MEMLO),Y ; store data in BUFFER
2700 07B7 C8           INY        ; next byte
2710 07B8 D0ED         BNE READLP
2720 07BA E6FF         INC MEMHI
2730 07BC CA           DEX        ; next page read
2740 07BD D0EB         BNE READLP
2750 07BF 20DE06       JSR UNLDHD ; unload head & ret Z=1
2760 07C2 60           RTS        ; ready, ret with Z=1
2770            ;
2780 07C3 C6FD  NXREAD DEC ATTMPT
2790 07C5 D0A7         BNE READA
2800 07C7 C6FB         DEC IOSTPS
2810 07C9 F009         BEQ RDERR
2820 07CB 20A706       JSR STPIN
2830 07CE 20AE06       JSR STPOUT
2840 07D1 4C6A07       JMP READA
2850            ;
2860 07D4 20DE06 RDERR JSR UNLDHD
2870            ;
2880 07D7 0901  ERR    ORA #1     ; ret with Z=0: error
2890 07D9 60           RTS
2900            ;
2910            ;
2920            ; TRKBEG : init ACIA, load head & wait for index pu
2930            ;
2940 07DA 20CB06 TRKBEG JSR LDHEAD
```

```
2950            ;
2960 07DD AD00E0  INDST  LDA FDRA    ; wait till index start
2970 07E0 30FB           BMI INDST
2980 07E2 AD00E0  INDEND LDA FDRA    ; wait till index end
2990 07E5 10FB           BPL INDEND
3000            ;
3010 07E7 A903    INACIA LDA #3      ; init the ACIA
3020 07E9 8D10E0         STA CACIA   ; master reset
3030 07EC A958           LDA #%01011000
3040 07EE 8D10E0         STA CACIA
3050 07F1 60             RTS
3060            ;
3070            ;
3080            ; RWBEG : start read/write access
3090            ;
3100 07F2 209A07  RWBEG  JSR TRKBEG  ; init ACIA etc
3110 07F5 202908  HEADA  JSR RBYTET  ; read byte & test INDEX
3120 07F8 D01A           BNE RET
3130 07FA C943    HEADB  CMP #$43    ; track header 1st byte
3140 07FC D0F7           BNE HEADA
3150 07FE 202908         JSR RBYTET
3160 0801 D011           BNE RET
3170 0803 C957           CMP #$57   ; track header 2nd byte
3180 0805 D0F3           BNE HEADB
3190 0807 2001F5         JSR RBYTE
3200 080A CD5D23         CMP TRACK  ; 3rd byte
3210 080D D0EB           BNE HEADB
3220 080F 2001F5         JSR RBYTE
3230 0812 C958           CMP #$58   ; 4th byte
3240            ;
3250 0814 60      RET    RTS
3260            ;
3270 0815 202908  HEADC  JSR RBYTET
3280 0818 D0FA           BNE RET
3290 081A C976    HEADD  CMP #$76    ; sector header 1st byte
3300 081C 2001F5         JSR RBYTE
3310 081F C901           CMP #$01   ; 2nd byte (sector nr)
3320 0821 D0F7           BNE HEADD
3330 0823 2001F5         JSR RBYTE
3340 0826 C908           CMP #$8    ; 3rd byte (sector length)
3350 0828 60             RTS        ; all right, ret with Z=1
3360            ;
3370            ;
3380            ; RBYTET : read byte, error exit if INDEX=0
3390            ;
3400 0829 AD00E0  RBYTET LDA FDRA
3410 082C 10A9           BPL ERR    ; error if INDEX=0
3420 082E AD10E0         LDA CACIA  ; receive data reg full?
3430 0831 4A             LSR A
3440 0832 90F5           BCC RBYTET ; wait, if not
3450 0834 AD11E0         LDA DACIA
3460 0837 2C3B08         BIT ZERO   ; set Z flag
3470 083A 60             RTS        ; Z=1: ok   Z=0: error
3480            ;
3490 083B 00      ZERO   .BYTE 0
3500            ;
3510            ;
3520            ; WRITE : write memory from buffer onto disk
3530            ;
3540 083C AD5C23  WRITE  LDA DRIVE
3550 083F C904           CMP #4
3560 0841 9003           BCC WRITEO
3570 0843 4C09D8         JMP FWRITE
3580            ;
3590 0846 A920    WRITEO LDA #%00100000
3600 0848 2C00E0         BIT FDRA   ; is track write protected?
3610 084B F08A           BEQ ERR
3620 084D A904           LDA #4     ; max 4 write attempts
3630 084F 85FB           STA IOSTPS
3640 0851 A902    WRITEA LDA #2      ; max 2 reread attempts
3650 0853 85FD           STA ATTMPT
3660            ;
3670 0855 A900    WRITEB LDA #BLOCK
3680 0857 85FE           STA MEMLO
3690 0859 A924           LDA #BLOCK/256
3700 085B 85FF           STA MEMHI
3710 085D A908           LDA #8
3720 085F 85F9           STA PAGES
3730 0861 AD5D23         LDA TRACK  ; write on track zero?
3740 0864 D052           BNE WRITEC
3750            ;
3760 0866 20CB06         JSR LDHEAD
3770 0869 20E707         JSR INACIA ; init the ACIA
3780 086C AD00E0  YINEND LDA FDRA
3790 086F 10FB           BPL YINEND ; wait till index end
3800 0871 AD00E0  YINDST LDA FDRA
3810 0874 30FB           BMI YINDST ; wait for index start
3820 0876 A9FE           LDA #$FE
3830 0878 2D02E0         AND FDRB   ; clear WRITE
3840 087B 8D02E0         STA FDRB
3850 087E AD00E0  YINDEX LDA FDRA
3860 0881 10FB           BPL YINDEX ; wait till index end
3870 0883 A208           LDX #8     ; (16)
3880 0885 20F6F4         JSR FDELAY ; delay 10 ms
3890 0888 A202           LDX #2
3900 088A 205409         JSR WRBYTE ; put load vector on disk
3910 088D A200           LDX #0
3920 088F 205409         JSR WRBYTE
3930 0892 A208           LDX #8     ; sector length
3940 0894 205409         JSR WRBYTE
3950 0897 A000           LDY #0
3960            ;
3970 0899 B1FE    YWRLOP LDA (MEMLO),Y
3980 089B AA             TAX
3990 089C 205409         JSR WRBYTE
4000 089F C8             INY
4010 08A0 D0F7           BNE YWRLOP
4020 08A2 E6FF           INC MEMHI
4030 08A4 C6F9           DEC PAGES
4040 08A6 D0F1           BNE YWRLOP
4050 08A8 AD00E0  ENDTRZ LDA FDRA
4060 08AB 30FB           BMI ENDTRZ
4070 08AD A901           LDA #1
4080 08AF 0D02E0         ORA FDRB   ; set WRITE
4090 08B2 8D02E0         STA FDRB
4100 08B5 4C6607         JMP READD  ; check track
4110            ;
4120 08B8 20F207  WRITEC JSR RWBEG  ; load head, wait until index
4130                                ; pulse over and read header
4140 08BB A24E           LDX #78    ; (157) delay 400 us
4150 08BD 205F09         JSR DELX
4160 08C0 A9FE           LDA #$FE
4170 08C2 2D02E0         AND FDRB   ; clear WRITE
4180 08C5 8D02E0         STA FDRB
4190 08C8 A212           LDX #18    ; (37) delay 100 us
4200 08CA 205F09         JSR DELX
4210 08CD A900           LDA #BLOCK
4220 08CF 85FE           STA MEMLO
4230 08D1 A924           LDA #BLOCK/256
4240 08D3 85FF           STA MEMHI
4250 08D5 A908           LDA #8     ; there are 8 pages to be
4260 08D7 85F9           STA PAGES  ; written on disk
4270 08D9 A276           LDX #$76   ; sector header 1st byte
4280 08DB 205409         JSR WRBYTE
4290 08DE A201           LDX #1     ; 2nd byte (sector nr)
4300 08E0 205409         JSR WRBYTE
4310 08E3 A208           LDX #8     ; 3rd byte (sector length)
4320 08E5 205409         JSR WRBYTE
4330 08E8 A000           LDY #0
4340            ;
4350 08EA B1FE    WRITLP LDA (MEMLO),Y
4360 08EC AA             TAX
4370 08ED 205409         JSR WRBYTE
4380 08F0 C8             INY
4390 08F1 D0F7           BNE WRITLP
4400 08F3 E6FF           INC MEMHI
4410 08F5 C6F9           DEC PAGES
4420 08F7 D0F1           BNE WRITLP
4430 08F9 A247           LDX #$47   ; sector trailer 1st byte
4440 08FB 205409         JSR WRBYTE
4450 08FE A253           LDX #$53   ; sector trailer 2nd byte
4460 0900 205409         JSR WRBYTE
4470 0903 A212           LDX #18    ; (37) delay 100 us
4480 0905 205F09         JSR DELX
4490 0908 A901           LDA #1
4500 090A 0D02E0         ORA FDRB   ; set WRITE bit
4510 090D 8D02E0         STA FDRB
4520            ;
4530 0910 A924    CHECK  LDA #BLOCK/256
4540 0912 85FF           STA MEMHI
4550 0914 20F207         JSR RWBEG  ; read track header
4560 0917 201508         JSR HEADC  ; read sector header
4570 091A D021           BNE NXCHEK
4580 091C A208           LDX #8     ; there are 8 pages to check
```

```
4590 091E A000          LDY #0
4600            ;
4610 0920 AD10E0  CHEKLP LDA CACIA   ; receive data reg full?
4620 0923 4A            LSR A
4630 0924 90FA          BCC CHEKLP
4640 0926 AD11E0        LDA DACIA
4650 0929 2C10E0        BIT CACIA    ; parity error?
4660 092C 700F          BVS NXCHEK
4670 092E D1FE          CMP (MEMLO),Y
4680 0930 D00B          BNE NXCHEK
4690 0932 C8            INY
4700 0933 D0EB          BNE CHEKLP
4710 0935 E6FF          INC MEMHI
4720 0937 CA            DEX
4730 0938 D0E6          BNE CHEKLP
4740 093A 4CDE06        JMP UNLDHD   ; unload head & ret Z=1
4750            ;
4760 093D C6FD    NXCHEK DEC ATTMPT
4770 093F D0CF          BNE CHECK
4780 0941 C6FB          DEC IOSTPS
4790 0943 F009          BEQ WRERR
4800 0945 20A706        JSR STPIN
4810 0948 20AE06        JSR STPOUT
4820 094B 4C5108        JMP WRITEA
4830            ;
4840 094E 20DE06  WRERR  JSR UNLDHD
4850 0951 0901          ORA #1
4860 0953 60            RTS          ; Z=1: ok   Z=0: error
4870            ;
4880            ;
4890            ; WRBYTE : write byte in X on disk
4900            ;
4910 0954 AD10E0  WRBYTE LDA CACIA   ; transmit data reg empty?
4920 0957 4A            LSR A
4930 0958 4A            LSR A
4940 0959 90F9          BCC WRBYTE
4950 095B BE11E0        STX DACIA    ; write data
4960 095E 60            RTS
4970            ;
4980            ;
4990            ; DELX : a JSR needs 5+X+14 us @ 1 MHz
5000            ;
5010 095F CA      DELX   DEX
5020 0960 D0FD          BNE DELX
5030 0962 60            RTS
5040            ;
5050            ; INITRK : init one track on disk
5060            ;
5070 0963 AD5C23  INITRK LDA DRIVE   ; get drive number
5080 0966 C904          CMP #4
5090 0968 9003          BCC INITRA   ; then o.k.
5100 096A 4C0CD8        JMP FINTRK   ; else enter 1797 driver
5110            ;
5120 096D A507    INITRA LDA IOTRK
5130 096F 20EF06        JSR SETTK    ; set track
5140 0972 D053          BNE REINTR   ; error
5150 0974 A920          LDA #$20
5160 0976 2D00E0        AND FDRA     ; disk write protected?
5170 0979 4920          EOR #$20     ; complement bit
5180 097B D04A          BNE REINTR   ; error
5190 097D 78            SEI          ; disable IRQ
5200 097E 20DA07        JSR TRKBEG   ; init ACIA, load head
5210                                 ; & wait for index pulse
5220 0981 A9FE          LDA #$FE
5230 0983 2D02E0        AND FDRB     ; clear WRITE
5240 0986 8D02E0        STA FDRB
5250 0989 A208          LDX #8       ; (17)
5260 098B 20F6F4        JSR FDELAY   ; delay approx. 10 ms
5270 098E A243          LDX #$43     ; track header 1st byte
5280 0990 205409        JSR WRBYTE
5290 0993 A257          LDX #$57     ; track header 2nd byte
5300 0995 205409        JSR WRBYTE
5310 0998 AE5D23        LDX TRACK    ; track header 3rd byte
5320 099B 205409        JSR WRBYTE
5330 099E A25B          LDX #$5B     ; track header 4th byte
5340 09A0 205409        JSR WRBYTE
5350 09A3 A924          LDA #BLOCK/256
5360 09A5 8501          STA POINTH
5370 09A7 A900          LDA #BLOCK
5380 09A9 8500          STA POINTL
5390 09AB A000          LDY #0
5400 09AD A208          LDX #2048/256
5410 09AF A9E5          LDA #$E5
5420            ;
5430            ; Fill 2 K of BLOCK with $E5 and write it on disk
5440            ;
5450 09B1 9100    INTRLP STA (POINTL),Y
5460 09B3 C8            INY
5470 09B4 D0FB          BNE INTRLP
5480 09B6 E601          INC POINTH
5490 09B8 CA            DEX
5500 09B9 D0F6          BNE INTRLP
5510 09BB A901          LDA #1
5520 09BD 0D02E0        ORA FDRB
5530 09C0 8D02E0        STA FDRB
5540 09C3 203C08        JSR WRITE
5550 09C6 58            CLI          ; enable IRQ
5560            ;
5570 09C7 60      REINTR RTS
5580            ;
5590            ;
5600 09C8 00      CNTR   .BYTE 0
5610            ;
5615            ;
5620            ; COUNT : get time from index pulse to index pulse
5630            ;
5640 09C9 A000    COUNT  LDY #0      ; reset lowest counter
5650 09CB 20DA07         JSR TRKBEG  ; load head & wait index over
5660            ;
5670 09CE C8      COUNTA INY         ; increment the counter
5680 09CF D003           BNE COUNTB
5690 09D1 EEC809         INC CNTR
5700            ;
5710 09D4 AD00E0  COUNTB LDA FDRA    ; get floppy signals
5720 09D7 30F5           BMI COUNTA
5730 09D9 202E06         JSR UNLDHD
5740 09DC ADC809         LDA CNTR    ; return counter
5750 09DF 60             RTS
5760            ;
5770            ;
5780            ; ADJTIM : adjust timing for 2 MHz CPU
5790            ;
5800 09E0 A920    ADJTIM LDA #32
5810 09E2 8DC706         STA STPDEL+1
5820 09E5 A910           LDA #16
5830 09E7 8D8408         STA WINDEX+6
5840 09EA A99D           LDA #157
5850 09EC 8DBC08         STA WRITEC+4
5860 09EF A925           LDA #37
5870 09F1 8DC908         STA WRITEC+17
5880 09F4 8D0409         STA WRITLP+26
5890 09F7 A911           LDA #17
5900 09F9 8D8A09         STA INITRA+29
5910 09FC 60             RTS
5920            ;
5930            ; I/O Routines for 2nd Floppy Controller
5935            ;                 and RAM Disk
5940            ;
5950 D800=        FSETTK = $D800
5960 D803=        FSTDRV = $D803
5970 D806=        FREAD  = $D806
5980 D809=        FWRITE = $D809
5990 D80C=        FINTRK = $D80C

'ic ,01
```



EC-65: CP/M-Software

Ein paar schnuckelige CP/M-Programme für den EC-65

Frank Schmidt

Leider kann der EC-65 zur Zeit kein einziges der sonst auf dem Markt vorhandenen CP/M-Diskettenformate lesen. Leider können aber andere Rechner auch das vom EC-65 benutzte OHIO-Format nicht lesen. Denn sonst könnten sich alle CP/M-Freaks die Programme kostenlos beim Software-Verteiler besorgen, die wir hier vorstellen...

Als vorläufiges Trostpflaster für diejenigen, die sich die Z80-Karte für den EC-65 zulegen wollen, bieten wir über den Softwareverteiler ein paar ganz besonders interessante Programme an. Es handelt sich dabei ausschließlich um Public-Domain-Software: Programme, die von Computer-Freaks in der ganzen Welt entwickelt und von der SIG/M User Group zusammengestellt wurden. Wir haben für Sie die interessantesten ausgewählt und stellen sie hier kurz vor.

MONITOR

Der Monitor ist das einzige Programm, das nicht über den Softwareverteiler zu bekommen ist. Er sitzt nämlich schon fix und fertig im EPROM auf der Z80-Karte und wird ganz einfach mit dem ZCPR2-Kommando "JUMP F000H" gestartet. Ist das geschehen, so kann man sich eine Befehlsübersicht ausgeben lassen, indem man "T <CR>" eintippt. Bild 1 gibt einen Überblick: Das Kommando "DF000 F7FF <CR>" listet beispielsweise einen Hex-Dump des entsprechenden Speicherinhalts auf dem Bildschirm.

Mit Ctrl-C kommt man wieder in die ZCPR2-Kommandoebene. Dabei wird wie gewohnt ein Warmstart ausgelöst. Mit dem Kommando "B <CR>" erreichen sie den gleichen Zweck auf eine etwas feinere Art: Sie landen ohne Warmstart im ZCPR2-Kernel.

Etwas Vorsicht ist beim Umgang mit den Befehlen zur Port-Ein/Ausgabe geboten. Da beim EC-65 die gesamte Ein/Ausgabe über nur zwei Ports abläuft, kann ein "Querschläger", ein Schreib-oder Lesebefehl zuviel, ohne weiteres das ganze System zum Absturz oder zum Hängen bringen.

Das Monitorprogramm benutzt übrigens keine BDOS- oder sonst irgendwelche CP/M-Funktionen. Deshalb ist es nicht überflüssig, kurz auf die Bedeutung der verschiedenen Steuerzeichen hinzuweisen.

Ctrl-S hält - wie bei CP/M - die Bildschirm-Ausgabe an. Bei der nächsten Eingabe dieses Steuerzeichens wird sie fortgesetzt.

Auch Ctrl-X stoppt die Ausgabe. Hier kehrt das Programm allerdings sofort wieder in den Kommando-Modus zurück. Mit Ctrl-X beendet man außerdem die Eingabe von ASCII-Zeichen bei der Ausführung des "AL"-Befehls.

Ctrl-H (Backspace) und Rubout werden gleich behandelt. In beiden Fällen wird das zuletzt eingegebene Zeichen gelöscht.

Die ausführliche Dokumentation des Monitors (mal wieder in Englisch) findet sich auf der CP/M-Diskette Nr. 2 (= EC-65-Disk 41), die man beim Softwareverteiler bekommen kann. Legt man diese in Laufwerk B, so kommt man mit "TYPE B:MONITOR.DOC" an die gewünschten Informationen.

1

```
->T

ASCII LOAD, DUMP -->A[ L  D  S ] [1ST ADR] [2ND ADR]
      or SEARCH
BACK GENTLY      -->B
CALL SUBROUTINE  -->C[ADR]
DUMP MEMORY      -->D[1ST ADR] [2ND ADR]
FILL MEMORY      -->F[1ST ADR] [2ND ADR] [HEX BYTE]
                 -->F[1ST ADR] [2ND ADR] '[ASCII]
GO ANYWHERE      -->G[ADR]
HEX MATH + -     -->H[1ST HEX] [2ND HEX]
INPUT FROM PORT  -->I[PORT NUMBER]
LOAD MEMORY      -->L[START ADR]
                    ADR BYTE [NEW BYTE]   ^X HALT
MOVE MEMORY      -->M[1ST ADR] [2ND ADR] [NEW START]
OUTPUT TO PORT   -->O[PORT NUMBER] [BYTE]
PRINTER TOGGLE   -->P
REPLACE MEMORY   -->R[1ST ADR] [2ND ADR] [TARGET] [BYTE]
SEARCH FOR BYTE  -->S[1ST ADR] [2ND ADR] [BYTE1] [BYTE2]
COMMAND TABLE    -->T
COMPARE BLOCKS   -->V[1ST ADR] [2ND ADR] [1ST ADR BLOCK2]
DISP STACK POINT -->X


->
```

Bild 1. Diese Befehle versteht das im EPROM auf der Z80-Platine beheimatete Monitor-Programm. Wer mal nicht mehr weiter weiß, kann die Liste mit dem Kommando "T" abrufen.

FORTH

"Schon wieder?" fragen Sie? "FORTH gibt es doch schon längst für den EC-65!" Ganz recht, schon wieder. Aber diesmal mit Pfiff! Bild 2 zeigt den Befehlsvorrat des 24 (!) KByte großen Compilers. Ganz schön umfangreich, nicht wahr? Der Compiler enthält neben anderen Annehmlichkeiten einen eigenen Editor, einen kompletten 8080-Assembler, einen FORTH Run-Time-Debugger, einen Decompiler, einen Multitasker (!) und (für die Unverbesserlichen) einen nach Belieben zuladbaren BASIC-Compiler - ein recht kurzes FORTH-Prgrämmchen übrigens, das in sehr beeindruckender Weise die Möglichkeiten dieses Systems demonstriert. Bild 3 zeigt ein solches BASIC-Programm, wie es auf der FORTH-Diskette zu finden ist.

Für den ersten Kontakt reicht es aus, wenn Sie die FORTH-Diskette in Laufwerk B einlegen und "B:F83" eintippen. Nach dem Laden meldet sich das FORTH und erwartet eine Eingabe. Wer sich schon ein bißchen mit der Programmiersprache auskennt, der tippt natürlich gleich "VLIST" ein und - erhält prompt eine Fehlermeldung (so ging es auch dem geplagten Autor dieses Artikels). Will man den Befehlsvorrat ausgeben lassen, so muß man den Befehl "WORDS" benutzen. Das liegt daran, daß es sich hier nicht um einen Compiler nach dem FIG-FORTH-, sondern nach dem FORTH-83-Standard handelt. Nun können Sie auch mal ein bißchen herumspielen. Das Kommando "BYE" bringt Sie dann zurück ins Betriebssystem.

Wenn Sie richtig in dieses FORTH einsteigen wollen, müssen Sie zuerst etwas Vorarbeit leisten. Denn die meisten Files auf

2

```
EMPTY   MARK   HELLO   BACKGROUND:   ACTIVATE   SET-TASK   TASK:
RESUME   DEBUG   LISTING   SHOW   (SEMIT)   (PAGE)   FORM-FEED   PAGE
#PAGE   LOGO   L/PAGE   FOOTING   INIT-PR   EPSON   SEE   (SEE)
ASSOCIATIVE:   CASE:   MAP   OUT   DL   DU   DUMP   .HEAD   ?.A   ?.N
DLN   EMIT.   D.2   .2   A   SHADOW   (WHERE)   FIX   EDIT   ED   DONE
EDITOR   DARK   AT   -LINE   BLOT   REPLACE   INSERT   DELETE   SEARCH
SCAN-1ST   FOUND   TO   CONVEY   (CONVEY)   .TO   HOP   CONVEY-COPY
U/D   HOPPED   VIEW   §VIEW   COPY   (COPY)   ESTABLISH   L   B   N
::   MANY   TIMES   #TIMES   WORDS   LARGEST   IND   INDEX   .LINEO
TRIAD   LIST   .SCR   ?CR   ?LINE   RMARGIN   LMARGIN   HIDDEN   O<=
O>=   >=   <=   U>=   U<=   MS   FUDGE   PC!   PC§   MULTI   SINGLE
STOP   WAKE   SLEEP   !LINK   §LINK   LOCAL   RESTART   (PAUSE)
UNBUG   BUG   DOES?   DOES-SIZE   DOES-OP   LABEL   UTILITY.BLK
CPU8080.BLK   EXTEND80.BLK   KERNEL80.BLK   VIEWS   VIEW-FILES
SAVE-SYSTEM   FROM   OPEN   DEFINE   B:   A:   DRIVE?   DIR
CREATE-FILE   MORE   ROOT   -->   +THRU   THRU   ?ENOUGH   ?   (S   Ö
L/SCR   C/L   RECURSE   DUMP   .ID   .S   DEPTH   BYE   START   OK
INITIAL   COLD   WARM   BOOT   QUIT   RUN   IS   (IS)   >IS   USER
#USER   CODE   AVOC   2VARIABLE   2CONSTANT   DEFINITIONS   VOCABULARY
DEFER   VARIABLE   CONSTANT   RECURSIVE   ;   :   Ü   Ä   DOES>
;CODE   (;CODE)   ;USES   ASSEMBLER   (;USES)   REVEAL   HIDE   ?CSP
!CSP   CREATE   "CREATE   ,VIEW   WHILE   ELSE   IF   REPEAT   AGAIN
UNTIL   +LOOP   LOOP   ?DO   DO   THEN   BEGIN   ?LEAVE   LEAVE
?<RESOLVE   ?<MARK   ?>RESOLVE   ?>MARK   <RESOLVE   <MARK   >RESOLVE
>MARK   ?CONDITION   ABORT   ABORT"   (ABORT")   (?ERROR)   ?ERROR
WHERE   FORGET   (FORGET)   TRIM   FENCE   "   ."   ,"   (.")   (")
ÄCOMPILEÜ   Ä'Ü   '   ?MISSING   CRASH   CONTROL   ASCII   DLITERAL
LITERAL   IMMEDIATE   COMPILE   EVEN   ALIGN   C,   ,   ALLOT
INTERPRET   STATUS   ?STACK   DEFINED   ?UPPERCASE   FIND   #THREADS
(FIND)   HASH   VIEW>   >VIEW   >LINK   >NAME   >BODY   LINK>   NAME>
BODY>   L>NAME   N>LINK   FORTH-83   DONE?   TRAVERSE   ÖS   (   .(
>TYPE   WORD   'WORD   PARSE   PARSE-WORD   SOURCE   (SOURCE)   PLACE
/STRING   SCAN   SKIP   D.R   D.   (D.)   UD.R   UD.   (UD.)   .R   .
(.)   U.R   U.   (U.)   OCTAL   DECIMAL   HEX   #S   #   SIGN   #>
<#   HOLD   NUMBER   (NUMBER)   NUMBER?   (NUMBER?)   CONVERT
DOUBLE?   DIGIT   LOAD   (LOAD)   DEFAULT   VIEW#   FLUSH
SAVE-BUFFERS   EMPTY-BUFFERS   IN-BLOCK   BLOCK   (BLOCK)   BUFFER
(BUFFER)   MISSING   DISCARD   UPDATE   ABSENT?   LATEST?   CAPACITY
DOS   SWITCH   FILE?   .FILE   WRITE-BLOCK   READ-BLOCK   >UPDATE
BUFFER#   >END   >BUFFERS   INIT-RO   FIRST   >SIZE   LIMIT
DISK-ERROR   B/FCB   REC/BLK   B/REC   B/BUF   #BUFFERS   QUERY   TIB
EXPECT   CC-FORTH   CC   DEL-IN   CHAR   (CHAR)   CR-IN   P-IN
RES-IN   BACK-UP   (DEL-IN)   BS-IN   BEEP   BACKSPACES   SPACES
SPACE   TYPE   CRLF   (EMIT)   (PRINT)   PR-STAT   CR   KEY   KEY?
(CONSOLE)   (KEY)   (KEY?)   BIOS   BDOS   COMPARE   CAPS-COMP   COMP
-TRAILING   PAD   HERE   UPPER   UPC   MOVE   LENGTH   COUNT   BLANK
ERASE   FILL   CAPS   BELL   BS   BL   END?   #TIB   SPAN   >IN   BLK
VOC-LINK   WIDTH   'TIB   CONTEXT   #VOCS   CURRENT   CSP   LAST   R#
DPL   WARNING   STATE   PRIOR   SCR   EMIT   PRINTING   IN-FILE   FILE
HLD   BASE   OFFSET   #LINE   #OUT   DP   RPO   SPO   LINK   ENTRY
TOS   */   */MOD   MOD   /   /MOD   *   MU/MOD   M/MOD   *D   DMAX
DMIN   D>   D<   DU<   D=   DO=   ?DNEGATE   D-   D2/   D2*   DABS
S>D   DNEGATE   D+   2ROT   4DUP   3DUP   2OVER   2SWAP   2DUP   2DROP
2!   2§   WITHIN   BETWEEN   MAX   MIN   >   <   U>   U<   ?NEGATE
<>   =   O<>   O>   O<   O=   UM/MOD   U*D   UM*   2-   1-   2+   1+
8*   U2/   2/   2*   3   2   1   0   +!   ABS   -   NEGATE   +   OFF
ON   CTOGGLE   CRESET   CSET   FALSE   TRUE   NOT   XOR   OR   AND
ROLL   PICK   R§   >R   R>   ?DUP   FLIP   -ROT   ROT   NIP   TUCK
OVER   SWAP   DUP   DROP   RP!   RP§   SP!   SP§   CMOVE>   CMOVE   C!
C§   !   §   (?LEAVE)   (LEAVE)   J   I   PAUSE   NOOP   GO   PERFORM
EXECUTE   >NEXT   BOUNDS   (?DO)   (DO)   (+LOOP)   (LOOP)   ?BRANCH
BRANCH   (LIT)   UP   UNNEST   EXIT   RP   FORTH
```

Bild 2. Das sind die "WORDS", die der 24 KByte große FORTH-Compiler direkt nach dem "Einschalten" versteht. Neben Editor, Assembler, Debugger und Decompiler ist sogar ein Multitasker mit eingebaut.

3

```
F>secrtary <CR>

CP/M SECRTARY REL 9.76     820320A
USED: 0   AVAILABLE: 36775
CONFIGURE (Y/N)? Y

TYPE CHAR USED FOR BACKSPACE <BS>
STOP AT END OF EACH PAGE (Y/N)? N
LINES PER SCREEN OR ZERO? 24 <CR>
INPUT LINE LENGTH? 80 <CR>
LINE WRAP AROUND ON INPUT (Y/N)? Y
UNDERLINE USING BS OR CR (B/C)? C


IF YOU WANT TO SAVE THIS CONFIGURATION,
"QUIT" AND ENTER "SAVE 48 SECRTARY.COM"


CP/M SECRTARY REL 9.76     820320A
USED: 0   AVAILABLE: 36775
#
```

```
Scr # 10          BASIC.BLK
  0 ( basic: branching demo )
  1 INTEGER J   INTEGER K
  2
  3 : RUN   BASIC
  4 10 FOR K = 1 TO 15 STEP 3
  5 15 LET J = J + K
  6 20 IF K >= 8 THEN 35
  7 25 PRINT K
  8 30 GOTO 40
  9 35 PRINT K , J , " SUM "
 10 40 NEXT K
 11 50 PRINT " DONE  "
 12 80 END
 13
 14 RUN
 15
```

4

```
'65-CP/M.003    0k  --  EC-65 CP/M Disketten Nummer
BASIC   .BQK    5k  --  BASIC Compiler in FORTH geschrieben
CPU8080 .BQK   10k  --  Assembler, Run-Time-Debugger, Multitasker (D)
EXTEND80.BQK    9k  --  FORTH-CP/M Interface (D)
F83     .COM   24k  --  FORTH-83 Compiler
F83-FIXS.TQT    4k  --  Updates gegenüber älterer Version (D)
HUFFMAN .BQK   13k  --  FORTH-Programm zum Packen von Dateien - nicht
                        kompatibel mit USQ / SHOW
META80  .BQK   13k  --  FORTH Metacompiler (D)
README  .8Q    13k  --  FORTH Kurzdokumentation (D)
SHOW    .COM    3k  --  Utility zum Lesen von gepackten Dateien
USQ     .COM    2k  --  übersetzt gepackte Files ins Original
UTILITY .BQK   37k  --  verschiedene FORTH-Utilities (D)

12 files occupy  133K bytes
```

Bild 3. So sieht ein Programm für den in FORTH geschriebenen BASIC-Compiler aus. Der Compiler wird von FORTH aus mit "OPEN BASIC.BLK <CR>" und "OK <CR>" geladen.

Bild 4. Diese Dateien befinden sich auf der FORTH-Diskette. Sind sie mit Extensionen wie ".TQT" oder ".BQK" gekennzeichnet, so handelt es sich um "squeezed", um gepackt abgespeicherte Files, die erst mit der Utility "USQ.COM" expandiert werden müssen. Befindet sich hinter dem Kommentar zu einem Dateinamen ein "(D)", so bedeutet das, daß es sich hierbei um eine Datei mit ausschließlich dokumentarischem Wert handelt.

der Diskette lassen sich so ohne weiteres nicht nutzen. Sie sind in einem "gepackten" Format dort abgespeichert. Nur so war es uns möglich, all die Bytes unterzubringen. Um diese Dateien wieder expandieren zu können, ist auf der Diskette die Utility "USQ.COM" untergebracht, die sie am besten so benutzen:

- FORTH-Diskette in Laufwerk B einlegen
- mit "E: <CR>" auf die RAM-Disk umschalten
- "DIR B: <CR>" eingeben, es erscheint das Inhaltsverzeichnis von Laufwerk B
- Files aussuchen, die expandiert werden sollen. Alle gepackt abgespeicherten Dateien haben ein "Q" in der Extension des Dateinamens (z. B. BASIC.BQK).
- Mit "B:USQ B:fn.ext <CR>" werden die Dateien expandiert, auf die der angegebene Name zutrifft (auch die Wildcards "*" und "?" sind erlaubt). Sie werden dann auf Laufwerk E abgespeichert.
- Jetzt sollte man die expandierten Dateien noch auf einer "richtigen" Diskette sichern, um immer Zugriff darauf zu haben.

Auf Dateien mit der Extension ".BLK" haben Sie nur von FORTH aus einen vernünftigen Zugriff. Hier sind die verschiedenen Screens abgespeichert, über die FORTH das Handling des Massenspeichers abwickelt.

So kann man FORTH auch mit "F83 E:BASIC.BLK" starten. Dann hat man Zugriff auf alle Screens, die unter dieser Sammelbezeichnung in der RAM-Disk abgelegt sind. Mit "1 LIST" kann man sich den Inhalt des ersten dieser Screens ausgeben lassen, mit "1 7 INDEX" die Überschriften der Screens 1..7. Auf diese Weise kann man sich auch fast den gesamten Source des FORTH sowie eine ausführliche Dokumentation beschaffen. Bild 4 erläutert die Bedeutung der verschiedenen auf der Diskette vorhandenen Dateien.

SECRETARY

Dieses Wort aus dem angelsächsischen Sprachgebrauch bedeutet - wer hätte das gedacht - so viel wie "Sekretär". Die Bedeutung des Programmes, das sich dahinter verbirgt, ist damit nur grob umrissen. Es handelt sich nämlich um ein relativ leistungsfähiges Textverarbeitungssystem auf der Basis eines (leider) zeilenorientierten Editors. Es kann sowohl zum Erstellen auch umfangreicher Texte als auch zum Entwickeln von Programm-Quelltexten benutzt werden.

Bild 5. Der Initialisierungsdialog für den Textverarbeiter "SECRETARY". Er ist für einen Matrixdrucker mit Endlospapier ausgelegt. Benutzen Sie Einzelpapier, so müssen Sie die Frage "STOP AT ... ?" unbedingt mit "Y" beantworten.

Bild 6. Insgesamt sechs verschiedene CP/M-Disketten kann man zur Zeit vom Software-Verteiler beziehen. Jede enthält etwa 150 KByte Programme und Dokumentation (80-Track-Format). Damit läßt sich schon allerhand anfangen. Wie man an noch mehr CP/M-Software herankommt, verraten wir in den nächsten ec-Heften.
Für EC-65-Benutzer, denen nur 40-Track Laufwerke zur Verfügung stehen, haben wir den Inhalt der sechs Scheiben noch einmal auf nunmehr 11 Disketten verteilt, von denen jede eine Kapazität von etwa 70 KByte besitzt. Das ergibt insgesamt 17 neue EC-65-Disketten, die wir für Sie zusammengestellt haben!

5

```
F>secrtary <CR>

CP/M SECRTARY REL 9.76     820320A
USED: 0   AVAILABLE: 36775
CONFIGURE (Y/N)? Y

TYPE CHAR USED FOR BACKSPACE <BS>
STOP AT END OF EACH PAGE (Y/N)? N
LINES PER SCREEN OR ZERO? 24 <CR>
INPUT LINE LENGTH? 80 <CR>
LINE WRAP AROUND ON INPUT (Y/N)? Y
UNDERLINE USING BS OR CR (B/C)? C


IF YOU WANT TO SAVE THIS CON   ATION,
"QUIT" AND ENTER "SAVE 48 SEC   RY.COM"


CP/M SECRTARY REL 9.76     820320A
USED: 0   AVAILABLE: 36775
#
```

Obwohl es sich um einen zeilenorientierten Editor handelt, gehört der bei Leibe nicht zu den schlechtesten: Es ist an so ziemlich alles gedacht, was man sich wünschen kann.
Wir wollen hier ein paar Features nennen, die das Textverarbeitungssystem kennzeichen:
— Automatischer (und einstellbarer) Zeilenumbruch, der während der Ausgabe des Textes durchgeführt wird.
— Einfügen von Kommentaren, die später nicht gedruckt werden.
— Automatisches Formatieren des Textes: Kopfzeile, Seitennummern, Seitenlänge, Zeilenlänge, Ränder,...
— Einmischen von Daten aus speziell dafür erstellten Files (wie z. B. Adressenlisten). Damit lassen sich Serienbriefe auf einfache Art und Weise erstellen.
Und das ist nur eine Auswahl!

Auf der Diskette ist auch ein komplettes, sehr ausführliches Handbuch zu diesem Programm enthalten (in Englisch). Wenn Sie einen Drucker besitzen, können sie sich dieses insgesamt 38 Seiten starke-Werk folgendermaßen auf's Papier bannen lassen:
SECRETARY-Diskette in Laufwerk B einlegen und das Programm mit "SECRTARY" (die Schreibweise ist hier richtig: Das "E" in der Mitte fehlt, weil CP/M-Dateinamen nur maximal acht Zeichen lang sein dürfen!) und anschließendem Betätigen der RETURN-Taste starten. Es folgt dann ein kurzer Initialisierungsdialog wie in Bild 5. Machen Sie Ihren Drucker bereit und lenken Sie dann die Ausgabe mit dem Befehl "DEVICE L" dorthin um. (Wenn Sie keinen Drucker zur Verfügung haben, unterlassen Sie diese Eingabe. Der Ausdruck erfolgt dann auf der Konsole.) Nun laden Sie mit "LOAD USERMAN0.TXT" den ersten Teil der Dokumentation (das Inhaltsverzeichnis) und schicken es mit dem Kommando "PRINT" zum Drucker. Das Gleiche tun Sie dann mit dem File

6

```
80-Track-Disketten (einseitig)


Disk 40:   ZCPR2-Systemdiskette, Utilities
Disk 41:   ROM-Source, ZCPR2-Source (1), Utilities
Disk 42:   F83 mit Source, Dokumentation ohne KERNEL.BLK
Disk 43:   XLISP mit Source und Dokumentation
Disk 44:   SECRETARY mit Dokumentation, ohne Source
Disk 45:   SECRETARY-, ZCPR2-Source (2) und KERNEL.BLK



40-Track-Disketten (einseitig)


Disk 46:   ZCPR2-Systemdiskette
Disk 47:   Utilities (1)
Disk 48:   Utilities (2)
Disk 49:   ROM-Source, ZCPR2-Source (1)
Disk 50:   F83 mit Dokumentation ohne HUFFMAN.BLK
Disk 51:   F83-Source ohne KERNEL.BLK
Disk 52:   KERNEL.BLK
Disk 53:   XLISP mit Dokumentation
Disk 54:   SECRETARY mit Dokumentation (1)
Disk 55:   SECRETARY-Dokumentation (2)
Disk 56:   SECRETARY- und ZCPR2-Source (2)
```

"USERMAN1.TXT". Es enthält die eigentliche Dokumentation (33 Seiten). Wenn Ihr Drucker schließlich auch diese Arbeit erledigt hat, kommt der letzte Teil (nur noch drei Seiten) dran. Er hat den Namen "USERMAN5.TXT". Die anderen Teile wurden zwischendurch automatisch eingelesen und ausgegeben.
Verlassen wird der Textverarbeiter mit dem "QUIT"-Kommando. "HELP" bietet dagegen nur dann eine echte Hilfe, wenn man das Programm schon ganz gut kennt und nur mal eine Befehlsbezeichnung vergessen hat. Nach der Eingabe des Schlüsselwortes wird nämlich lediglich der gesamte Befehlsvorrat ohne Kommentar aufgelistet.
Der letzte Dokumentationsteil nimmt eine Sonderstellung ein und sollte nicht vernachlässigt werden. Das Programm selbst wurde nämlich eigentlich gar nicht für CP/M geschrieben, aber fast die ganze Dokumentation bezieht sich auf diese "alte" Version. Der letzte Teil dagegen betrifft ausschließlich die CP/M-Version von SECRETARY. Falls also irgendetwas nicht so funktioniert, wie es laut Handbuch sollte, dann finden Sie hier sicherlich die Erklärung.

XLISP

An dieser Stelle sollten sich alle angesprochen fühlen, die sich gern mal ein bißchen mit dem Thema KI (Künstliche Intelligenz) beschäftigen wollen. Der Autor von XLISP - das übrigens in "C" geschrieben ist- hat sich die Sprache LISP, die als Lieblingskind der KI'ler gilt, als Vorbild für diesen Interpreter genommen. Es handelt sich allerdings um eine recht freie Implementation derselben. Dennoch kann man sich damit ganz gut in's listengesteuerte Programmieren einarbeiten. Man sollte sich aber zuerst einmal ein ganz klein wenig mit LISP und dessen Unterschiede zu anderen Programmiersprachen vertraut machen. Denn andernfalls ist der Einstieg in die ansonsten recht detaillierte Dokumentation des Programmes ein Ding der Unmöglichkeit.
Diese Dokumentation können Sie sich mit dem Befehl "TYPE XLISP.DOC" bzw. "LIST XLISP.DOC" ausgeben lassen. Im zweiten Fall wird die Ausgabe zum Drucker gelenkt. Da sich aber auch dieses File in gepacktem Zustand auf der Diskette befindet, muß man wie bei FORTH vorher noch die Utility "USQ.COM" bemühen.
Der Interpreter wird einfach mit dem Kommando "XLISP" gestartet. Soll ein XLISP-Programm gleich mitgestartet werden, so lautet die Eingabe "XLISP fn.ext". Ist die Extension ".LSP", so kann diese beim Aufruf auch entfallen.
LISP-Source-Files können Sie übrigens auch mit SECRETARY erstellen. Bevor Sie das so erstellte Programm übersetzen lassen, müssen Sie es aber abspeichern, und zwar mit "BSAVE" und nicht, wie Sie das bei normalen Textfiles tun, mit "SAVE". Außerdem müssen die Zeilennummern aus Programmen entfernt werden. Das geht ganz einfach, indem Sie auf die entsprechende Frage, die das Programm Ihnen nach der Eingabe von "BSAVE" stellt, mit "Y" für "YES" antworten.

Auch der gesamte Source des Interpreters befindet sich auf der XLISP-Diskette. Er ist aber gleich doppelt verschlüsselt. Und zwar unter dem Namen "XLISP.LQR". Zuerst muß man wieder "USQ.COM" darauf ansetzen. Man erhält das Library-File "XLISP.LBR". Mit der Utility "LU.COM" kann man daraus dann die gewünschten Einzelfiles lösen.

Alle angeführten Programme sind einschließlich Dokumentation (und meist auch Source) beim Software-Verteiler zu bekommen. Bild 6 gibt nähere Auskunft darüber: Da wir EC-65-Fans mit 40-Track-Laufwerken nicht einfach übergehen wollten, haben wir insgesamt 17 (!) neue Disketten für Sie zusammengestellt!
Der Umgang mit den Programmen auf den Disketten mit der kleineren Kapazität unterscheidet sich logischerweise etwas von der hier vorgestellten Bedienung der 160-KByte-Scheiben. Die Unterschiede sind aber nicht so groß, daß sie einer extra Erläuterung bedürften.
Man sollte an dieser Stelle aber nicht verschweigen, daß das Arbeiten unter CP/M mit 40-Track-Disketten doch recht mühsam sein kann. Deshalb ist hier noch einmal der Hinweis angebracht, die Z80-CPU-Karte mit 256 KByte DRAM zu bestücken, um den "Rest" als RAM-Floppy benutzen zu können. Wenn Sie mit dem System wirklich arbeiten wollen, werden Sie dankbar sein für diesen Hinweis!

Und nun wünschen wir Ihnen viel Spaß beim Umgang mit all der neuen Software. Wenn Sie meinen, daß wir eines der Softwarepakete oder auch ZCPR2 selbst mal genauer vorstellen sollten, uns mal intensiver um eine bestimmte Programmiersprache kümmern sollten, oder wenn Sie uns einfach von Ihren (positiven oder negativen) Erfolgen berichen wollen, dann schreiben Sie uns doch! Anregungen und Kritik sind immer herzlich willkommen und helfen letztendlich auch, das Heft so zu gestalten, wie Sie, unsere Leser, es wünschen!

Centronics-Adapter

Gerd Klein

Mit einem "Überspielkabel" und etwas Software kann der "EC-65" als "Daten-Recorder" arbeiten. "Signal-Quelle" können alle Computer sein, die einen "Centronics-Ausgang" haben. An diesem Ausgang (Drucker-Ausgang) wird ein "Centronics-Adapter-Kabel" angeschlossen. Schaltet man den "EC-65" durch das "Daten-Recorder-Programm" auf Aufnahme, werden alle Programme, Texte oder Dateien in den Arbeitsspeicher des "EC-65" überspielt und können dann auf Diskette oder Kassette (Basicode) aufgenommen werden. Weil der "EC-65" anstelle eines Druckers angeschlossen ist, spielt es keine Rolle, mit welchem Betriebssystem der "Signal-Quellen-Computer" arbeitet. Es ist gleichgültig, ob seine CPU eine "6" oder "8" als erste Ziffer trägt und ob sein Datenbus schon 16 bit "breit" ist. Alles was dieser Computer zur Druckerschnittstelle schickt, wird vom Datenrecorder aufgenommen. Auf diese Weise lassen sich sehr einfach Programme "EC-65"-lauffähig machen oder Random-Dateien, die z. B. mit Dbase II erstellt wurden, überspielen. Es werden Ihnen bestimmt noch mehr Nützlichkeiten für den "Daten- Recorder" einfallen. Grund genug, Sie nicht noch länger mit der Beschreibung auf die "Folter" zu spannen.

Die Software

Manche Adapter-Kabel benötigen Elektronik, um zwei Geräte miteinander zu verbinden. Das Centronics-Adapter-Kabel braucht dazu Software. Auf der "EC-65"-Seite des Adapterkabels muß nämlich ein zweiter Centronics- Eingang geschaffen werden. Sehen wir uns dehalb die Programmierung und Beschaltung der beiden VIAs auf der CPU-Karte an:

VIA A Port A = Tastatur-Eingang
VIA A Port B = Centronics-Ausgang
VIA B Port A = Lese-Eingang zur RS-232-Programmierung.
VIA B Port B = Centronics-Ausgang.

Wir entscheiden uns für VIA B Port A, da man bei einem 6522 Port B nicht als "Eingang" programmieren kann und die Tastatur auch weiterhin unentbehrlich bleibt.

Will man einen VIA programmieren, sollte man sich mit den einzelnen Registern dieses Bausteines vertraut machen. Eine ausführliche Beschreibung aller Register und deren Arbeitsweise findet man im "Elektor-VIA-Buch". "Pflichtlektüre" bei solchen Vorhaben.

"Read Handshake" heißt die Operation, wenn der Prozessor über die Portleitungen Daten von einem peripheren Gerät liest und gleichzeitig Steuersignale sendet und empfängt. Genau diese Steuer-

signale sind es, die programmiert werden müssen und uns deshalb im Folgenden beschäftigen. Unser "Signal-Quellen-Computer" setzt Daten auf die Leitungen von Port A und teilt dem "EC-65" über die CA1-Leitung mit, daß gültige Daten auf den Port-Leitungen stehen. Der "EC-65" teilt wiederum dem peripheren Gerät über die CA2-Leitung mit, daß er zur Zeit im Interupt-Mode die Daten liest und solange auch keine neuen Daten gesendet werden dürfen (Input Busy). Sobald das Zeichen im Akku steht, wird das CA1-Flag zurückgesetzt, und der "EC-65" kann das nächste Zeichen empfangen. Dieser Vorgang wiederholt sich, bis alle Daten vom "EC-65" übernommen worden sind.

Folgende Register werden beeinflußt, wenn man mit Read-Handshake arbeiten will:

VIA B Port A DATA DIRECTION R. (VBPADD)
VIA B PERIPHERAL CONTROL R. (VBPCR)
VIA B Port A INPUT DATA R. (VBPAD)
VIA B AUXILIARY CONTROL R. (VBACR)
VIA B INTERRUPT FLAG R. (VBIFR)

Das Bitmuster im Data-Direction-Register bestimmt, ob die Port-A-Leitungen als Eingang oder Ausgang arbeiten. Eine Null in diesem Register programmiert die Leitungen als "Eingang".

Das Peripheral-Control-Register ist in zwei Nibble aufgeteilt. Das untere Nibble ist für Port A und das obere für Port B zuständig. Das jeweils erste Bit eines Nibbles bestimmt, ob CA1 (CB1) mit der negativen oder positiven Flanke getriggert werden soll. Hier muß die negative Flanke programmiert werden, weil das Druckerschnittstellen-Signal "Data Strobe" immer dann von logisch 1 nach Ø springt, wenn die Daten auf den Portleitungen stabil sind. Die drei restlichen Bits der zwei Nibble entscheiden wie CA2 (CB2) arbeiten soll. In unserem Fall soll an CA2 eine logische 1 anliegen, wenn der "EC-65" keine Daten annehmen kann. In dieses Nibble schreiben wir ØE für "CA2-Eingang ist besetzt" und ØC für "CA2-Eingang ist frei", es können Daten gesendet werden.

In das Auxiliary-Control-Register wird eine Ø1 geschrieben. Daduch erreicht man das Zwischenspeichern der Daten im VIA selbst (Latching).

Mit dem Interrupt-Flag-Register stellt man fest, welcher logische Zustand an CA1 vorhanden ist. In unserem Fall lösen wir keinen Interrupt aus, sondern prüfen in einer Programmschleife nach, ob Bit 2 in diesem Register gesetzt ist. Ist es gesetzt, stehen die Daten (ein Zeichen) stabil auf den Portleitungen und können im VBPAD-Register ausgelesen werden.

Der "Signal-Weg"

Das Auslesen eines Zeichens aus dem VBPAD-Register erledigen wir mit einer selbsthergestellten Inputroutine. Sie ist Bestandteil des "Daten-Rekorder- Programms", das in Tabelle 1 abgedruckt ist.

Tabelle 1.

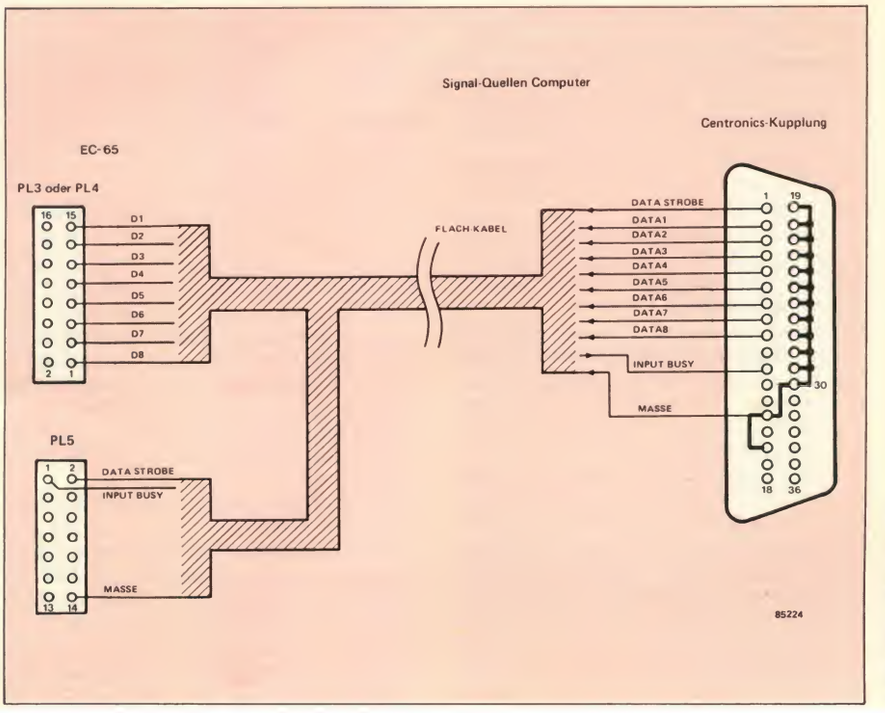
Die Adaptersoftware kann in ein BASIC-Programm eingebunden werden. Dadurch lassen sich ganze Dateien überspielen. Durch den Befehl DISK!"GO D8ØØ" werden die Zeichen bei einem BASIC-INPUT-Befehl nicht von der Tastatur, sondern vom Signalquellen-Computer gelesen. Mit dem Befehl DISK!"IO Ø1" kann die Tastatur wieder eingeschaltet werden.

Die Hardware

Diesmal ist kaum etwas zu der Hardware, zu sagen. Wird die serielle Schnittstelle gleichzeitig benötigt, müssen die entsprechenden Brücken auf der Rückseite der Platine gelegt werden. Man kann aber auch nach dem Booten oder zusammen mit dem Starten der Adaptersoftware den ACIA programmieren, in dem man in das ACIA-Register die entsprechenden Daten schreibt.

Tabelle 1.

```
D800 A9 08     DOSINI  LDAIM $08     Input-Distributor zeigt auf die Nullinput-
D802 8D 21 23          STA   INDST   routiene (Nichtbenutzte BASIC-IN-Routine).
D805 A9 09             LDAIM $09     Output-Distributor zeigt auf Bildschirm
D807 8D 22 23          STA   OUTDST  und parallelen Drucker

D80A A9 01     VIAINI  LDAIM $01     Erlaubt das Zwischenspeichern der Daten
D80C 8D 1B E1          STA   VBACR   im VIA selbst (Latching).
D80F A9 0E             LDAIM $0E     Triggert die neg. Flanke an CA1 und gibt
D811 8D 1C E1          STA   VBPCR   eine logische 0 an CA2 aus (Input Busy).
D814 A9 00             LDAIM $00     Pogrammiert VIA B Port A als
D816 8D 13 E1          STA   VBPADD  "Eingang".
D819 A0 25             LDYIM AUS     Schreibt die Startadresse von der Input-
D81B 88                DEY           routine, die das VBPAD-Register ausliest,
D81C 8C 07 23          STY   NULLIN  in die Distributor-Tabelle (Nullinput).
D81F A0 D8             LDY   AUS /256  Da der Inputdistributor auf diese Adresse
D821 8C 08 23          STY   NULLIN +1 zeigt, (siehe oben) wird bei jedem Input
D824 60                RTS             das VBPAD-Register ausgelesen.

D825 A9 0C     AUS     LDAIM $0C     CA2 erhält eine logische 0, der EC-65 ist
D827 8D 1C E1          STA VBPCR     bereit, Daten zu empfangen.
D82A AD 1D E1          LDA VBIFR     überprüfe, ob Daten anliegen
D82D 29 02             ANDIM $02
D82F F0 F4             BEQ AUS       Warte bis Daten anliegen
D831 A9 0E             LDAIM $0E     CA2 erhält eine logische 1 (INPUT BUSY)
D833 8D 1C E1          STA VBPCR     Eingang ist gesperrt
D836 AD 11 E1          LDA VBPAD     Lade die Daten
D839 29 7F             ANDIM $7F     Erlaube nur ALPHA-NUMERISCHE-Zeichen
D83B 8D 63 23          STA AHOLD     Speicher das Zeichen in AHOLD
D83E 60                RTS
```



Datenbanksysteme für den EC-68

Paul Hogenboom

Das systematische Erfassen, Verarbeiten und Speichern von Daten ist ein Hauptanwendungsgebiet von Computersystemen aller Art. Mit dem Flex-Computer EC68 läßt sich zwar keine Großdatenbank realisieren, aber andererseits ist er zu mehr fähig als nur zur Katelogisierung des häuslichen Bücherbestands oder zum Erfassen einer Schallplattensammlung.

Das Marktangebot ist in der Sparte der Dateiverwaltungs- und Dateimanagement-Systeme sehr vielfältig; kein Wunder, daß dies auch für Flex-Computer gilt. Die Leistungsfähigkeit (und der Preis) der einzelnen Produkte klafft allerdings stark auseinander. Die Angebotspalette reicht von relativ einfachen Systemen, deren Möglichkeiten schnell überschaubar sind, bis zu sehr komplexen und deshalb nicht ganz leicht zu handhabenden Softwarepaketen. Deshalb müssen wir uns an dieser Stelle im wesentlichen auf die Beschreibung eines gängigen Beispiels beschränken.

Wir sprechen hier übrigens mit Absicht von "Datenbank-Systemen und nicht von "Datenbank-Programmen", weil zu einem Datenbank-System eine Vielzahl von Programmen der unterschiedlichsten Art gehören kann. Der Umfang des Programmpakets ist bei einem Datenbanksystem in hohem Maß mitbestimmend für den Grad der Universalität.

Prinzipiell lassen sich die Teilprogramme eines Datenbank-Systems einer der drei folgenden Kategorien zuordnen:

1. Programme, die die Grundvoraussetzungen für die Datenerzeugung schaffen. Unter anderem generieren sie die Eingabemasken auf dem Bildschirm, und sie formatieren auch die Datenfelder auf der Diskette.
2. Editoren, mit denen Datensätze eingegeben, geändert und gegebenenfalls auch in komplizierterer Weise manipuliert werden können.
3. Programme, die den Datenbestand nach den vom Benutzer festgelegten Kriterien verwalten und präsentieren. Sie selektieren beispielsweise einen gewünschten Datensatz und geben ihn auf dem Bildschirm aus.

Auch die Beschreibung eines Datenbanksystems erscheint uns an Hand vorstehender Unterteilung sinnvoll. Wir haben exemplarisch ein relativ einfaches und überschaubares Datenbank-System ausgewählt: es heißt RMS (= Record Management System) und stammt von Washington Computer Services. Nach der allgemeinen Beschreibung des Softwarepakets zeigen wir an einem unkomplizierten Musterbeispiel, wie man in der Praxis mit dem System umgeht.

Das Datenbank-System RMS besteht aus relativ wenigen Teilprogrammen:

- RMS
- RMSNEW
- INDEX
- REPORT
- RMSCOPY

Ordnet man diese fünf Paketteile den drei zuvor spezifizierten Funktionsgruppen zu, so ergibt sich Folgendes:

1. Einrichtung: Reservieren von Speicherplatz auf der Diskette durch RMSNEW und RMSCOPY.
2. Datenerfassung: Eingeben, Suchen und Ändern von Datensätzen mit RMS.
3. Datenverwaltung: Erzeugen und Anzeigen von Verzeichnissen der Datensätze durch INDEX und REPORT.

Das Konfigurieren einer Datei beginnt in der Regel mit der Analyse des Informationsbedarfs. Falls man beispielsweise eine Adressendatei erstellen will, ist vorab folgende generelle Frage zu klären: Welche Informationen soll das System ausgeben, wenn der Benutzer Daten abfragt? Die Antwort wird in der Regel lauten: Name, Vorname, Straße und Hausnummer, Postleitzahl und Wohnort, Telefonnummer mit Vorwahl, Datum der Eingabe und ähnliches mehr. Die Gesamtheit der Informationen, die wir (in diesem Fall) über eine Person speichern, heißt in der Fachsprache "Informationseinheit" (englisch: entity). Eine solche Informationseinheit kann aus einem oder mehreren "Datensätzen" (records) bestehen.

Ein Pluspunkt des Datenbank-Systems RMS ist die Option, daß die Informationseinheiten aus einem primären Datensatz und einem oder mehreren sekundären Datensätzen bestehen können. Die sekundären Datensätze sind über das Schlüsselfeld (darüber gleich mehr) fest mit dem zugehörigen primären Datenfeld verbunden; eine Trennung ist nicht möglich. Auf die sekundären Datensätze können weniger wichtige Informationen verlagert werden: In eine Adressendatei wird man vielleicht Angaben über die Hobbies der erfaßten Personen aufnehmen. Wenn sich das Hobby beispielsweise auf das Züchten und Sammeln von Orchideen erstreckt, ist die Liste des Orchideenbestands von den vorgelagerten Informationen wie Name, Adresse und Telefonnummer eindeutig getrennt.

Jeder Datensatz besteht aus einem oder mehreren Datenfeldern, von denen das Schlüsselfeld, wie der Name bereits sagt, den Schlüssel zum Datensatz darstellt. Es enthält den Suchbegriff, der zu der betreffenden Informationseinheit führt. Bei RMS hat jeder Datensatz leider nur ein einziges Schlüsselfeld, so daß im Beispiel der Adressendatei nur der Name (er ist sicherlich das sinnvollste Kriterium) als Schlüssel benutzt werden kann, nicht jedoch weitere zum Datensatz gehörende Datenwörter.

Der Inhalt des Schlüsselfelds ist beliebig wählbar, er darf jedoch innerhalb der Datei nicht doppelt vorkommen. Den Versuch, einen Datensatz mit genau gleichem Schlüsselwort zu speichern, den ein schon vorhandener Datensatz trägt, beantwortet der RMS-Editor mit der Fehlermeldung "RECORD ALREADY EXISTS, PUSH ANY KEY TO CONTINUE".

Nachdem wir Begriffe wie "Informationseinheit" und "Schlüsselfeld" geklärt haben, können wir uns den drei Funktionsebenen des Datenbanksystems RMS zuwenden.

1. Strukturieren der Informationseinheiten: Nehmen wir an, die Informationsbedarfsanalyse ist erledigt; wir haben bereits festgelegt, aus wieviel Feldern der primäre und gegebenenfalls auch die sekundären Datensätze bestehen sollen. Der nächste Schritt ist das Festlegen von Länge und Typ jedes einzelnen Felds (alphanumerische oder numerische Zeichen, aktuelles Datum usw.). Mit einem Textverarbeiter, beispielsweise SCREEN, wird eine Textfile mit dem Kennzeichen DIC eröffnet. Das Hauptprogramm von RMS, das ebenfalls RMS heißt, benutzt die DIC-File als Maske für die Eingabe der Datensätze. Die DIC-File für unsere Adressendatei kann zum Beispiel so aussehen wie in Bild 1 dargestellt. In jedem Feld des primären Datensatzes steht ein in Anführungszeichen gesetzter Text: das Feldprompt, das bei der Datenausgabe den variablen Daten automatisch vorangestellt wird.

Wenn die Strukturierung des Datensatzes vollendet ist, wird die DIC-File gespeichert, zum Beispiel unter dem File-Namen ADRES.DIC. Anschließend wird das Programm RMSNEW gestartet; dieses Teilprogramm von RMS richtet auf der Diskette Platz für die Datensätze ein. Zuvor fragt RMSNEW den Benutzer noch nach der Gesamtlänge der Datensätze und nach ihrer Anzahl. Aufgrund der Eingaben werden entsprechend viele Sektoren för RMS reserviert und vorläufig mit "Dummies" (Scheindaten) gefüllt. Das kann, abhängig vom Umfang der Datei, mehr oder weniger Zeit beanspruchen.

2. Nachdem die unumgänglichen Vorbereitungen erledigt sind, beginnt die eigentliche Handarbeit: die Dateneingabe. Dazu wird die Kommando-File RMS.CMD durch Eintippen von "RMS ADRES" aktiviert. RMS stellt fest, ob die File ADRES.DIC (die mit dem Textverarbeiter eröffnete Dictonary-File) und die File ADRES.RMS (die mit RMSNEW formatierte Datei) auf der Diskette stehen. Ist beides vorhanden, so erscheint auf dem

1

```
"*** List of addresses of persons ***"
NAME        30   A   "Name followed by first name: ";
STREET      25   A   "Street: ";
CITY        25   A   "City: ";
COUNTRY     20   A   "Country: ";
TEL         15   N   "Telephone number: ";
AGE          2   N*  "Age (presented numerical): ";
SEX          1   A*  "Sex (M, m or F, f): "[M,m,F,f];
$

"*** List of addresses of persons ***"
H1          70   A*   "Hobby #1: ";
H2          70   A*   "Hobby #2: ";
H3          70   A*   "Hobby #3: ";
$
```

86253-1

2

```
*** List of addresses of persons ***
Name followed by first name: ______________________________Street: _________
_______________City: ___________________________Country: ____________________
Telephone number: ______________Age (presented numerical): __Sex (M, m or F
,f): _

Het secundaire record als volgt:

*** List of addresses of persons ***
Hobby #1: ____________________________________________________________________
_____Hobby #2: ______________________________________________________________
__________Hobby #3: _________________________________________________________
_______________
```

86253-2

3

```
T "# Addresslist processed in alphabetical order sorted by name #"@1;
P "Date: "@1 $D@8 "Pagenumber: "@20 $P@33;
P "Name: "@1 NAME@8 "Country: "@40 COUNTRY@50 "Age: "@60 AGE@65 "Sex: "@70
  SEX@75;
P "Street: "@1 STREET@10 "City: "@40 CITY@47;
X ADRES.NDX
```

86253-3

Schirm die in Bild 2 wiedergegebene Maske.
Nach dem allerersten Programmstart ist noch kein Datenbestand vorhanden; es können nur neue Daten eingegeben werden. Die Striche, die auf die Feldprompts folgen (siehe Bild 2), werden dabei mit den eingegebenen Informationen überschrieben. Vor dem Speichern auf Diskette prüft RMS, ob die Daten die vorgegebenen Kriterien erfüllen. Beispielsweise dürfen in numerische Felder keine Buchstaben gesetzt werden, und im Feld "Geschlecht" darf natürlich nur "m" für "männlich" oder "w" für "weiblich" stehen, nicht aber irgendetwas anderes!

Nach Eingabe eines bestimmten Control-Zeichens erscheint die Maske des ersten sekundären Datensatzes auf dem Bildschirm; sekundäre Datensätze werden auf die gleiche Weise wie primäre Datensätze eingegeben. Ein weiteres Control-Zeichen leitet das Speichern des primären und der eventuell angehängten sekundären Datensätze ein.
Abgerufen wird eine gespeicherte Informationseinheit, indem man das zugehörige Schlüsselwort in die Maske schreibt und anschließend ein bestimmtes Control-Zeichen eingibt. Daraufhin setzt das System die zum Schlüsselfeld gehörenden Informationen in die übrigen Felder.

Bild 1. Beispiel für den Inhalt der .DIC-File, mit dem die Datensätze formatiert werden. Zur Demonstration haben wir eine Adressendatei gewählt.

Bild 2. Nach Starten des Hauptprogramms RMS erscheinen die Masken für den primären Datensatz und die angehängten sekundären Datensätze auf dem Schirm.

Bild 3. In die File ADRES.REP werden Anweisungen geschrieben, nach denen das Programm REPORT Listen aus dem vorhandenen Datenbestand erstellt.

4

```
* Bunny Bugs              * Duck Donald             * Duck Duffy
  Carrotstreet 2           Duckstreet 1              Pond 17
  Vegetable City           Duckcity                  Vegetable City
  USA                      USA                       USA
  01-230045                055-11223                 01-230167
  20                       33                        17
  M                        M                         M
  eating carrots           nagging                   acting in cartoons
  fooling stupid hunters   being lazy                boring Bugs Bunny

* Flintstone Fred         * Flintstone Wilma        * Frog Kermit (the)
  Rocklane 3               Rocklane 3                Muppetstreet 2
  Bedrock                  Bedrock                   Gonzo City
  USA                      USA                       USA
  023-11150                023-11150                 033-100023
  35                       32                        25
  M                        F                         M
  bowling                  having holidays           frogging
  eating                   domestic plants           presenting muppet shows

* Goofy                   * Panther Pink (the)      * Piggy (Mrs)
  Strawberry Lane 12       Insp.Clouseau Drive 11    Muppetstreet 2
  Duckcity                 Sellerscity               Gonzo City
  USA                      USA                       USA
  055-19902                016-673310                033-100133
  36                       45                        23
  M                        M                         F
  doing euheu euheu        imitating Peter Sellers   hitting frogs
                                                     being Miss Universe

* Rubble Barney           * Bear Yogi
  Rocklane 5               Round the Corner 27
  Bedrock                  Yellowstone Park
  USA                      USA
  023-11200                -
  34                       65
  M                        M
  bowling                  eating cookies
  drinking big pints       being lazy
                           snatching picknik baskets      86253-4
```

Bild 4. Gesamtbestand einer fiktiven Adressendatei. Eine Übereinstimmung dieser Daten mit den Daten lebender oder toter Personen ist rein zufällig!

5

```
# Addresslist processed in alphabetical order sorted by name #
Date:   1/17/86      Pagenumber:  00001

Name:   Bunny Bugs                  Country:  USA       Age: 20   Sex: M
Street:   Carrotstreet 2            City:  Vegetable City
Name:   Duck Donald                 Country:  USA       Age: 33   Sex: M
Street:   Duckstreet 1              City:  Duckcity
Name:   Duck Duffy                  Country:  USA       Age: 17   Sex: M
Street:   Pond 17                   City:  Vegetable City
Name:   Flintstone Fred             Country:  USA       Age: 35   Sex: M
Street:   Rocklane 3                City:  Bedrock
Name:   Flintstone Wilma            Country:  USA       Age: 32   Sex: F
Street:   Rocklane 3                City:  Bedrock
Name:   Frog Kermit (the)           Country:  USA       Age:25    Sex: M
Street:   Muppetstreet 2            City:  Gonzo City
                                                        86253-5
```

Bild 5. Diese alphabetisch geordnete Liste wurde mit einer .NDX-(Index-)File formatiert und mit REPORT erzeugt.

Anschließend können die ausgegebenen Daten mit Hilfe weiterer Control-Zeichen geändert oder auch gelöscht werden. Vor dem definitiven Löschen vergewissert sich das System noch einmal, ob dies wirklich die erklärte Absicht des Benutzers ist.

3. Ein Datenbank-System steht und fällt mit der Organisation und Präsentation des Datenbestands. Damit der Benutzer den Überblick über den Datenbestand behält, stehen ihm zum Erstellen von Listen, Verzeichnissen usw. die beiden Paketkomponenten INDEX und REPORT zur Verfügung. Ein naheliegendes Ordnungskriterium für ein Inhaltsverzeichnis ist die alphabetische Reihenfolge. Um ein alphabetisches Verzeichnis zu erstellen, eröffnet man mit dem Programm INDEX eine sogenannte Index-File (File-Kennzeichen .NXD).

Das wichtigste Programm zum Anfertigen von gedruckten Listen ist REPORT. Gesteuert wird REPORT von einer Textfile, die als Kennzeichen nicht .TXT, sondern .REP trägt. Die in dieser Textfile stehenden Zeilen (siehe Bild 3) interpretiert REPORT als Kommandos. Ausführliche Erläuterungen zur Syntax und zur Bedeutung der Kommandos enthält das zum RMS-Paket mitgelieferte Handtuch. Mit REPORT kann man aus den vorhandenen Datensätzen beliebige Datenfelder ausfiltern, Seitenüberschriften setzen und ähnliches mehr. Hier ein kurzer Überblick über die vielfältigen Möglichkeiten:

- automatisches Einsetzen eines Titels am Anfang jeder neuen Seite,
- Bearbeitung von Datensätzen in beliebiger Reihenfolge,
- Festlegen der Anzahl der zu druckenden Zeilen und der Zeilenanzahl pro Seite,
- Selektieren von Datensätzen nach vorzugebenden Kriterien,
- automatisches Einsetzen von Seitenzahlen,
- Zählen der in eine Liste übernommenen Datensätze,
- Addieren der Inhalte von numerischen Feldern (z.B. Geldbeträge),
- Ausdrucken des Tagesdatums an der gewünschten Stelle,
- Ausdrucken von Texten und Feldinhalten dort, wo sie erscheinen sollen.

Bild 4 gibt den gesamten Datenbestand unserer fiktiven Adressendatei wieder, wie er mit der File ADRES.DIC (Bild 1) eingegeben wurde. Eine daraus mit ADRES.REP (Bild 3) erzeugte Liste ist in Bild 5 zu finden. Die Reihenfolge der in der Liste enthaltenen Daten wird vom Inhalt der File ADRES.NDX bestimmt, in der stets die Inhalte der zu verarbeitenden Schlüsselfelder stehen. An der Entstehung der Liste in Bild 5 war die File ADRES.NDX in Bild 6 beteiligt. Eine .NDX-File kann automatisch mit dem Programm INDEX erzeugt werden, wobei das Datenfeld, nach dessen Inhalt alphabetisch sortiert wird, beliebig wählbar ist. Man kann auch eine willkürliche Reihenfolge festlegen, indem man die Schlüsselfeld-Inhalte in dieser Reihenfolge in eine .NDX-File aufnimmt.

Außer den beschriebenen Systemkomponenten gehört zu RMS noch ein Programm mit den Namen RMSCOPY, das sich eigentlich keiner der drei Programm-Kategorien zuordnen läßt. Hauptsächlich dient dieses Programm zur Erweiterung der bestehenden Datensätze durch zusätzliche Datenfelder. Auf einer zweiten Diskette eröffnet man eine neue .DIC-File mit der gewünschten Datenfeld-Anzahl und -Länge, und anschließend wird mit RMSNEW eine neue Datei eingerichtet. Den vorhandenen Datenbestand kopiert man einfach mit RMSCOPY von der alten auf die neue Diskette, wobei die Quell- und die Ziel-File den gleichen Namen haben müssen.

Jedes Datenfeld des auf der ersten Diskette stehenden Bestands wird in das entsprechende Datenfeld (mit mindestens gleicher Länge) auf der zweiten Diskette kopiert. Dabei gehen Informationen aus weggelassenen Datenfeldern verloren, neu hinzugenommene Datenfelder bleiben leer. Die neuen Felder können anschließend mit RMS beschrieben werden.

Zur Abrundung unserer Übersicht über RMS haben wir dieses Softwarepaket in Bild 7 noch einmal schematisch dargestellt und seine Vor- und Nachteile in Kurzform zusammengefaßt.

Selbstverständlich ist RMS nicht das einzige Produkt, das der Softwaremarkt für den beschriebenen Zweck anbietet. Für das Betriebssystem FLEX gibt es eine Reihe ähnlicher oder zumindest vergleichbarer Datenbank-Management-Programme, von denen wir einige nicht

Bild 6. Eine weitere, alphabetisch geordnete Liste.

Bild 7. Schematische Übersicht über das Datenbank-System RMS.

6

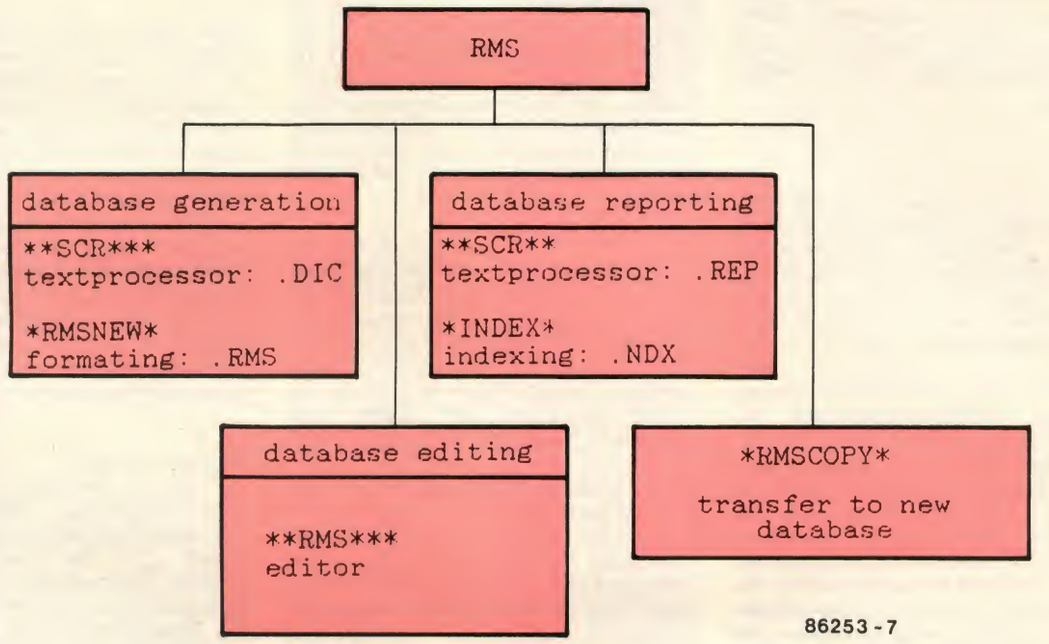
```
Bunny Bugs
Duck Donald
Duck Duffy
Flintstone Fred
Flintstone Wilma
Frog Kermit (the)
Goofy
Panther Pink (the)
Piggy (Mrs)
Rubble Barny
Yogi Bear
```

86253 - 6

Pluspunkte
- einfach und Übersichtlich
- Feldlängen sind sichtbar, kein Overflow
- Sortieren und Bearbeiten von Datensätzen nach beliebigen Kriterien
- benutzerfreundliche Präsentation der Datensätze auf den Eingabemasken
- fast beliebige Formatierung der Datenausgabe
- Kopieren des vollständigen Datenbestands möglich

Minuspunkte
- sequentiell organisierte Datei, daher relativ langsam
- im Editiermodus ist die Datensatz-Identifikation nur über ein einziges Schlüsselwort möglich
- primärer Datensatz und sekundäre Datensätze erscheinen nicht gleichzeitig auf dem Bildschirm

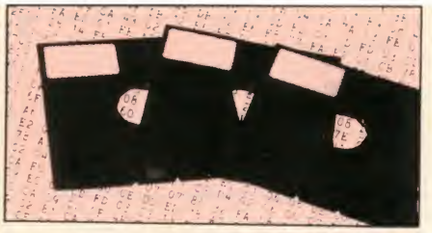
7



86253 - 7

unerwähnt lassen wollen. Eine der Alternativen zu RMS heißt XDMS: das ist ein Programm-Paket mit ähnlichen Eigenschaften wie RMS, seine Leistungsfähigkeit übertrifft jedoch noch die von RMS. Ebenso wie RMS gehört auch XDMS zur Gruppe der Programme, die nicht in einer Hochsprache, sondern maschinensprachen-orientiert geschrieben wurden. Auch XDMS besteht aus verschiedenen modularen Sub-Files. So ist zum Beispiel die File XDMS.CMD ein Kernprogramm (nucleus), das für sich allein nicht laufen kann, sondern "nur" die etwa 200 Grundfunktionen erledigt. Ein Vergleich zwischen XDMS und RMS läßt im wesentlichen folgende Unterschiede erkennen: Bei der Selektion von Datensätzen kann RMS (im Editiermodus) nur die Inhalte der Schlüsselfelder auswerten, während XDMS den Inhalt jedes beliebigen Felds als Suchkriterium nimmt. Eingegeben werden die Daten bei RMS mit Hilfe der Eingabemaske; XDMS zeigt dem Benutzer die Feldnamen einzeln an, die Daten werden bei XDMS nach jedem Feldnamen eingetippt.

Der Editor von XDMS (er heißt dort "UPDATE", weil er auch noch andere Funktionen hat) meldet sich mit dem Kommando-Prompt, die Kommandos bestehen aus jeweils einem alphanumerischen Zeichen. RMS wird dagegen mit Hilfe einer begrenzten Anzahl von Steuerzeichen (control codes) bedient. XDMS belegt nur so viel Speicherplatz, wie wie es zum Speichern der Daten notwendig ist, während RMS unabhängig vom tatsächlichen Bedarf feste Bereiche für die Feldinhalte reserviert. XDMS kann daher in der Regel auf der gleichen Diskette mehr Datensätze unterbringen als RMS. Zusammenfassend läßt sich feststellen, daß sicherlich auch XDMS zu den interessanten Flex-Anwenderprogrammen gehört, es ist jedoch noch ein Stück komplizierter als RMS. Wie schon erwähnt, bilden die in BASIC (oder in einer anderen Hochsprache) geschriebenen Programme die zweite Gruppe des Software-Angebots. Diese Programme sind normalerweise bei weitem durchsichtiger als RMS oder XDMS. Anwender, die auf einem käuflichen Produkt aufbauend ihre eigene, an ihre speziellen Bedürfnisse angepaßte Version schreiben wollen, werden zu einem Datenbank-Programm aus dieser Gruppe greifen. Ein Handicap ist allerdings die deutlich niedrigere Arbeitsgeschwindigkeit, denn ein in BASIC geschriebenes und von einem Interpreter (z.B. TSC's XBASIC) verarbeitetes Programm muß zwangsläufig langsamer laufen als RMS oder XDMS. Andererseits sind die BASIC-Datenbank-Programme meistens menügesteuert und deshalb ein Stück bedienfreundlicher. Leider wird dies oft wieder dadurch aufgewogen, daß viele Sub-Moduln (für jede Menü-Funktion eins) eigenständige BASIC-Programme sind, die erst bei Aufruf der Funktion von der Diskette geladen werden müssen. Zum Schluß hier noch die Namen von drei typischen Vertretern dieser Programm-Kategorie: sie heißen DATATREE, INFOMAG und DATAMASTER.



EC-65K / 65816: Software

'Die Welt wird schöner mit jedem Tag,. . .'

Peter von der Linden

'. . . wer weiß, wie das noch enden mag': So oder ähnlich heißt es in einem bekannten Frühlingsgedicht. Angesichts von Tschernobyl gewinnt diese Aussage wohl eine Seite, die der Dichter nicht einkalkuliert hat. Was jedoch den EC-65K angeht, wissen wir recht gut, worauf es hinausläuft: Auf viel leistungsfähige, interessante und dazu noch preiswerte Software für diesen Rechner - und dies insbesondere für seinen 65816-Teil.

Wer das Editorial gelesen hat, weiß: Eigentlich müßten wir den EC-65K in EMMS (Elektors Modulares Multiprozessor-System) umbenennen. Doch das kommt nicht in Frage: Wir ärgern uns immer noch, daß wir unseren 'SAMSON 65' in EC-65 umbenennen mußten . . .

Jetzt kommt uns dies allerdings zugute: Denn das 65816-Betriebssystem des EC-65K heißt schon von Hause aus EC-DOS - es stammt von der Münchner Firma ees, der Name steht als Abkürzung für ees-Computer-DOS. Man könnte sagen: 'Nomen est omen' - nämlich ein gutes Omen für die Zusammenarbeit zwischen der Firma ees und der ec-Redaktion. Bevor wir auf die für unsere Leser interessantesten Fragen eingehen (was das Betriebssystem kann, welche Software hierfür verfügbar ist und was das alles kostet), zunächst einige Worte zur Firma ees und einigen Zusammenhängen.

Die Firma ees arbeitet im Bereich der industriellen Computertechnik und innerhalb dieser insbesondere auf dem Gebiet der hochauflösenden Bilderfassung in Echtzeit (speziell für die Qualitätssicherung). Der Geschäftsführer Jürgen Helbing kennt seit etwa drei Jahren den Entwickler des 65816, William D. ("Bill") Mensch und arbeitet mit seinem Team bereits seit etwa zwei Jahren an der Software-Entwicklung für den 65816. Bereits seit Anfang dieses Jahres wird der modular aufgebaute EC65000 und der darauf basierende Bildverarbeitungs-Rechner OPTOCOS mit der zugehörigen 65816-Software von der Firma ees vertrieben. Herr Helbing hat einen Standard für Betriebssystemaufrufe beim 65816/65802 entwickelt, den 'COP - 65 K'.

Wir stellen diesen Standard im Rahmen dieses Artikels vor, und dies aus gutem Grunde: Während diese Zeilen entstehen, wird aufgrund einer Vereinbarung zwischen der Firma ees und der ec-Redaktion der COP - 65 K auf dem EC-65K installiert. Sobald dies geschehen ist, müßte theoretisch die gesamte ees-Software auf dem EC-65K lauffähig sein. Jedoch sind beide Seiten aufgrund jahrelanger Erfahrung klug geworden: Es wird eine gründliche Erprobungsphase folgen, hierzu stellt die Firma ees einigen ec-Autoren die Software kostenlos zur Verfügung (diese Autoren werden dann auch entsprechende Einführungsartikel schreiben).

Die komplette, recht umfangsreiche Version des COP 65K können wir aus Platzgründen hier nicht abdrucken. Wir beschränken uns daher auf eine gekürzte Version, aus der deutlich wird, worum es geht. Darüber hinaus wird die vollständige Version auf einer der Disketten enthalten sein, die wir in Kürze den Software-Verteilern zuleiten (im Wordprocessor-Format). Interessierte Firmen und Fachverlage können von der ec-Redaktion oder der Firma ees einen Ausdruck erhalten.

Bevor wir nun die Software selber beschreiben, noch eine wichtige Bitte: Rufen Sie nun nicht einfach bei der Firma ees an, um sich zu erkundigen, wann es so weit ist - sie könnten nicht nur die normalen Arbeiten dort stören, sondern auch die Anpassung und Erprobung, die die Firma ees nach Kräften fördern will, und die Entwicklung weiterer 65816-Software, die dort im vollen Gange ist. Die ec-Redaktion wird die Software-Verteiler per Rundbrief über die Fortschritte informieren, wir hoffen zuversichtlich, daß wir in ec 6 bereits mit ganz konkreten Berichten beginnen können - und Sie bereits mit dieser Software arbeiten können.

Worum es geht. . .

. . . ist ein Betriebssystem nach MS-DOS/UNIX-Vorbild. Dieses Betriebssystem kann Festplatten bis 4 Gigabyte verwalten (man hat vorausschauend an optische Speichermedien gedacht) und Floppy-Laufwerke beliebiger Kapazitäten: Die Parameter-Tabellen sind in FORTH geschrieben und deshalb leicht und praktisch beliebig an vorhandene Hardware anpaßbar. Dies gilt übrigens auch

COP 65K

Ein Standard für Betriebssystemaufrufe beim 65816/65802

COP 65K ist der Versuch einer Normierung der Betriebssytemaufrufe für die CPU-Serie 65xxx nach MS-DOS/Unix-Vorbild. Es soll versucht werden, einen möglichst weit verbreiteten Konsens über die Aufrufe des Betriebssystems zu erreichen, um einem ungeordneten Chaos (siehe 6502 1976 - 1984) vorzubeugen, das eine Weiterverbreitung der 65xxx-Serie verhindern könnte.

Die Erreichung dieses Ziels wird durch das frühzeitige Erscheinen des COP 65K möglich. Unter dem COP 65K, der schnell und einfach auch in vorhandenen Betriebssystemen implementiert werden kann, laufen bereits heute (Stand Januar 1986):

Ein DOS, ein 65xxx-Assembler, ein Kommando-Interpreter, die Sprache FORTH und ein 'in-the-flight'-BASIC-Interpreter.

Die Bezeichnung COP 65K ist abgeleitet aus 65000 und dem Befehl 'COP' der 65xxx-Serie. Der COP-Befehl wurde vom Entwickler der 65xxx-Prozessoren, Mr. William D. Mensch, für Coprozessor-Funktionen vorgesehen und wird im COP 65K als Betriebssystemaufruf benutzt.

Der COP-Befehl

Der COP-Befehl wird aus folgenden Gründen als Betriebssystemaufruf verwendet:

- COP ist sehr schnell (8 CPU-Zyklen)
- COP ist kurz (2 Bytes Programmspeicher)
- COP ist lageunabhängig. Jede Speicheraufteilung kann verwendet werden (ROM/RAM/. . .). Eine Kenntnis von Einsprungadressen ist nicht notwendig
- COP-RTI ist eine Softwareunterbrechung. Der Prozessor-Status (Registerlängen, IRQ-Flag) wird nach RTI wieder hergestellt, ohne daß zusätzlich Zeit oder Befehle notwendig wären.

Das unmittelbar hinter COP angegebene Byte wird bei den Betriebssystemaufrufen nicht verwendet, da es nur sehr zeitaufwendig geladen werden kann.

Die COP-Philosophie

Jede Ein/Ausgabe-Operation erfolgt über

für eine Memory-Disk (RAM-Disk oder EPROM-Disk). Das Betriebssystem ist Netzwerk-fähig (EC-Net, über V24 / 19200bd) und so fort.

... ist ein leistungsfähiger 65xxx-Assembler, der natürlich den kompletten Befehlssatz des 65816 beherrscht. Der Assembler bietet allen Komfort, den ein moderner Assembler nun einmal bieten sollte. Selbstverständlich kann man auch mit Relativ-Files arbeiten, Programmbibliotheken erstellen und ähnliches.

... ist ein leistungsfähiger Editor, der sich auch als Textverarbeiter nutzen läßt.

... ist ein BASIC-Interpreter, der bis zu 5 MByte BASIC-Source verwalten kann und durch das 'in-the-flight'-Prinzip fast so schnell ist wie ein Compiler. Von der Syntax und den Möglichkeiten her ist dieser Interpreter mit den modernsten Versionen des Mikrosoft-BASICs vergleichbar. Man kann label-orientiert arbeiten, und zum Programmpaket gehört natürlich auch ein komfortabler BASIC-Editor. Für die BASIC-Programmierer unter unseren Lesern schon einmal ein paar der Befehle (kennen Sie die schon alle?): TRON und TROFF, COLLECT, CONT, SYS mit REGSET, REGREAD und REGINIT, BRKON und BRKOFF, HCOPY, FLOAT, LABEL, ELSE, SWAP, WHILE, WENT, REPEAT, UNTIL, ST, ... Genug für den Moment.

... ist ein FORTH 65K. Es basiert auf dem fig-FORTH-Standard, enthält aber bereits wesentliche Erweiterungen. Auch der Kommando-Interpreter ist in FORTH geschrieben und kann daher vom Anwender erweitert und an spezielle Anforderungen angepaßt werden.
Und nun zu den schönsten 'Sächelchen': In Arbeit ist ein BASIC-Compiler und PASCAL 65K. Turbo-Pascal ist bekanntlich schnell. PASCAL 65K soll in der Syntax identisch sein - und noch schneller! Und dies traut die ec-Redaktion den Software-Profis bei ees ohne weiteres zu! Man kann also die vielfältige Literatur zu Turbo-Pascal verwenden - inklusive ausgedruckter Programme. Und für die besonders fleißigen Hardware-Nachbauer unter unseren Lesern sind auch noch ganz andere Möglichkeiten denkbar (Stichworte: Z80-Karte, V30-Karte und UDC-Karte im EC-65K...).

'Und was kostet das?'

Sitzen Sie gut? In dem Artikel 'EC-65K: Software' in ec 4 haben wir die Vermutung geäußert: 'Nun ist solche Software für einen neuen Prozessor nicht gerade billig...'. Stimmt auch, und trotzdem haben wir daneben gelegen. Denn es wird zwei getrennte Lizenzformen geben, die wir - damit 'das Kind einen Namen hat' - mal als 'Industrie-Lizenz' und 'Hobby-Lizenz' bezeichnen wollen. Die Industrie-Lizenz müssen Sie erwerben, wenn Sie diese Software kommerziell einsetzen wollen (egal, ob auf einem Rechner der Fa. ees oder auf dem EC-65K). Die Hobby-Lizenz können Sie erwerben, wenn Sie diese Software nur privat und ohne hieraus kommerziellen Nutzen zu ziehen einsetzen wollen. Im Rahmen der Hobby-Lizenz wird die Möglichkeit geschaffen, die Software legal zu günstigen Konditionen weiterzugeben (Unter-Lizenzen), daher wird hier auch die gesamte Dokumentation auf Diskette geliefert. Außerdem erhalten Schüler und Studenten, die einen gültigen Ausbildungsnachweis vorlegen, generell einen Preisnachlaß von 50%, zahlen also nur die Hälfte.

Und nun zu den Preisen: Die Industrie-Linzenz für DOS, Kommando-Interpreter und COP 65K (diese drei Teile bilden ein zusammengehörendes Software-Paket) kostet 500,— DM, das BASIC kostet 600,— DM und der Assembler mit Editor 300,— DM. Die Industrie-Kopier-Lizenz kostet 50% der genannten Preise. Die Hobby-Lizenz kostet für den Erstkaüfer 30% und die Hobby-Kopierlinzenz die Hälfte (also 15% des ursprünglichen Preises). Ein Schüler oder Student landet als Zweitkäufer mithin bei 7,5%. Jetzt dürfen Sie rechnen... Und wenn Sie als Schüler oder Student 105,— DM für die komplette Software errechnen, haben Sie richtig gerechnet.

Wer nun meint, die Idee mit der Hobby-Lizenz stamme von der ec-Redaktion, hat falsch getippt: Sie stammt von Herrn Helbing. Das wir sofort begeistert waren, versteht sich von selbst. Dahinter steckt folgende Überlegung: Man muß einen Unterschied machen zwischen Raubkopierern, die gute und daher zu Recht teure Software schlicht klauen, um sich zu bereichern, und Computeristen, die gerne rein privat oder im Rahmen ihrer Berufsausbildung mit bestimmter Software arbeiten wollen, sie aber nicht bezahlen können und sich deshalb mit schlechtem Gewissen Raubkopien besorgen. Und dabei werden Letztgenannte von den Erstgenannten häufig genug noch übers Ohr gehauen, um es einmal auf gut Deutsch zu sagen.

Nun gut: Für einen Schüler oder Studenten als Zweitkäufer liegt der Preis also bei 105,- DM - legal, ohne schlechtes Gewissen! Und da haben wir als Redaktion auch noch eine Idee: Es gibt leider trotz anziehender Konjunktur noch immer rund 2 Millionen Arbeitslose. Da ist vielleicht mancher darunter, der sich jetzt im Bereich der Computertechnik selbstständig fortbilden will, um sich eine bessere Startposition für die Zukunft zu verschaffen. Deshalb offene Frage an Sie, Herr Helbing: Läßt sich hier nicht vielleicht auch etwas machen, zum Beispiel gegen Vorlage einer Bescheinigung des Arbeitsamtes?

(Wir konnten die telefonische Antwort noch hier einfügen: 'Ja, warum denn nicht.')

Eins ist wohl klar: Wer da noch meint, er müsse sich als Raubkopierer betätigen, handelt unverantwortlich und schäbig! Und wer meint, damit auch noch Geschäftchen machen zu können, gehört vor den Kadi!

Eine Frage ist vielleicht noch zu beantworten: Wie kommt der Chef einer Firma, die im industriellen Sektor arbeitet, auf eine solche Idee (als Anhaltpunkt dazu noch: Der EC65000 von ees kostet in der

---

einen COP. Die Gerätenummern sind festgelegt. Ein spezieller COP-Befehl wählt ein Gerät an. Alle weiteren COP's gehen an das zuletzt angewählte Gerät, soweit nicht ein neues angewählt wird.

Die gewünschte Funktion wird im X-Register übergeben. Nach der Rückkehr wird ein Gerätestatus im X-Register zurückübergeben. Ein gesetztes Carry-Flag zeigt einen gültigen Status an.

Funktionsparameter werden im Akku übergeben und zurückübergeben. Das Y-Register bleibt meistens unverändert (!).

COP 65K ist für den Anwendungsprogrammierer transparent, für den Systemprogrammierer in weiten Bereichen implementierbar und erweiterbar und verarbeitet Einzelzeichen, Zeichenketten und Speicherblöcke.

Einige typische COP-Aufrufe

Sende das Zeichen '0' an das zuletzt aufgerufene Gerät:

```
LDA # $30      ;ASCII '0'
LDX # 10       ;Funktion 'Sende Zeichen'
COP # 0
```

Empfange ein Zeichen vom zuletzt angewählten Gerät:

```
LDX # 8        ;Funktion 'Empfange
               Zeichen'
COP # 0
STA Zeichen
```

Wähle Gerät # 1 (Bildschirm) an:

```
LDA # 1        ;Gerätenummer 1
LDX # $84      ;Funktion 'Wähle Gerät'
COP # 0
```

Status im COP 65K:

```
LDX # 8        ;Funktion 'Empfange
               Zeichen'
COP # 0
BCS Fehler     ;gesetztes Carry-Flag
               zeigt Status an
STA Zeichen
```

Fehler: STX Status

Zeichenorientierte COP-Aufrufe

Die meisten Aufrufe des Betriebssystems über die COP-Funktion behandeln Einzelzeichen mit 8 Bit, also einem Byte.

In diesem Modus wird ein COP-Aufruf mit kurzem Akku (für das zu sendende oder zu empfangene Zeichen) und mit kurzen Indexregistern (für die Funktionsnummer im X-Register) ge-

PC-Ausrüstung knapp 10.000,— DM, allein die fertig bestückten und geprüften Platinen liegen bereits bei etwa 5000,— DM — das sind natürlich Industrie-Produkte entsprechender Qualität!). Die Antwort auf diese Frage ist eigentlich einfach: Auf eine solche Idee kommt man, wenn man im zarten Alter von 10 Jahren die ersten Transistoren 'zusammengebrutzelt' (wörtliches Zitat) hat und seit dem 14 Lebensjahr Elektor liest (aus dem Lesen ist aufgrund mangelnder Zeit in den letzten Jahren zwar mehr ein durchblättern geworden, aber das Herz für den Hobbyisten ist halt geblieben).

Und ansonsten?

Unser Zeitplan für den EC-65K ist leider etwas durcheinander geraten. Worum wir uns unter anderem unplanmäßig schnell gekümmert haben, haben Sie gerade gelesen. Aber es gab auch noch andere Faktoren: Zum Beispiel die Verzögerung bei der Platinenherstellung, zum Beispiel die Tatsache, das der 65816 in der 4-MHz-Version gegen alle Erwartung noch nicht in größeren Stückzahlen lieferbar ist (was schon anders sein kann, wenn Sie diese Zeilen lesen).

Wir haben uns dann gesagt: Na ja, da brauchen wir uns mit der Anpassung der Ohio-Software nicht allzusehr zu beeilen. Das war ein Trugschluß. Denn noch etwas war nicht vorhersebar: Die Anpassung der Ohio-DOS-Software an den EC-65K erwies sich als unerwartet problematisch. Nicht wegen unserer eigenen, gut dokumentierten Erweiterungen: In der Ohio-Version 3.3 sind einige 'Patches' enthalten, die in dieser Form eigentlich schlicht verboten gehören. Obwohl wir in den 'Memory-Checker' des DOS $CF als oberste Page eingeschrieben hatten, schrieben da Patches munter in den D-Bereich, der ja beim EC-65K System-RAM ist — und überschrieben so die Bildschirm-Parameter: Der Salat auf dem Schirm war perfekt. Dies ist inzwischen gelöst, wovon sich alle Leser überzeugen konnten, die sich auf der Hobbytronic in Dortmund den EC-65K angesehen haben. Aber es ist einige Zeit verloren.

Deshalb jetzt folgendes Verfahren: Sobald die Schlußredaktion dieses Heftes beendet ist, erhalten die Software-Verteiler lauffähige, aber noch fehlerbehaftete Disketten und schriftliche Informationen darüber, was wir bisher angepaßt haben. Dann können sich alle Leser, die hieran Interesse haben, am 'Entlausen' der Ohio-Software beteiligen. Was Sie herausfinden, schicken Sie uns in knapper schriftlicher Form. Wir leiten es an einen der beiden Software-Verteiler in Aachen weiter (Herr Pfeiffer), der sich bereit erklärt hat, hier als eine Art Koordinierungsstelle mitzuarbeiten, um die Redaktion zeitlich etwas zu entlasten. Wir denken, auf diesem Wege wird die restliche Anpassung nicht nur sehr schnell, sondern auch sehr gründlich vonstatten gehen, und dies wird der Ohio-Software gut tun: Sie wird auf diesem Wege gleich auch noch gründlich entlaust.

UCSD-p-Pascal

Eine Beschreibung des UCSD-p-Systems ist an anderer Stelle in diesem Heft enthalten, hier nur kurz einige Worte zum Stand der Dinge: Wegen der bisher geschilderten Probleme hat sich die End-Abnahme der Version für den EC-65 durch die Redaktion verzögert (schimpfen Sie also nicht über die Fa. Rieger, falls Sie es dort bestellt haben). Die End-Abnahme muß natürlich stattfinden, bevor wir diese Version als EC-65-Version des UCSD-p-Pascals autorisieren können. Gleichzeitig hat einer unserer externen Mitarbeiter, Herr Frey, einen neuen BIOS für das UCSD-Pascal entwickelt. Und der war auf einmal fertig, bevor die End-Abnahme stattfinden konnte... (saubere Arbeit, Herr Frey!).

Ursprünglich sollte der NEWBIOS als Patchdiskette für etwa 30,- DM von der Fa. Rieger nachgeliefert werden können und Herr Frey für seine Bemühungen einen Anteil pro verkaufte Diskette erhalten - also ähnlich, wie es bei anderen besonders guten Software-Paketen auch gehandhabt wird (BASIC+, ASS114). Das hätte gleichzeitig den lizenzrechtlichen Bedingungen entsprochen, die den Endverkaufspreis der Fa. Rieger schlichtweg festschreiben. Nachteil: Dann hätte es zwei Versionen gegeben. Und nun - was tun?

Ganz einfach: Die Firma Rieger wird die Version mit dem NEWBIOS liefern - zu ihrem festgelegten Preis. Aber man muß eine Beschreibung des NEWBIOS für 10,- DM mitkaufen - und die gibt die Fa. Rieger ohne Abzug an Herrn Frey weiter. Die Beschreibung braucht man schließlich sowieso, Herr Frey wird sie auf der Grundlage eines Artikelentwurfes für dieses Heft, für den leider kein Platz mehr war, erstellen. Die mitgelieferte Dokumentation wächst damit auf fast 370 Seiten! Die End-Abnahme dieser Version haben wir seitens der Redaktion fest für den 10.6. eingeplant, unmittelbar darauf kann die Auslieferung beginnen (das ist noch vor dem Erscheinungstermin dieses Heftes).

Noch schnell ein paar Worte zum NEWBIOS, zum Teil fast wörtlich aus dem Artikelentwurf von Herrn Frey zitiert: Ein Betriebssystem mit den Möglichkeiten des UCSD-p-Systems ist notgedrungen sehr umfangreich. Die Dienstprogramme und die Systemprogramme umfassen alleine schon etwa 140 KByte, die ständig auf der Diskette verfügbar sein müssen. Die Disketten-Kapazität ist beim OSI-Format selbst bei einem 80-Track-Laufwerk auf 160 KByte beschränkt, auch das Directory braucht Platz... Kurzum: Der NEWBIOS ist in der Lage, ein doppelseitiges 80-Track-Laufwerk als ein Laufwerk zu behandeln. Der ursprüngliche BIOS dagegen war nicht einmal in der Lage, ein doppelseitiges 80-Track-Laufwerk zu verwalten, denn Pascal erwartet, daß es die Schreib/Leseköpfe unabhängig voneinander bewegen kann. Ein Versuch, dies zu ignorieren, führt schon beim Booten zu Fehlern.

Mit dem NEWBIOS kann man daher den Komfort, den das UCSD-p-System bietet, erst wirklich nutzen. Außerdem hat Herr Frey auch noch wichtige Utilities auf den NEBIOS und den EC-65 / EC-65K abgestimmt und bereitet eine Version für den

---

startet und kehrt auch so mit einem 8-Bit-Status (im X-Register) zurück.

Bei zeichenorientierten COP-Aufrufen werden also alle Parameter und Resultate in den CPU-Registern übergeben.

Blockorientierte COP-Aufrufe

Blockverarbeitung ist zur Steigerung des Durchsatzes des Betriebssystems notwendig. Wichtige Beispiele sind:

- Senden eines Strings auf den Bildschirm
- Senden einer festen Anzahl von Zeichen auf eine Plattendatei
- Empfangen von ganzen Programmblöcken von Plattendateien

Bei vielen Befehlen reichen die CPU-Register nicht aus, um alle Parameter eines Aufrufs zu übergeben (offene Datei mit Dateiname, Gerätenummern, Zugriffsart etc.).

In all diesen Fällen erfolgt die Parameter-Übergabe über eine Direkt-Page in Bank 0 des Computers.

Struktur eines COP-Aufruf-Parameterblocks

Ein Parameterblock besteht aus bis zu 8 Bytes. Er kann prinzipiell überall in der Bank 0 abgelegt werden. Organisatorisch und aus Zeitgründen ist es sinnvoll, immer die Bytes 0 bis 7 der Direkt-Page des Anwenderprogramms für COP-Aufrufe zu reservieren!

Bei blockorientierten COP-Aufrufen wird im langen (!) Akku die Adresse des Parameterblocks an den COP-Aufruf mitgeteilt. Die 16 bit des langen Akkus adressieren die gesamte Bank 0! Beispiele:

Parameterblock bei $3F80

```
LDA # $3F80   ;Parameterblock
LDX # 14      ;Funktion 'Sende Block
              bis Endezeichen'
COP # 0
```

Parameterblock am Anfang der verwendeten Direkt-Page

```
TDC           ;Direkt-Page-Register
              nach Akku
LDX # 14      ;Funktion 'Sende Block
              bis Endezeichen'
COP # 0
```

Inhalt des Parameterblocks

Da die 65816-CPU bis zu 16 MByte adres-

EC-65K vor, die voraussichtlich sehr rasch verfügbar sein wird.

Schlußbemerkung

Sie sehen: Hier hat sich sehr rasch sehr viel getan. Und wir sind sicher: Angesichts der Tatsache, daß der Aufbau des EC-65K bei den meisten Lesern doch erst jetzt beginnen kann - woran die Redaktion schlicht unschuldig ist - werden Sie uns zustimmen: Wir haben in ec 4 nicht zuviel versprochen, obwohl wir uns teilweise in Bezug auf den zeitlichen Ablauf verschätzt haben.

---

sieren kann, sind alle COP-Zeiger 3-Byte-Zeiger und alle Blocklängen als 3-Byte-Längen ausgelegt. Aufbau des Parameterblocks:

```
Byte 0
Byte 1          Zeiger auf Daten
Byte 2

Byte 3
Byte 4          Länge der Daten
Byte 5

Byte 6          Zusatzinfo

Byte 7          reserviert
```

Dies sei verdeutlicht am Beispiel des Parameterblocks für folgenden Funktionsablauf: 'Empfange Zeichen bis zum Ende-Zeichen 'cr' (ASCII 13) und lege sie ab dem Datenzeiger ab. Maximale Länge = Länge im Parameterblock; Endezeichen = Zusatzinfo im Parameterblock'.

```
ACL             ;Akku auf 'lang'

XYS             ;X- und Y-Register auf
                 'kurz'

LDA
#<Buffer        ;Anfang
LDY #↑Buffer
STA Param
+ 0
STY Param
+ 2

LDA
#<Buffer        ;Länge, niederwertige
                 Bytes
LDY #↑Buffer    ;Länge, höherwertiges
                 Byte
STA Param + 3
STX Param + 5

LDY # 13        ;Endezeichen
STY Param + 6

LDA
#<Param         ;(TDC, wenn Direkt-Page
                 0...7)
```

Die I/O-Selekt-Funktionen für Geräte

Die I/O-Selekt-Funktionen nach Nummern

Die I/O-Selekt-Steuerfunktionen

Sonder-Funktionen (Nummern noch nicht festgelegt)

(Die vier letztgenannten Abschnitte entfallen aus Platzgründen, die Red.)

Die Geräte-Nummern

Alle Geräte werden über Geräte-Nummern von 0 bis 63 angewählt. Die Standard-Geräte (0 bis 7) sind in der folgenden Tabelle aufgeführt. Wird ein Terminal als Bedienkonsole verwendet, so ist seine Geräte-Nummer auch zum Senden auf den Bildschirm die '0'.

Die derzeit üblichen Erweiterungsgeräte belegen die Nummern 8 bis 17. Weitere übliche Geräte werden noch mit den Nummern 18 bis 31 definiert. Die Nummern 32 bis 63 stehen für die Erweiterungen des Anwenders zur Verfügung und sind nicht normiert.

Die Geräte 192 bis 255 stehen für Dateien auf Massenspeichern zur Verfügung, von denen mehr als eine gleichzeitig offen sein kann.

(Tabelle entfällt aus Platzgründen. Die folgenden Abschnitte: 'Die interne COP-Struktur', 'Folgendes ist zu beachten', 'Für den Programmierer von COP-65-Gerätetreibern', 'Ein Minimaltreiber für die Blockfunktionen', 'COP 65K - Start (Reset)', 'COP-Vektoren' und 'Vorschlag für die Speicheraufteilung' wenden sich an die Systemprogrammierer und entfallen daher hier ebenfalls. Wichtig scheint uns noch folgendes Zitat aus diesem Teil: 'Das Aufrufprinzip von COP 65K ist einfach und übersichtlich... Von vielen Aufrufalternativen wurde diese aufgrund der Lageunabhängigkeit, der Erweiterbarkeit, der Transparenz und der Geschwindigkeit ausgewählt.' Die Red.)

Die Autoren bitten diejenigen, die Erweiterungen und Änderungen am COP 65K vornehmen, um die Bekanntgabe von Speicheraufteilung, Vektoren, Gerätenummern (Standard und Erweiterungen) und Funktionsnummern (Standard und Erweiterungen).

Um sich COP 65K - System nennen zu dürfen, muß ein System minimal Tastatur und Bildschirm oder ein Terminal mit den Funktionen $00 bis $16 und $80 bis $86 steuern können.

Die Benutzung / Verwendung des COP 65K - Standards ist meldepflichtig und gebührenfrei!

ees Fertigungs- und Vertriebs GmbH
Pelkovenstr. 51
8000 München 50

Jürgen Helbing 1986

Als deutscher Lizenzinhaber für das Ohio-DOS und als lizensierter Distributor für das UCSD-p-Pascal vertreiben wir in Absprache mit der ec-Redaktion die von der Redaktion geprüfte Software für den

EC-65 — EC-65K

Auf Anfrage senden wir Ihnen gerne die aktuelle Übersicht.

OSI-Tutorial-Disk 5 (= Ohio-DOS-Grundlizenz) ..... **160,— DM**
Inklusive der in diesem Heft besprochenen Z80-Disketten:
Alle 40-Spur-Free-Soft-Disketten je **20,— DM**
Alle 80-Spur-Free-Soft-Disketten je **30,— DM**
Disk 18: ASS 114 ..... **80,— DM**
Neue Disk 17: BASIC+ mit
- Laufwerkserkennung 40/80 und automatischer Umschaltung
- Shorthands
- Ausführungs-Meldung für Befehle KI und NN
- DOS 4.01

mit Dokumentation ..... **80,— DM**
ROM I ESS 533A ..... **35,— DM**
ROM II ESS 523D ..... **35,— DM**
ROM III ESS 546 ..... **35,— DM**

Heureka, es ist soweit:

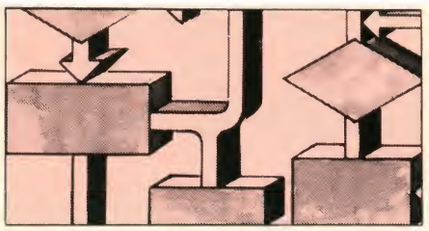
UCSD-p-Pascal

und gleich mit dem **NEWBIOS**
2 Disketten 80 Spur doppelseitig, inkl. 360 Seiten Dokumentation
350,— DM

plus 10,— DM für Dokumentation NEWBIOS

Alle Preise exkl. MWSt., Porto und Verp.

I. Rieger
Büromaschinen
Mühlenredder 11
2071 Kuddewörde-Rotenbek
Tel. 04154/3614



EC-65; Listing-Stopper

Peter Wehner

Sicher haben Sie sich beim Durchsuchen eines Listings schon gewünscht, den Durchlauf auf dem Bildschirm anders als mit CTL-C (Control-C, BREAK) stoppen und nach Belieben weiterlaufen lassen zu können. Integrieren Sie dieses Source-Listing in Ihrem Betriebssystem (Track 00, das zugehörige Source-Listing ist auf Diskette 6 enthalten), und schon können Sie Ihr Listing mit CTL-S (für Stop) stoppen und mit CTL-Q (für Quittierung) weiterlaufen lassen oder "wie gewohnt" mit CTL-C abbrechen.
(Anmerkung der Redaktion: Eine praktische Sache. Wir hoffen, es zeitlich noch zu schaffen, diese Ergänzung auf der Patch-Diskette für den EC-65K gleich mit unterzubringen. PL)

```
SAMSON'S ASSEMBLER       PAGE 01



0005: 24BE                    ORG   $24BE
0010: 24BE           VAIFR    X     $E10D
0015: 24BE           VAPAD    X     $E101
0020: 24BE           VIDEO    X     $F000
0025: 24BE           KPDO     X     $2325
0030: 24BE           RECCHA   X     $F71D
0035:
0040: 24BE 20 00 F0           JSR   VIDEO
0045: 24C1 AD 0D E1           LDA   VAIFR   key pressed?
0050: 24C4 29 02              ANDIM $02
0055: 24C6 D0 01              BNE   BZ
0060:
0065: 24C8 60         BY      RTS           if not return
0070:
0075: 24C9 AD 01 E1   BZ      LDA   VAPAD   get the character
0080: 24CC 29 7F              ANDIM $07       and mask it
0085: 24CE C9 13      CTLS    CMPIM $13     was it a CTL-S ?
0090: 24D0 F0 08              BEQ   CTLC
0095: 24D2 C9 03              CMPIM $03     was it a CTL-C?
0100: 24D4 D0 F2              BNE   BY
0105: 24D6 8D 25 23   ABORT   STA   KPDO
0110: 24D9 60                 RTS
0115:
0120:
0125: 24DA 20 1D F7   CTLC    JSR   RECCHA
0130: 24DD C9 03              CMPIM $03     was it a CTL-C ?
0135: 24DF F0 F5              BEQ   ABORT
0140: 24E1 C9 11              CMPIM $11     is it a CTL-Q?
0145: 24E3 D0 F5              BNE   CTLC
0150:
0155: 24E5 60                 RTS


   SYMBOL TABLE 3400 343C
   ABORT  24D6   BY     24C8   BZ     24C9   CTLC   24DA
   CTLS   24CE   KPDO   2325   RECCHA F71D   VAIFR  E10D
   VAPAD  E101   VIDEO  F000


   SYMBOL TABLE 3400 343C
   KPDO   2325   BY     24C8   BZ     24C9   CTLS   24CE
   ABORT  24D6   CTLC   24DA   VAPAD  E101   VAIFR  E10D
   VIDEO  F000   RECCHA F71D
```

Korrekturen und Ergänzungen

EC-65K (ec 4):

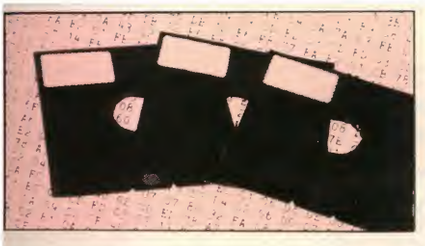
In ec 4 muß es auf Seite 40 in Tabelle 2 bei der Anpassung der FCU-Karte in Punkt (c) heißen: ...Anschluß 28a der 64-poligen Messerleiste...

Bei der Platine der CPU-Karte hat die Platinen-Fabrik eine von uns entfernte Durchkontaktierung wieder eingebaut! Leider waren schon gut 100 Platinen ausgeliefert, bevor uns ein Software-Verteiler dies mitteilte (er hat eine der ersten ausgelieferten Platinen erhalten). Bei allen seitdem ausgelieferten Platinen ist bereits eine Korrektur eingefügt, die nächste Serie wird die Nummer 86207-1 tragen und natürlich in Ordnung sein. Ansonsten: Trennen Sie auf der Lötseite das kurze Leiterbahnstück (knapp ein Millimeter) zwischen C6 und der direkt daneben liegenden Durchkontaktierung auf!

Bei der PSIO-RTC-Karte ist R19 zweimal vorhanden. Richtig ist, wie im Schaltbild zu sehen: Der "R19" bei der Echtzeituhr ist 39 Ohm, der "R19" an der Steckerleiste ist 2k2. Im Bestückungsdruck ist die Steckerleiste JA leider etwas verrutscht, die Platine selbst stimmt trotzdem. Einige Blockkondensatoren konnten entfallen, sind aber noch in der Stückliste enthalten.

Mini-Modem (ec 4):

Bei manchen Nachbauern scheint AI zu oszillieren. Auflösung: Löten Sie einen Kondensator von 270 pF über R9 (270k) ein - aber nur, wenn nötig!



Software-Maus

Für den C64 und andere Computer

Bernd Fasel

Natürlich ist auch hier die Maus selber ein Stück Hardware, und auch einige Widerstände am Userport müssen zur Hardware gerechnet werden. Im Gegensatz zu dem im Hardware-Teil dieses Heftes beschriebenen Maus-Interface kommt man hier jedoch ohne einen speziellen Peripherie-Prozessor aus: Die Auswertung der Maus-Signale bleibt dem Mikroprozessor des Computers überlassen. Das macht das Interface um vieles preiswerter, jedoch auch langsamer und weniger komfortabel als die eigenständige Hardware-Lösung.

Als Alternative zu dem im Hardware-Teil dieses Heftes vorgestellten Maus-Peripherie-Prozessor bietet es sich an, die Signale der Maus über eine Minimal-Hardware per Software auszuwerten. Die hier vorgestellte Lösung wurde für den C64 entwickelt, sie ist jedoch so ausführlich dokumentiert (u.a. durch Flußdiagramme), daß sie leicht auch an andere Computer angepaßt werden kann.

Wie Bild 1 zeigt, werden die Maussignale einfach über den Userport des C64 eingelesen. Die Pinbelegung des Steckers ist natürlich (wie sollte es anders sein) für jede Maus unterschiedlich: Auf eine wirklich einheitliche Schnittstelle haben sich die Hersteller bisher nicht einigen können. Zum Glück sind die Anschlüsse leicht mit einem Voltmeter bestimmbar. Die 470-Ohm-Widerstände dienen nur zum Schutz des Port-ICs im C64. Bei der Initialisierung können nämlich einige der Portleitungen als Ausgänge geschaltet sein, so daß ein Kurzschluß mit dem Mausausgängen nicht auszuschließen ist. Die Widerstände begrenzen dann den Kurzschlußstrom auf einen ungefährlichen Wert.

Die Software wurde in 6502-Assembler geschrieben, Bild 5 zeigt das entsprechende Source-Listing und Bild 6 den zugehörigen BASIC-Lader. Bevor man sich diese Listings im Detail ansieht, sollte man jedoch zunächst einmal weiterlesen. Bild 2 gehört noch zum 'Hardware-Teil' dieses Artikels, bleibt jedoch für den Schluß aufgespart. Das Flußdiagramm Bild 3a zeigt die Auswertung der X- bzw. Y-Koordinate. Das Grundprinzip entspricht dabei der Lösung beim Peripherie-Prozessor im Hardware-Teil dieses Heftes. Am Beispiel X-Koordinate: Zunächst muß eine positive Flanke von PX' erkannt werden. Dazu wird der aktuelle Wert von PX (XNEU) mit dem vor-

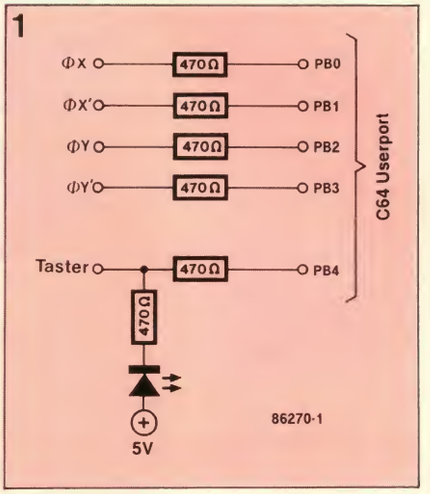
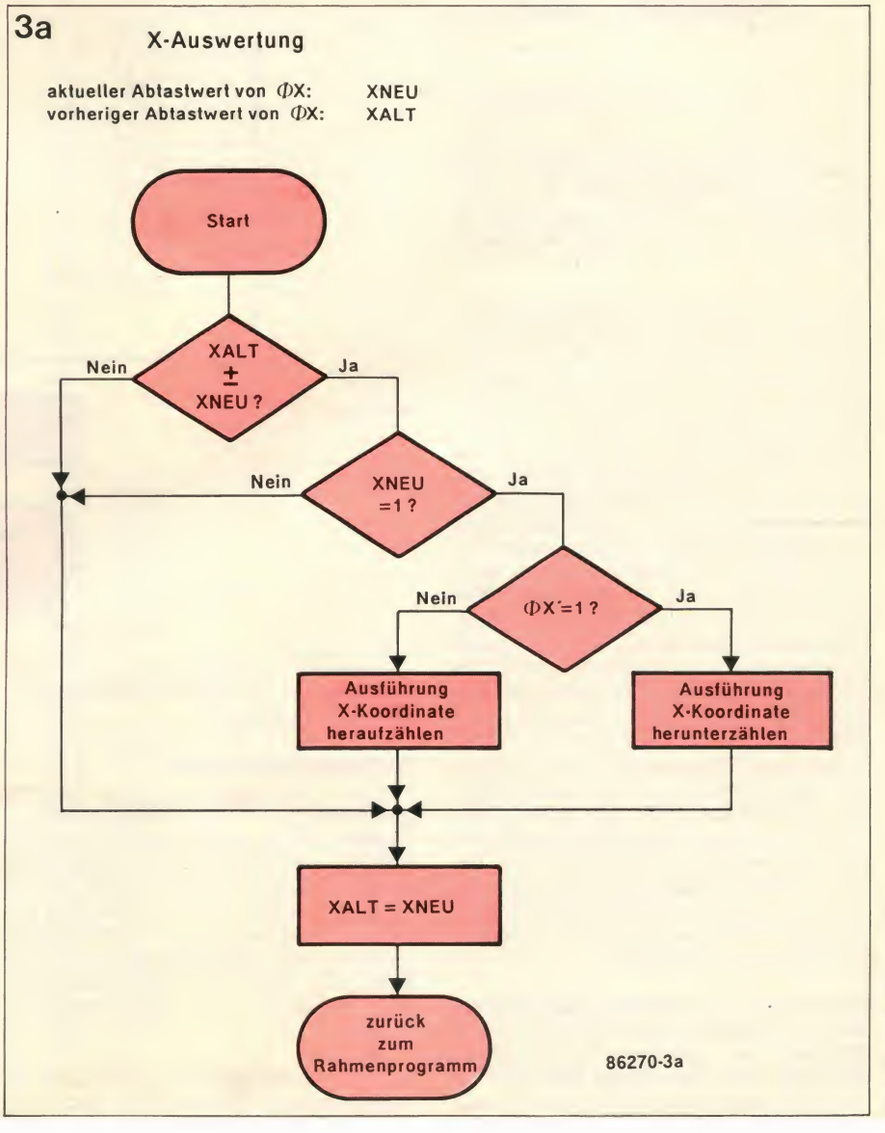
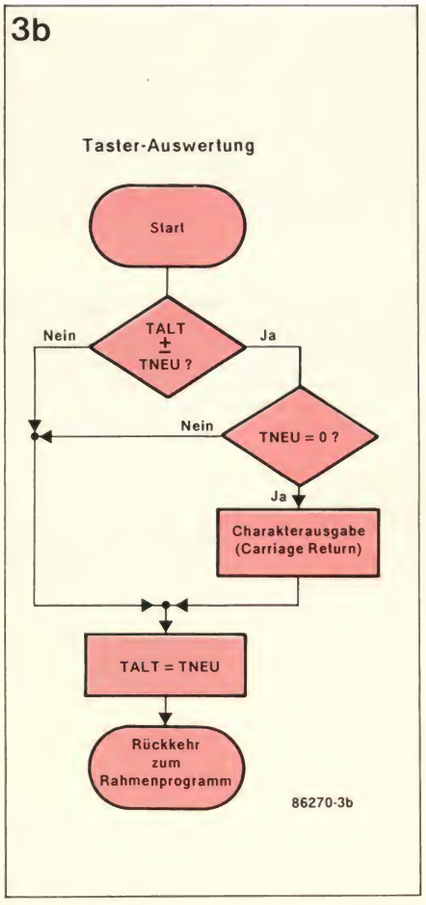


Bild 1. Sechs Widerstände und eine LED: Das ist die gesamte Hardware für die 'Software-Maus'. Fünf der Widerstände dienen lediglich als Schutz für den Userport, der sechste als Strombegrenzung für die LED. Diese signalisiert nur 'Maus-Taster gedrückt' und kann daher notfalls samt Widerstand entfallen.

Bild 3a. Anhand dieses Flußdiagramms erkennt man die Methodik der Auswertung für die X- und Y-Koordinate (hier am Beispiel der X-Koordinate).

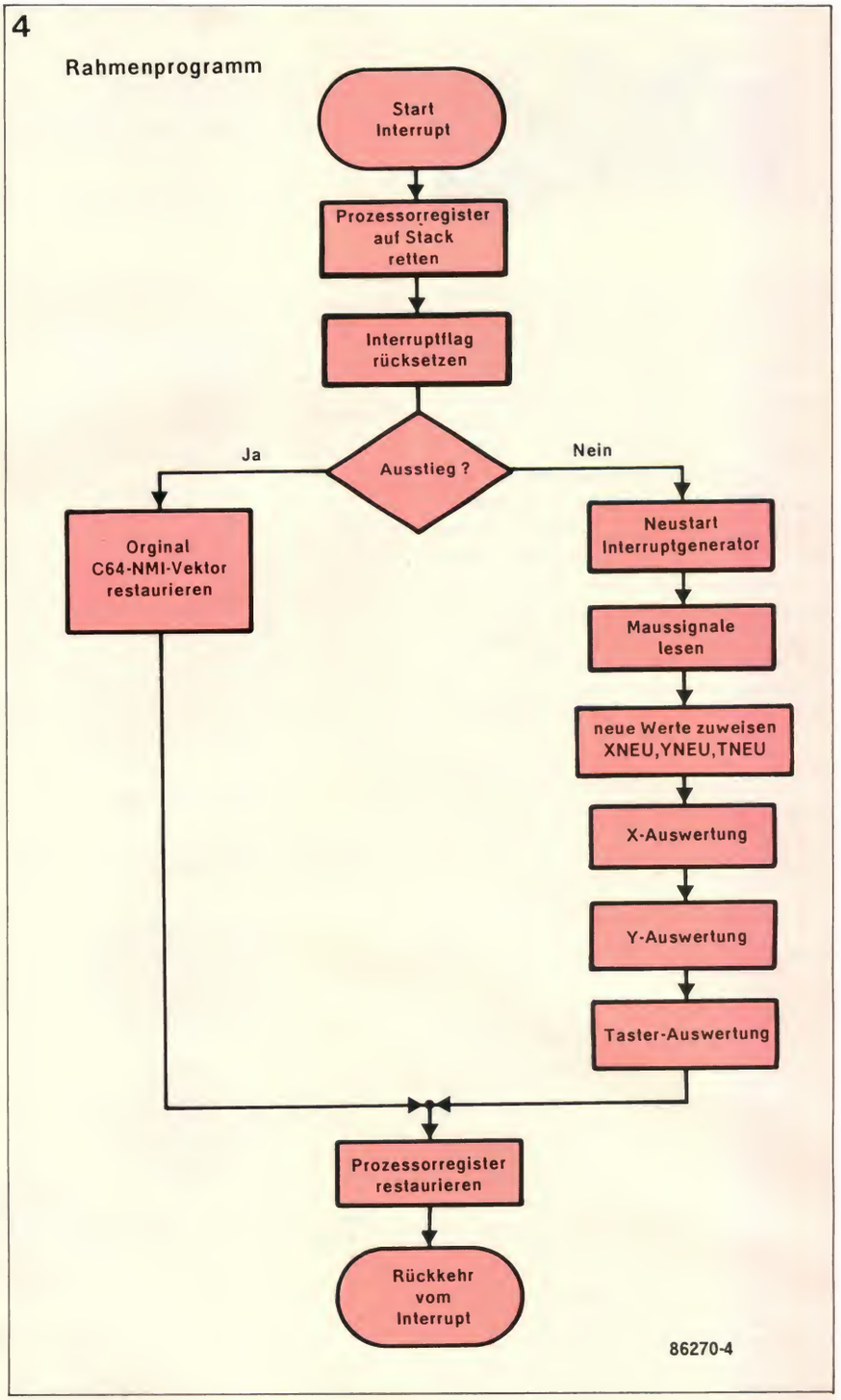




herigen (XALT) verglichen. Ist XNEU ungleich XALT und XNEU gleich '1', liegt eine positive Flanke von ΦX vor. In Abhängigkeit von ΦX' muß dann die X-Koordinate vergrößert oder verkleinert werden.

Ähnlich funktioniert die Auswertung des Tasters (Flußdiagramm Bild 3b). Hier wird die negative Flanke festgestellt, was bedeutet, das der Taster gedrückt wurde. Für diesen Fall soll das Zeichen 'Carriage Return' ausgegeben werden. Der Maus-Taster bekommt also die gleiche Funktion wie die Return-Taste auf der Tastatur. Das ist im übrigen auch das Konzept, wie die X- und Y-Koordinaten auf dem Bildschirm ausgegeben werden, nämlich über Zeichen. Wird zum Beispiel die Maus nach rechts bewegt, wird das Zeichen 'CRSR' (Cursor right) in den Eingangs-Puffer des C64 geschrieben. Es passiert also genau das Gleiche, als würde die Taste für 'Cursor rechts' auf der Tastatur gedrückt. Dies läßt sich besonders einfach erreichen, indem man den Akkumalator des Prozessors (beim C64 der 6510, bei anderen Systemen der 6502) mit dem entsprechenden ASCII-Wert lädt und dann einen 'Jump Subroutine' auf $EB35 macht, wo die entsprechende C64-Routine steht. Dadurch werden Maus und Tastatur gleichwertig, was eine besonders einfache und elegante Softwareschnittstelle zu BASIC-Programmen ermöglicht.

Das eigentliche Programm spaltet sich in einen Initialisierungsteil und einen Hauptteil auf. Beides sind Interrupt-Routinen, die durch einen NMI aktiviert werden. Nach Laden des Programms muß



also zunächst der NMI-Vektor 'verbogen' werden, was durch

POKE792,1:POKE793,128

geschieht, da die Initialisierung bei der Adresse 32769 ($8001) beginnt. Durch Drücken der 'Restore'-Taste muß dann noch ein NMI ausgelöst werden, und schon ist die Maus 'scharf'. Nach der Initialisierung generiert das Port-IC für den Userport (ein CIA 6526) kontinuierlich NMIs. Also auch hier wieder genau das gleiche Prinzip wie beim Peripherie-Prozessor.

Wie das Hauptprogramm aufgebaut ist, kann man dem Flußdiagramm Bild 4 ent-

Bild 3b. Nach diesem Schema wird der Taster der Maus überwacht.

Bild 4. Das Rahmenprogramm der Maus-Software ist in diesem Flußdiagramm schematisch wiedergegeben.

nehmen. Alle Variablen wie XALT oder YALT wurden im Adreßbereich für Sprite 15 abgelegt. Besonders wichtig: Um einen bequemen Ausstieg aus der Routine zu ermöglichen, wurde ein Ausstiegsflag definiert. Es steht in Adresse 966 ($03C6). Flag = 1 bedeutet: Maus aktiv; Flag = 0 bewirkt, daß das Unterprogramm 'Resto' durchlaufen wird. Dieses Unterprogramm restauriert den Original-NMI-Vektor des C64. Vom BASIC aus wird der Ausstieg durch

POKE966,0

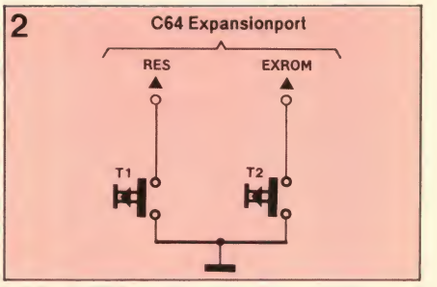
erreicht.

Das Programm wurde absichtlich in den Adreßbereich ab $8001 gelegt, um es direkt in ein EPROM am Expansionsport des C64 'brennen' zu können. Die dazu notwendige Hardware wurde in elektor computing 2 in dem Artikel 'C64-Expansionsport' vorgestellt. Für alle, die den Aufwand für ein EPROM-Modul scheuen, noch ein kleiner Trick: Man kann sich beim C64 durch eine einfache Hardware-Maßnahme einen geschützten Adreßbereich für Assembler-Programme reservieren. Er beginnt bei $8000 und umfaßt 8 KByte, also genausoviel wie ein EPROM-Modul. Dazu muß einfach während des Einschaltens oder während eines Resets die EXROM-Leitung am Expansionsport auf 'Low' gezogen werden: Jetzt leuchtet sofort ein, was mit Bild 2 gemeint ist! Der C64 meldet sich dann mit '30719 Basic Bytes free'. Durch diese Maßnahme ist jedes unbeabsichtigte Überschreiben des Maus-Programmes unmöglich.

Bild 2. Mit diesen beiden Tastern kann man sich einen geschützten Bereich für die Maus-Software schaffen. Näheres dazu im Text.

Bild 5. Das Assembler-Listing der Maus-Software läßt sich anhand der Kommentare und der Flußdiagramme nicht nur leicht überschauen, sondern auch leicht an andere Computer anpassen.

2



5

```
                        0010          .BA $8001                ;STARTADRESSE !
                        0020          .OS
8001- 78                0030 START    SEI
8002- 48                0032          PHA
8003- A9 3B             0035          LDA #$3B                 ;LOWBYTE NMI-ZEIGER
8005- 8D 18 03          0040          STA $0318
8008- A9 80             0050          LDA #$80                 ;HIGHBYTE NMI-ZEIGER
800A- 8D 19 03          0060          STA $0319
800D- A9 01             0070          LDA #%00000001           ;STARTWERT XALT
800F- 8D C0 03          0080          STA $03C0                ;IN SPRITE 15 ABGELEGT
8012- 8D C6 03          0085          STA $03C6                ;FLAG FUER AUSSTIEG
8015- A9 04             0090          LDA #%00000100           ;STARTWERT YALT
8017- 8D C2 03          0100          STA $03C2
801A- A9 10             0102          LDA #%00010000           ;STARTWERT TALT
801C- 8D C4 03          0105          STA $03C4
801F- A9 FF             0110          LDA #$FF                 ;STARTWERTE TIMER CIA
8021- 8D 04 DD          0120          STA $DD04                ;LOWBYTE
8024- A9 10             0125          LDA #$10
8026- 8D 05 DD          0130          STA $DD05                ;HIGHBYTE
8029- A9 81             0140          LDA #%10000001           ;NMI BEI TIMERUNTERLAUF
802B- 8D 0D DD          0150          STA $DD0D
802E- A9 00             0160          LDA #$00                 ;PB AUF EINGANG !
8030- 8D 03 DD          0170          STA $DD03
8033- A9 99             0180          LDA #%10011001           ;START TIMER A
8035- 8D 0E DD          0190          STA $DD0E
8038- 68                0200          PLA
8039- 58                0220          CLI
803A- 40                0250          RTI
803B- 78                0300 NMI      SEI
803C- 48                0305          PHA
803D- 8A                0310          TXA
803E- 48                0320          PHA
803F- 98                0330          TYA
8040- 48                0340          PHA
8041- AD 0D DD          0350          LDA $DD0D                ;CIA-FLAG LOESCHEN
8044- AD C6 03          0352          LDA $03C6                ;AUSSTIEG ?
8047- D0 06             0353          BNE MAUS
8049- 20 D2 80          0355          JSR RESTO                ;WIEDER ALTER NMI-VEKTOR !
804C- 4C CB 80          0357          JMP RAUS
804F- A9 99             0360 MAUS     LDA #%10011001           ;START TIMER CIA
8051- 8D 0E DD          0370          STA $DD0E
8054- AD 01 DD          0380          LDA $DD01                ;WERT HOLEN
8057- A8                0390          TAY                      ;NACH Y-REG !
8058- 29 01             0400          AND #%00000001           ;XNEU
805A- 8D C1 03          0410          STA $03C1
805D- 98                0420          TYA
805E- 29 04             0430          AND #%00000100           ;YNEU
8060- 8D C3 03          0440          STA $03C3
8063- 98                0442          TYA
8064- 29 10             0444          AND #%00010000           ;TNEU
8066- 8D C5 03          0446          STA $03C5
8069- AD C1 03          0450          LDA $03C1                ;START VERGLEICH
806C- 4D C0 03          0460          EOR $03C0
806F- F0 17             0470          BEQ X1
8071- AD C1 03          0480          LDA $03C1                ;POSITIVE FLANKE ?
8074- F0 12             0490          BEQ X1
8076- 98                0500          TYA
8077- 29 02             0510          AND #%00000010           ;RECHTS - LINKS ?
8079- F0 08             0520          BEQ RECHTS
807B- A9 9D             0530          LDA #157                 ;CHARACTER LINKS
807D- 20 35 EB          0535          JSR $EB35                ;AUSFUEHRUNG !
8080- 4C 88 80          0540          JMP X1
8083- A9 1D             0550 RECHTS   LDA #29                  ;CHARACTER RECHTS
8085- 20 35 EB          0555          JSR $EB35
```

```
8088- AD C1 03   0560 X1      LDA $03C1              ;XALT=XNEU
808B- 8D C0 03   0570         STA $03C0
808E- AD C3 03   0600         LDA $03C3              ;START VERGLEICH Y
8091- 4D C2 03   0610         EOR $03C2
8094- F0 17      0620         BEQ Y1
8096- AD C3 03   0630         LDA $03C3
8099- F0 12      0640         BEQ Y1
809B- 98         0650         TYA
809C- 29 08      0660         AND #%00001000         ;OBEN - UNTEN ?
809E- F0 08      0670         BEQ OBEN
80A0- A9 11      0680         LDA #17                ;CHARACTER UNTEN
80A2- 20 35 EB   0690         JSR $EB35
80A5- 4C AD 80   0700         JMP Y1
80A8- A9 91      0710 OBEN    LDA #145               ;CHARACTER OBEN
80AA- 20 35 EB   0720         JSR $EB35
80AD- AD C3 03   0730 Y1      LDA $03C3              ;YALT=YNEU
80B0- 8D C2 03   0740         STA $03C2
80B3- AD C5 03   0745         LDA $03C5              ;START VERGLEICH T
80B6- 4D C4 03   0750         EOR $03C4
80B9- F0 0A      0755         BEQ T1
80BB- AD C5 03   0760         LDA $03C5
80BE- D0 05      0765         BNE T1
80C0- A9 0D      0770         LDA #13                ;CHARACTER <CR>
80C2- 20 35 EB   0775         JSR $EB35
80C5- AD C5 03   0780 T1      LDA $03C5              ;TALT=TNEU
80C8- 8D C4 03   0785         STA $03C4
80CB- 68         0800 RAUS    PLA
80CC- A8         0810         TAY
80CD- 68         0820         PLA
80CE- AA         0830         TAX
80CF- 68         0840         PLA
80D0- 58         0845         CLI
80D1- 40         0850         RTI
80D2- A9 47      1000 RESTO   LDA #$47               ;LOWBYTE C64-NMI
80D4- 8D 18 03   1010         STA $318
80D7- A9 FE      1020         LDA #$FE               ;HIGHBYTE C64-NMI
80D9- 8D 19 03   1030         STA $319
80DC- 60         1040         RTS
                 1100         .EN
END OF ASSEMBLY!

 ---  LABEL FILE:  --

MAUS =804F              NMI =803B
OBEN =80A8              RAUS =80CB
RECHTS =8083            RESTO =80D2
START =8001             T1 =80C5
X1 =8088                Y1 =80AD

//0000,80DD,80DD
Ü
```

Bild 6. Dieses BASIC-Programm ist eigentlich der einzige Teil der Maus-Software, der speziell auf den C64 zugeschnitten ist: Es lädt die Maus-Software als Maschinensprache-File in den Speicher des C64.

6

```
10 DATA 120, 72, 169, 59, 141
20 DATA 24, 3, 169, 128, 141
30 DATA 25, 3, 169, 1, 141
40 DATA 192, 3, 141, 198, 3
50 DATA 169, 4, 141, 194, 3
60 DATA 169, 16, 141, 196, 3
70 DATA 169, 255, 141, 4, 221
80 DATA 169, 16, 141, 5, 221
90 DATA 169, 129, 141, 13, 221
100 DATA 169, 0, 141, 3, 221
110 DATA 169, 153, 141, 14, 221
120 DATA 104, 88, 64, 120, 72
130 DATA 138, 72, 152, 72, 173
140 DATA 13, 221, 173, 198, 3
150 DATA 208, 6, 32, 210, 128
160 DATA 76, 203, 128, 169, 153
170 DATA 141, 14, 221, 173, 1
180 DATA 221, 168, 41, 1, 141
190 DATA 193, 3, 152, 41, 4
200 DATA 141, 195, 3, 152, 41
210 DATA 16, 141, 197, 3, 173
220 DATA 193, 3, 77, 192, 3
230 DATA 240, 23, 173, 193, 3
240 DATA 240, 18, 152, 41, 2
250 DATA 240, 8, 169, 157, 32
260 DATA 53, 235, 76, 136, 128
270 DATA 169, 29, 32, 53, 235
280 DATA 173, 193, 3, 141, 192
290 DATA 3, 173, 195, 3, 77
300 DATA 194, 3, 240, 23, 173
310 DATA 195, 3, 240, 18, 152
320 DATA 41, 8, 240, 8, 169
330 DATA 17, 32, 53, 235, 76
340 DATA 173, 128, 169, 145, 32
350 DATA 53, 235, 173, 195, 3
360 DATA 141, 194, 3, 173, 197
370 DATA 3, 77, 196, 3, 240
380 DATA 10, 173, 197, 3, 208
390 DATA 5, 169, 13, 32, 53
400 DATA 235, 173, 197, 3, 141
410 DATA 196, 3, 104, 168, 104
420 DATA 170, 104, 88, 64, 169
430 DATA 71, 141, 24, 3, 169
440 DATA 254, 141, 25, 3, 96
450 FORI= 32769 TO 32988 :READJ
460 POKEI,J:NEXTI:PRINT"FERTIG"
470 END
```

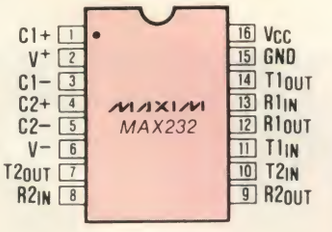
MARKT

MAX232 — ein V24-Leitungstreiber/-empfänger mit nur einer Betriebsspannung

Der MAX232 ist ein doppelter V24-Treiber/Empfänger, der alle Spezifikationen nach CCITT V.24 (EIA RS232C) mit nur einer einzigen Versorgungsspannung von +5 V erfüllt. Damit werden die beiden zusätzlichen Versorgungsspannungen von +12 V und —12 V zur Speisung der V24-Schnittstellen überflüssig. Der MAX232 besitzt zwei integrierte Spannungswandler nach dem Prinzip der Ladungspumpe, die Spannungen von +10 V und —10 V aus der +5-Versorgungsspannung erzeugen.
Im IC befinden sich vier Pegelwandler, von denen zwei als V24-Treiber arbeiten; sie setzen den TTL/CMOS-Eingangspegel auf V24-Ausgangsspannungen von ±9 V um. Die beiden anderen Pegelwandler sind V24-Empfänger, die die V24-Spannungspegel auf TTL-CMOS-Eingangspegel umsetzen. Diese Empfänger haben eine typische Schwellenspannung von 1,3 V, eine typische Hysterese von 0,5 V und können mit Eingangsspannungen bis ±30 V arbeiten.

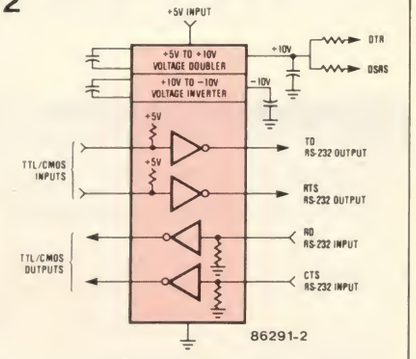
Spezial-Electronic
Kreuzbreite 14
Postfach 1308
3062 Bückeburg

1



86291-1

2



86291-2

FX-kompatibler Drucker von Riteman

Der Drucker Riteman F+ ist ein neuer Matrixdrucker, der an praktisch alle gängigen HCs und PCs angeschlossen werden kann. Er ist softwarekompatibel zu den Druckern der FX-Serie von Epson. Augenfälligstes Merkmal ist die flache Papierführung, bei der das Papier im Drucker nicht mehr gebogen oder in anderer Weise veformt wird. So können zum Beispiel Aufkleber, Formulare, Durchschlagsätze, Postkarten usw. problemlos bedruckt werden. Auch umgekehrter Papiertransport ist mit diesem Prinzip ohne Papierstau oder ähnliche, sonst gelegentlich auftretenden Probleme möglich.
In der Standardversion wird der F+ mit einer Centronics-Parallelschnittstelle geliefert. Die Verbindungen zu davon abweichenden Computer-Druckerausgängen stellen entweder Spezialkabel oder aufsteckbare Inteface-Module her. Dazu gehört auch ein IEEE488-Interface mit einem bis 128 KByte ausbaubaren Puferspeicher.

Einzel-Bezugsquelle z.B.:
Wiesemann
Mikrocomputertechnik
Winchenbachstr. 3a
5600 Wuppertal 2

Aus XT mach AT

Wem die Leistung seines IBM-PCs vom Typ XT oder eines entsprechenden Kompatiblen nicht genügt, der kann sich seinen AT selber machen: Das ist die Idee, die hinter der BULLET-286-Platine steckt. Es handelt sich praktisch um einen Ein-Platinen-Computer mit der CPU 80286 auf 8 oder 10 MHz Taktfrequenz, man erreicht damit die bis zu zwölffache Verarbeitungsgeschwindigkeit eines 8088-Prozessors unter den Standard-Betriebssystemen (PC-DOS, Concurrent PC-DOS). Auf der Platine befindet sich ein Sockel für den Coprozessor 80287.
Die Abmessungen, Befestigungspunkte, die Lage der Anschlüsse und alle Anschlußstecker sind identisch mit dem "IBM-XT-Motherboard", so daß die Platine gegen dieses Motherboard einfach ausgetauscht werden kann.

Technische Daten:

- 256 KByte oder 1 MByte auf der Platine, Zugriff ohne Wartezyklen, Paritätsprüfung, volle 16-Bit-Speicherbusbreite
- 8237-5 Vier-Kanal-DMA-Kontroller
- High-Performance-Buskontroller ohne Wartezyklen
- Synchronisierung mit dem 4,77-MHz-Erweiterungsbus des XT-Standards
- 8 "PC/XT"-kompatible, gepufferte Erweiterungs-Steckplätze ("Slots")
- Betriebssysteme: PC-DOS, Concurrent PC-DOS, XENIX, VENIX, OASIS, PICK, PROLOGUE, MUMPS, QNIX, SMC, COHEERENT, BOS, UCSD, Turbo-DOS
- Alle bekannten Software-Programme für die genannten Betriebssysteme, zusätzlich Branchen- und Individuallösungen möglich

Bei Bedarf wird ein Umtausch- und Austausch-Service angeboten, das alte Motherboard kann in Zahlung genommen werden. Auch die Lieferung von Komplettsystemen in individueller Zusammenstellung ist möglich. Nähere Informationen sind erhältlich beim BULLET-Fachhändler

HERGET Computersysteme
Marktstr. 36-37
1000 Berlin 51
Tel. 030/492 50 70

BÜCHER MARKT

Computergestützter Fremdsprachenunterricht

Herausgeber: G. Neuner
128 Seiten, 19,80 DM
ISBN 3-468-49434-3

Die Arbeit mit Computern im Fremdsprachenunterricht, in angloamerikanischer Abkürzungsmanier "CALL" (Computer-Assisted Language Learning) genannt, ist das Thema dieses Buchs, das aus dem dafür sach- und fachkompetenten Verlag Langenscheidt stammt. Wie so vieles ist auch CALL eine in den USA entwickelte Errungenschaft, und wie immer stößt sie bei uns (bisher) verbreitet auf Skepsis. Wie dem auch sei: Im vorliegenden Buch geben CALL-erfahrene Autoren aus den USA und der Bundesrepublik eine Einführung in die Geheimnisse dieser neuen Computer-Anwendung.
Die Verfasser sind in den Bereichen der Hochschulen, der Goethe-Institute und des Verlagswesens tätig; sie beleuchten das Für und Wider von verschiedenen Seiten. Das reicht von den Grundlagen der Computertechnologie bis zu den unterschiedlichen Typen von Lernprogrammen, von Anwendungsbeispielen für Grammatik bis zum Ausblick in die fernere Zukunft. Wer außerdem eine linguistische Analyse der Computer-Wortschatzarbeit sucht, der wird hier fündig.

Langenscheidt-Verlag
Postfach 40 11 20
Neusser Str. 3
8000 München 40

Die 8087/80287 numerischen Prozessor-Erweiterungen für 8086/80286-Systeme

von Klaus-Dieter Thies
374 Seiten, 69,— DM
ISBN 3-921803-53-5

Numerische Software auf Mikrocomputern leidet unter den Fehlern und Laufzeiten der Gleitpunktarithmetik auf den heutigen 8/16-Bit-Prozessoren. Die Fehler sind problematisch, weil Rundungen, Bereichsüberschreitungen usw. uneinheitlich behandelt werden und daher nur schwer mit den Mitteln der numerischen Mathematik abschätzbar sind. Untragbare Laufzeiten sind die Folge des Versuchs, auf Ganzzahlprozessoren mit Hilfe von Software Gleitpunktarithmetik zu installieren. Der amerikanische Verband IEEE hat deshalb eine Standardarithmetik für binäre Gleitpunktzahlen in Mikroprozessoren ausgearbeitet, deren Hardwarerealisierungen im 8087 und 80287 vorliegen und als Coprozessoren zum 8086/80186/80286 laufen. Die Ensemble 8086/8087 bzw. 80186/8087 bzw. 80286/80287 erscheinen dem Benutzer als nur eine Maschine, die allerdings über sechs weitere Datentypen und 120 zusätzliche Numerikbefehle für technisch-wissenschaftliche Programme verfügen.
In diesem Buch erhält der Leser eine detaillierte Funktionsbeschreibung des Prozessors 8087/80287 sowie Hintergrundwissen des Einsatzes seiner Software und eine umfangreiche Beispielsammlung. K.-D. Thies beschreibt ausführlich und datenblattgenau die Implementierung des IEEE-Standards im 8087/80287. Am Beispiel eines 8086/8087-"Gespanns" werden Hardware-Aufbau, Funktion und Programmierung mit dem Macro-Assembler ASM86 gezeigt. Alle Arithmetikbefehle können in sämtlichen Phasen ihrer Ausführung an Systembildern verfolgt werden.

te-wi Verlag GmbH
Theo-Prosel-Weg 1
8000 München 40

BASIC-Wegweiser für MSX-Computer

von Ekkehard Kaier
368 Seiten, 48,— DM
ISBN 3-528-04356-3

Dieses Buch aus der "Wegweiser"-Reihe des Vieweg-Verlages ist in drei Teile untergliedert: Der erste Teil vermittelt aktuelles, allgemeingültiges Grundwissen aus den Bereichen Datenverarbeitung und Informatik. Das Kapitel beginnt mit der Erklärung von Grundbegriffen, führt den Leser an Daten- und Programmstrukturen heran, erklärt in Kurzform gängige Betriebssyteme und Hochsprachen und geht dann auf die Entwicklung von Anwenderprogrammen ein.
Das zweite Kapitel ist der detaillierten Beschreibung des zum neuen Home-Computer-Standard erklärten MSX-Basic gewidmet. Dabei werden unter anderem folgende Möglichkeiten des MSX-Standards herausgestrichen: Strukturiertes Programmieren, Ausgabeformatierung, Stringsuche, Tabellenverarbeitung, Sortieren und Mischen, Error-Trapping, Programm-Overlay, Grafik, Farbeinstellung, Sprites und Musik. Sämtliche Befehle des MSX-Basic werden an einfachen Beispielen beschrieben.
Im dritten Teil findet der Leser einen abgeschlossenen Programmierkurs mit folgenden grundlegenden Basic-Anwendungen: Programme mit den wichtigsten Ablaufstrukturen (Folge-, Auswahl-, Wiederholungs- und Unterprogrammstrukturen), Verarbeitung von Texten und Tabellen, maschinennahe Programmierung, Suchen, Sortieren, Mischen und Gruppieren von Daten, Sequentielle und Direktzugriff-Dateien sowie das Programmieren von Grafiken, Sprites und Musik. Alles in allem ein nützliches, verständlich geschriebenes Buch, das dem Leser hilft, den Komplex MSX-Basic in den Gesamtrahmen der Datenverarbeitung bzw. Informatik einzuordnen.

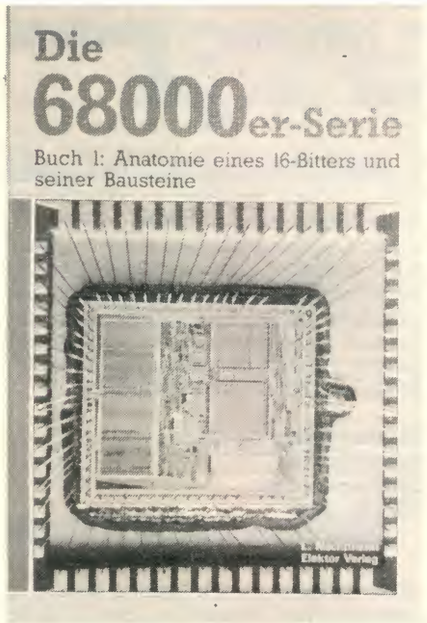
Friedr. Vieweg & Sohn
Verlagsgesellschaft mbH
Postfach 5829
6200 Wiesbaden 1

COMPUTERKUNST

von M.L. Prueitt
256 Seiten, 49,50 DM
ISBN 3-89028-041-2

Das Buch stellt einen Bericht über einige der hervorragenden Kunstwerke dar, die von Künstlern mit Hilfe von Computern geschaffen worden sind. Der Anfänger wie der Berufskünstler kann die Seiten studieren, um zu erfahren, was in diesem neu aufblühenden Bereich geschieht. Das Buch enthält Bilder von historischer Bedeutung aus dieser Pionierzeit der rechnergestützten Kunst. Außerdem findet man in dem Band neue, der Welt bisher noch nicht bekannte Kunstwerke.
Gegliedert ist das Buch in zwei Teile: "Der Gegenstand" und "Die Galerie". Der erste Teil befaßt sich mit dem Thema der Einführung des Computers als Werkzeug des Künstlers. Der Leser (oder besser: der Betrachter) findet hier zahlreiche Computergraphiken, die die Prinzipien veranschaulichen. Der zweite Teil stellt eine Sammlung einiger der schönsten mit dem Computer geschaffenen Kunstwerke dar, die es bisher gibt. Die Werke sind nach Herkunft oder Art in zwei Gruppen zusammengefaßt, mit einem Minimum an Text, so daß die Bilder für sich selbst sprechen können.
Ein Buch, das nicht nur Besitzern eines grafikfähigen EC-65(K) neue Anstöße zu eigenem künstlerischem Schaffen zu geben vermag!

Mc Graw-Hill
Book Company GmbH
Lademannbogen 136
2000 Hamburg 63



Die 68000er-Serie

Buch 1: Anatomie eines 16-Bitters und seiner Bausteine
Buch 2: Alle Instruktionen auf einen Blick

von L. Nachtmann
Buch 1: 232 Seiten, 42,50 DM
ISBN 3-921608-42-2
Buch 2: in Vorbereitung

Zwei neue Bücher aus dem eignen Haus, von einem Autor, der Ihnen, liebe EC-Leser, wohlbekannt sein dürfte. Loys Nachtmann ist ein 68000-Vollblut-Fan, und diese Sprache sprechen seine beiden neuesten Bücher sehr deutlich.
Das erste Buch beginnt mit einem einführenden Kapitel über die CPU-Architektur und die CPU-Signale. Dort werden unter anderem die Organisation der Register und die vom Prozessor vorgegebene Speicherorganisation erklärt. Im zweiten Kapitel geht es um Steuerleitungen und Busse, also um Hardware, wobei detaillierte Zeitdiagramme für alle relevanten Signale nicht fehlen. Mit den zum 68000 passenden Peripherie-Chips befaßt sich das dritte Kapitel. Außer den neuen 68XXX-Bausteinen sind hier auch die altbewährten 8-Bit-ICs der 65XX-/68XX-Serie berücksichtigt, denn sie fügen sich fast nahtlos in die 68000-Familie ein. Kapitel 4 ist wohl der wichtigste Teil des ersten Buchs: Beschrieben werden die Adressierungsarten und ihre erweiterten Möglichkeiten einschließlich aller Besonderheiten. Der Leser lernt an Hand zahlreicher Beispiele, wie man effizient mit ihnen umgeht.
Im Anhang erklärt der Autor schließlich den Aufbau und die Eigenschaften des 68008, einer Version des 68000, die nach außen hin als 8-Bit-Prozessor arbeitet.
Das zweite Buch, das zur Zeit noch in Vorbereitung ist, beschreibt die 68000er-Instruktionen Schritt für Schritt. Da zusätzlich alle wichtigen Informationen tabellarisch gelistet sind, wird sich dieses Buch insbesondere als Nachschlagewerk für die tägliche Arbeit des 68000-Programmierers eignen.

Elektor-Verlag
Süsterfeldstr. 25
5100 Aachen

takte *** Clubs *** Infos *** Kontakte *** Clubs *** Infos

Der Abdruck von Kontaktwünschen und Computerclub-Infos in dieser Rubrik erfolgt kostenfrei. Kontaktwünsche müssen sich dabei beschränken auf Name, Adresse, Tel.-Nr. und zwei weitere Informationen (zum Beispiel: Computertyp, bevorzugte Programmiersprache).

Kontaktwunsch:

Kontakt zu EC-68-Anwendern sucht:

Werner Duders
Am Heidekotten 30
4500 Osnabrück
Tel. 0541/38 71 35

Clubs:

CTM-User-Vereinigung

Eine große Anzahl von Anwendern von CTM Hard- und Software haben sich in einer User-Group zusammengeschlossen. Ziel dieser Vereinigung:

- Austausch von Informationen
- Gegenseitiger Rat und Hilfe bei auftretenden Problemen
- Laufende Information über CTM Hard- und Software-Entwicklung
- Gemeinsames Auftreten in der Öffentlichkeit und gegenüber CTM

Zu diesem Zweck fanden bisher mehrere Veranstaltungen in Heidelberg, Hannover und Duisburg statt.

Für interessierte CTM-Anwender steht das Sekretariat der User-Group, Tel. 02102/496-261, oder der Sprecher, Herr Detlev Groß, Kindernothilfe in Duisburg, Tel. 0203/7789-104, telefonisch zur Verfügung.

Space 86 mit 65816-CPU

Ein weiterer Rechner mit 65816-CPU: Der Space 86 von IBS macht aus einem normalen Apple einen 65816-Rechner. Die Grundidee ist ähnlich wie beim EC-65K: Statt eines ganz neuen Rechners wird auf dem Vorhandenen aufgebaut, und dennoch ergeben sich völlig neue Möglichkeiten. Die Taktfrequenz beträgt 4 MHz, 256 KByte DRAM (!) sind "on board", beim Zugriff auf die Peripherie und die unteren 64 KByte des Apple wird der Takt heruntergeschaltet (kommt uns auch sehr bekannt vor: Gleiches Problem - gleiche Lösung, unabhängig voneinander fast gleichzeitig entwickelt. Die Red.), so daß alle vorhandenen Peripheriekarten weiterverwendet werden können. Ähnlich sieht es bei der Software aus: Die vorhandene Apple-Software kann weiterverwendet werden, außerdem gibt es 65816-Software im EPROM (und die ec-Redaktion vermittelt den Kontakt zu ees, den warum sollte der Space 86 nicht auch ein COP 65K-Rechner werden. Siehe Artikel "EC-65K / 65816: Software" im Innenteil des Heftes. Die Red.).
Nähere Informationen sind erhältlich bei:

IBS Computertechnik
Olper Str. 10
4800 Bielefeld 14
Tel. 0521/44 40 32

Was in diesem Heft fehlt...

... ist die Speichererweiterung für den Big-Board II. Grund: Die 256 KByte-Erweiterungsplatine ist ausverkauft, an ihrer Stelle wird nun gleich eine Platine mit 1 MByte treten. Pech nicht nur für die Big-Board-Nachbauer, sondern auch für die Redaktion: Der Artikel war nämlich schon fertig!

... ist die Fortsetzung des EDMO-Artikels aus ec 4. Grund: Der Autor hat die Zeit genutzt, um den zweiten Teil noch einmal zu überarbeiten. Und dann hat die Bundespost eine gute Woche gebraucht, um den Brief von Westberlin nach Aachen zu beförden: Er kam leider zwei Tage zu spät an, der Artikel konnte nicht mehr gesetzt werden.

... ist das 6502-Modem-Programm. Es ist "Spitze" geworden: Menügesteuert, schnell (Mischung Maschinensprache- und BASIC-Teile), mit Upload und Download, sogar das Christensen-Protokoll ist schon eingebaut und so weiter. Nur hätten die Listings soviele Heftseiten gefüllt, daß es in diesem Heft leider nicht paßte. Daher geht das Programm mit Dokumentation auf Diskette zunächst an die Software-Verteiler, der Abdruck für die "Allgemeinheit" der 6502-Anwender erfolgt dann in ec 6.

... ist ein Praxisbericht (so wird unsere Rubrik "Test" künftig heißen) über eine spezielle Tastatur für den EC-65 und EC-65K von JB-Elektronik. Grund: Sie wird nochmals verbessert, darauf warten wir.

... sind die Platinen-Layouts der Z80-Karte und der neuen SRAM-Karte. Grund: Mindestens 80% (wahrscheinlich: über 95%) der Nachbauer kaufen bei doppelseitig-durchkontaktierten Platinen EPS-Platinen. Zudem sind beide Platinen bei maximaler Taktgeschwindigkeit zeitkritisch, eine Verzinnung der Leiterbahnen (bei EPS-Platinen automatisch) ist unumgänglich! Und dann ist da noch das Platzproblem, denn eigentlich müßten auch die Rückseiten freibleiben. Zusatzinfo: Bei elektor wird gerade mit Platinenfolien experimentiert. Schreiben Sie uns doch einfach mal Ihre Meinung dazu.

... sind Nachrichten (statt Gerüchte) über den 65816-Rechner von Apple. Grund: Auf die warten wir selber...

EC-65-Softwareverteiler:

Der Softwareverteiler wird zur Zeit verbessert: Zur Entlastung der Redaktion werden Zentral-Verteiler die einzelnen Verteiler in bestimmten Regionen versorgen (Rechnen Sie mal: Es sind fast 60 Verteiler inzwischen, und mit der CP/M-Free-Soft, dem FORTH, dem Modemprogramm und den EC-65K-Patchdisketten stehen rund 30 neue Disketten zur Verteilung an...). Deshalb in diesem Heft nur eine Reihe von neuen Adressen, in ec 6 wird es dann entweder eine komplette neue Tabelle geben oder eine Zusammenfassung aller Ergänzungen und Änderungen zu der in ec 3 veröffentlichten Tabelle.
Damit wir nicht nur alle neuen Verteiler, sondern auch alle Änderungen bei Verteilern "der ersten Stunde" sauber und ohne zu großen Zeitaufwand zusammenbekommen, starten wir erneut eine Fragebogen-Aktion. Wir bitten alle Leser, die beim Software-Verteiler neu mitmachen wollen, diesen Fragebogen per Postkarte anzufordern. Dies gilt auch für Leser, die sich zum Beispiel im Getümmel einer Messe oder in einem mit technischen Fragen verbundenen Brief als Verteiler gemeldet haben und deren Namen bisher nicht in einem ec-Heft aufgetaucht sind. Bitte fordern Sie mit dieser Postkarte nur den Fragebogen an, stellen Sie technische Fragen in einem gesonderten Brief. Senden Sie die Postkarte an

Elektor Verlag GmbH
Redaktion ec
Süsterfeldstr. 25
5100 Aachen

Alle Verteiler, deren Namen bereits in einem ec-Heft genannt wurden, erhalten den Fragebogen automatisch!

Hier also erstmal die neuen Adressen (am besten ersten Kontakt per Postkarte aufnehmen), diesmal (noch) nicht nach Postleitzahlen geordnet:

Nino Pistorius
Worpsweder Str. 35
2800 Bremen 1

Volker Meyer
Cesar-Klein-Ring 14
2000 Hamburg 60

Arnulf Oppl
Reitmoorstr. 23
8000 München 22

Herbert Gerdener
Bürgerweg 22
4905 Spenge

Stefan Peters
Planckstr. 65
4300 Essen 1

Farbmonitor und Netzteil

Immer wieder erreichen die Redaktion Fragen nach dem Farbmonitor, mit der wir auf der Dortmunder Hobbytronic die Grafik auf dem EC-65K demonstriert haben, und nach dem auf Fotos in ec 4 abgebildten Schaltnetzteil. Der Monitor ist ein CPD-1301 E von Sony und gehört zu einer Serie von Farbmonitoren mit Trinitron-Röhre. Die Trinitron-Röhre besticht durch ihre Farbbrillianz und hohe Auflösung, sie ist jedoch in der Herstellung aufwendiger und daher etwas teurer als normale Farbröhren. Das Netzteil ist das SMP1 von Elba. Sowohl Sony als auch Elba liefern diese Geräte nicht an Einzelkunden, jedoch sind die Sony-Monitore und das SMP1 einzeln erhältlich beim Hersteller des EC-65-Gehäuses, der Firma

Feltron-Zeissler
Auf dem Schellerod 22
5210 Troisdorf

Computer analysiert Schäden an Automatikgetrieben

Es ist geschrieben in der Programmiersprache MULISP und läuft auf Personalcomputern vom Typ IBM-PC-XT: Das von Renault und CAP GEMINI gemeinschaftlich entwickelte und realisierte Expertensystem zur Unterstützung von Diagnose und Reparatur im Falle von Defekten an automatischen Schaltgetrieben. Das für rund 9000 französische Renault-Vertragswerkstätten bestimmte System wurde Anfang Oktober auf dem Pariser Autosalon am Beispiel eines Renault 25 GTX der Öffentlichkeit vorgestellt. Renault plant, es auch deutschen Vertragswerkstätten zur Verfügung zu stellen.

Weil die Fehlerfindung und -behebung bei Schäden am Automatikgetriebe in aller Regel diffizil und darum zeit- und kostenaufwendig ist, zog der staatliche Automobilkonzern Renault das bekannte Softwarehaus CAP GEMINI zu Rate, um vermittels der sogenannten "Künstlichen Intelligenz" (KI) diesen Reparaturen den Charakter der jedesmal neuen Einmaligkeit zu nehmen und auf diese Weise den Kostendruck bei den Werkstätten zu verringern. Renault, das die Automatikgetriebe für seine verschiedenen Fahrzeugmodelle selbst herstellt, ließ von CAP GEMINI eine Software entwickeln, die in drei Betriebsarten abläuft.

Am unmittelbarsten wirksam wird sie im Beratungsmodus. Dieser Programmteil ist auf die Probleme des KFZ-Mechanikers vor Ort zugeschnitten. In einem ausgeklügelten Dialog mit dem System wird der Mechaniker dahin gebracht, alle beobachteten Fehlersymtome schrittweise einzugeben. Sodann wird er vom Computer aufgefordert, bestimmte technische Gegebenheiten zu überprüfen und im betroffenen Getriebebereich einige ergänzende Tests vorzunehmen. Als Resultat erhält der Mechaniker vom System eine Fehlerdiagnose und konkrete Reparaturhinweise.

Im Archiviermodus bietet das System dem Management des Technischen Kundendienstes die Möglichkeit, die aufgetretenen Getriebedefekte nach Art und Häufigkeit statistisch zu untersuchen. Der Update-Modus versetzt das Programmsystem in die Lage, permanent auf neuartige technische Problemstellungen eingestellt zu werden - und natürlich auf neuentwickelte Schaltgetriebe.

Wie man bei Renault betont, können die Vertragswerkstätten schon aus rechtlichen Gründen nicht gezwungen werden, dieses Expertensystem zu übernehmen. Man hält es jedoch für so kostensparend, daß demnächst beispielsweise auch deutsche Renault-Vertragshändler zur Vorführung nach Paris eingeladen werden sollen.

Weitere Informationen sind erhältlich bei

CAP GEMINI Deutschland GmbH
Ridlerstr. 35a
8000 München 2
Tel. 089 / 51 99 0

Europäische Chips

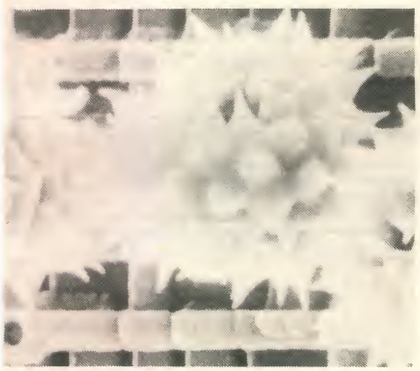
Japanern und Amerikanern Paroli zu bieten - das ist ein erklärtes Ziel vieler europäischer Elektronik-Hersteller, seien sie nun groß oder klein. Zu den "etwas größeren" gehört bekanntlich der Siemens-Konzern. Er hat es sich rund 220 Millionen Mark kosten lassen, in Villach im österreichischen Bundesland Kärnten sein dortiges Halbleiterwerk um eine 4000 qm große Fertigungshalle ("Villach II") auf 29.000 qm zu erweitern.

Mit dem Jahreswechsels 1984/85 wurde dort die Produktion von 256-KBit-DRAMs aufgenommen. Rund 1200 Mitarbeiter sind dort beschäftigt, Tendenz steigend.

Villach II arbeitet mit "Reinheitsklasse 10": Pro Kubikfuß Luft (=27 Liter) sind höchtens 10 Partikel von maximal 0,1 Mikrometer (!) Durchmesser zugelassen. Um die Luft laufend zu waschen und zu filtern, werden pro Stunde 2 Millionen Kubikmeter Luft umgewälzt.

Außer den RAMs sollen dort auch andere ICs mit Strukturen von 1,5 Mikrometer in NMOS- und in CMOS-Technologie gefertigt werden, darunter der 16-bit-Prozessor 80256 und der zugehörige ADMA-Baustein 82258.

Recht beeindruckend ist der Größenvergleich zwischen den beiden Fotos: Dem "Klotz" der neuen Halle links oben in dem einem Foto und den dort gefertigten ICs: Das zweite Foto ist mit einem Elektronenraster-Mikroskop aufgenommen und zeigt 3 Blütenpollen eines Gänseblümchens, Durchmesser knapp 20 Mikrometer, die mehrere Leiterbahnen in einem 256-KBit-DRAM überdecken.



EPS-Service

Anzeigen

Bezugsbedingungen

EPS-Prints werden vom Fachhandel geführt. Sind die Platinen im Fachhandel nicht vorrätig, so können sie auch durch Vorauszahlung nebenstehender Preise zuzüglich Versandkosten (DM 2,50 je Bestellung) auf Postscheckkonto Köln 22 97 44-507, Elektor Verlag, 5133 Gangelt, unter Angabe der Bestellnummer bezogen werden. Kein Nachnahmeversand.
Auslieferung für die Schweiz: Thali AG, CH-6285 Hitzkirch, Tel. 041/85 28 28, Ladenverkauf: Zürich, Schaffhauserstr. 24.
Auslieferung für Österreich: Fachbuch-Center ERB, Amerlingstr. 1, 1061 Wien, Tel.: 0222/56 62 09.

Elektor-Kopien-Service

Wenn Ihnen zum Aufbau der Schaltung die Bauanleitung fehlt, können Sie den EKS (Elektor-Kopien-Service) in Anspruch nehmen. Sie erhalten bei Angabe der Platinen-Nummer und bei Überweisung von DM 5,50 entweder die dazu gehörende Elektor-Ausgabe (falls vorrätig) oder den Artikel fotokopiert (unabhängig von der Seitenzahl). Bestellen Sie mit der Bestellkarte am Heftende.

Technische-Fragen-Service

Mit der Beantwortung technischer Fragen hilft Elektor den Lesern beim Nachbau von Elektor-Schaltungen. Damit diese Hilfe effektiv bleibt, bitten wir Sie, die folgenden Punkte zu beachten, bevor Sie zur Feder oder zum Telefon greifen:

- Für technische Fragen gibt es eine telefonische Fragestunde: Jeden Dienstag (außer im August) von 13.15 - 16.00 Uhr unter der Tel.-Nr. 02454/6047. **NEU!!!**
- **Fügen Sie Ihrer schriftlichen Anfrage einen adressierten und frankierten Umschlag für die Antwort** bei. Bei Anfragen aus dem Ausland bitten wir um einen internationalen Antwortschein.
- Eine **Kennzeichnung Ihres Schreibens mit dem Vermerk TF und dem Namen der Schaltung, zu der die Frage gestellt wird,** beschleunigt den Postweg in der Redaktion.
- Die beste Fehlerbeschreibung sind Meßwerte. Bitte schreiben Sie in jedem Fall möglichst sachlich und vor allem gut leserlich.
- Viele Bezugsquellen finden Sie im Anzeigenteil von Elektor. Fragen Sie uns möglichst erst, wenn Sie dort nichts gefunden haben.
- Wir können nicht alle Fragen technischer Art beantworten. Es sollte sich schon um Fragen zur Elektor-Schaltungen handeln. Ebenso ist es uns beim besten Willen nicht möglich, individuelle Schaltungswünsche und Änderungen auszuarbeiten.

Elektor-Platinen-Service

EPS	Preis (DM)	Platine
April 1982		
82017	39,90	Dynamische RAM-Karte
November 1983		
82159	37,45	Floppy-Disk-Interface
September 1983		
83082	39,60	VDU-Karte
November 1983		
83108-1	36,45	CPU-Karte
83108-2	22,70	
Dezember 1983		
83102	42,35	OmniBus
April 1984		
84017	21,00	EPROM-Löscher RADEFIX
Juni 1984		
84055	20,60	Schreibmaschinen-Schnittstelle
ec2		
65028	39,60	Kassettenschnittstellen-Karte
Oktober 1985		
85073	15,70	Serielle Schnittstelle
ec3		
85210	47,35	EC-68: CPU/DRAM-Karte
85211	47,35	DC-68 CRT/FDC-Karte
Januar 1986		
85080-2	60,20	Farbspeicherbänke Vektorgrafikkarte
ec4		
86207	47,–	EC-65K: CPU-Karte
86208	47,–	EC-65K: PSIO/RTC-Karte
86202	34,15	Mini-Modem
85114-1	67,15	Spooler-Hauptplatine
85114-2		Spooler-Steuerplatine
ec5		
86238	72,–	Z80-Karte
86240	71,25	EC-65K: SRAM-Karte
86053	18,–	Zusatzspeicher
86277	9,30	Auto-Dial

Tracer-Zusatzplatine für CPU-Karte: noch nicht lieferbar, bitte Anzeigenteil der Elektorhefte ab April beachten.

Elektor-Programmier-Service

Bezugsbedingungen wie EPS!

Sachgemäß verpackt eingesandte, einwandfreie PROMs und EPROMs werden unter Angabe der ESS-Bestellnummer mit folgenden Programmen programmiert:

ESS	Preis (DM)	Programm EPROM(s)
520	27,30	ASCII-Tastatur 1 × 2716
523	36,75	Zeichen-Generator für VDU-Karte, 1 × 2732
523D	36,75	wie 523, mit deutschem Zeichensatz, 1 × 2732
530	54,60	Schreibmaschinen-Schnittstelle 2 × 2716
533	36,75	SAMMON (Monitor-programm für SAMSON 65) 1 × 2732 (neue Version für BASICODE)
534	27,30	Anpassung Zeissler-Tastatur (Hinweis: die Tastatur wird normalerweise angepaßt geliefert!)
540	27,30	EC-68 Bootrom
541	27,30	EC-68 Assist 09
542	36,75	EC-68 Zeichengenerator
543	14,–	Kolorator: PROMS
545	20,–	Spooler-Steuerung
546	44,–	Monitor EC-65K
nnn	55,–	BIOS 80

EC 65 + EC 68

Wir sind jetzt bärenstark sortiert

74 S, 74 LS, 74 HC, 74 HCT, 74 F

uP + Speicher

ab Lager sofort

Lagerliste senden wir Ihnen gerne zu

6290 Weilburg 4 · Tel. 0 64 71/24 73 · Telex 482 289

edicta

WENZEL ELECTRONIC

Teile Nr.	zugr. nS	1 Stück	ab 10
RAMS			
MK48Z02	200nS	41.50	39.40
HM6116LP3	150nS	6.50	6.15
UPD4364L-15	150nS	8.45	8.05
UPD43256	120nS	149.00	141.55
UPD4164	150nS	4.90	4.65
UPD41256-12	120nS	8.95	8.50
EPROMS			
D2732A	250nS	10.95	10.40
M2764-2	200nS	7.80	7.40
D2764A-2	200nS	7.80	7.40
D26C64A-2 C-MOS	200nS	11.85	11.25
D27128	250nS	10.95	10.40
D27256-2	200nS	15.40	14.65
D27C256-2 C-MOS	200nS	21.00	19.95
uP Bausteine			
8748H		29.95	28.45
8250 UART		38.00	36.10
P8052AH-BASIC BASIC-52 USER MANUAL		79.00 74.00	75.05
Uhrenbausteine			
MM58167AN		44.85	42.60
RTC58321		21.95	20.85

Teile Nr.	zugr. nS	1 Stück	ab 10
uP Bausteine C-MOS			
Z80A CPU	C-MOS Z84C00	18.25	17.35
Z80A CTC	C-MOS Z84C20	14.45	13.75
Z80A PIO	C-MOS Z84C30	14.45	13.75
80C39	C-MOS CPU	9.50	9.00
80C31	C-MOS CPU	24.00	22.80
81C55	C-MOS RIOT	12.50	11.85
82C43	C-MOS I/O	8.90	8.45
UPD70108	V-20 CPU	34.00	32.30
UPD70116	V-30 CPU	37.00	35.15
UPD71051	C-MOS 8251	8.50	8.05
UPD71054	C-MOS 8254	9.15	8.70
UPD71055	C-MOS 8255	8.45	8.05
UPD71059	C-MOS 8259	10.50	9.95
AD Wandler			
TSC7109	12BIT+VZ	41.00	38.95
Spezialbausteine			
MAX-232	V24 INT	17.00	16.15
MC145406	V24 INT	7.95	7.55
MC1488	V24 INT	1.95	1.85
MC1489	V24 INT	1.95	1.85
ICL7660		11.45	10.90
ICL7665		8.15	7.75
ICL7673		8.10	7.70

Liefer und Zahlungsbedingungen:
Porto/Verpackung 5.80 DM - ab Auftragswert 200 DM frei. Lieferung erfolgt per Nachnahme Mindestauftragswert 30 DM. Alle Angebote freibleibend. Zwischenverkauf vorbehalten.

Wenzel Electronic 8751 Niedernberg
Hauptstr. 95 Tel. (06028) 7883

RATEV ELEKTRONIK-VERTRIEBS GMBH · 4030 RATINGEN 1 Postfach 1601
Gothaerstr. 15 Telefon: (02102) 42051 (Mailbox 02102/475400) Tx. 8585180

Rams

4164 150 nS	4,30
41256 120 nS	9,90
41256 150 nS	7,70
6116 LP-3	4,50
6264 LP-15	7,30
6116 Flat Pack	8,90
6264 Flat Pack	13,90
TMS 4464 150nS	19,90

Eproms

2716 450 nS	8,90
2732 250 nS	9,50
2764 250 nS	7,30
27C64 250 nS	10,90
27128 250 nS	7,90
27256 250 nS	13,95

ICs für Commodore

6510	33,90
6526	33,90
6569	98,00
6581	65,00

Mikroprozessoren

Z80A CPU	4,00
Z80A PIO	4,80
Z80A CTC	4,80
Z80A DMA	14,20
Z80A DART	13,50
Z80A SIO/0	13,50
Z80A STI	35,00
Z80B CPU	7,10
Z80B PIO	10,50
Z80B CTC	10,50
Z80B DART	19,50
Z80B SIO/0	21,30
6502P	9,85
6502A	11,80
6504P	12,50
6504A	13,90
6520P	7,80
6520A	9,30
6522P	9,85
6522A	10,80
6532P	12,30
6532A	16,50
6551P	12,80
6551A	13,40
65C02 1 MHz	18,50
65C02 4 MHz	39,90
6800	9,85
6802	8,35
6809	16,90
6821	4,50
6840	10,90
6845	13,75
6850	4,50
68000 8 MHz	44,90
68008 8 MHz	34,95

Controller

UPD765	16,50
FD 1770	57,45
FD 1771	33,50
FD 1791	24,30
FD 1793	24,30
FD 1797	24,30
FD 2793	27,30
FD 2797	27,30
WD 1691	27,50
WD 2143	21,90
FDC 9216B	14,95
FDC 9229B	43,90

Grafik-ICs

EF 9365	68,50
EF 9366	68,50
EF 9367	75,00
UPD 7220	39,00

Sonder-ICs

UPD 70108 D8	29,00
UPD 70116 D8	35,00
AM 7910	79,00
AM 7911	79,00
AM 25 LS 2538	8,70
HD3-4702	24,90
MM 58167	48,75
MSM 5832	18,50
MSM 58321	18,50
RTC 58321	17,80
UPD 1990	9,15
8087 5 MHz	375,00
8087 8 MHz	525,00

Quarzoszillatoren

alle gängigen Frequenzen	8,80

Quarze

0.032768 MHz	1,70
1.0 MHz	8,20
2.0 MHz	5,10
2.4576 MHz	4,90
alle anderen Standard-frequenzen	2,10

Kartenstecker (Floppy)

20 pol vergoldet	3,50
26 pol vergoldet	3,60
34 pol vergoldet	3,80
40 pol vergoldet	4,95
50 pol vergoldet	6,50

VG-Steckverbinder

64 Stift	2,15
64 Feder	2,90
96 Stift	4,20
96 Feder	5,25
64 Feder WW	4,50
96 Feder WW	6,80
64 Stift anschl.	28,90
64 Feder anschl.	12,20

D-Sub Steckverb. löt

Stift 9 pol	1,50
Stift 15 pol	1,80
Stift 25 pol	2,45
Stift 37 pol	4,30
Stift 50 pol	4,90
Feder 9 pol	1,90
Feder 15 pol	2,45
Feder 25 pol	2,90
Feder 37 pol	5,10
Feder 50 pol	7,90

D-Sub Steckverbinder Flachkabel

Stift 25 pol	7,95
Feder 25 pol	8,35

Gehäuse für D-Sub Steckverbinder

9 pol	1,60
15 pol	1,70
25 pol	1,80
37 pol	2,20
50 pol	3,50

Pfostenfederleisten 2-reihig

14 pol vergoldet	1,80
16 pol vergoldet	2,05
26 pol vergoldet	3,15
34 pol vergoldet	3,85
40 pol vergoldet	4,60
50 pol vergoldet	5,70

Pfostenstiftleisten mit Verpolungsschutz gerade

14 pol vergoldet	1,65
16 pol vergoldet	1,85
26 pol vergoldet	2,70
34 pol vergoldet	3,70
40 pol vergoldet	4,25
50 pol vergoldet	5,10

Pfostenstiftleisten 90°

14 pol vergoldet	1,95
16 pol vergoldet	2,40
26 pol vergoldet	3,35
34 pol vergoldet	4,20
40 pol vergoldet	4,85
50 pol vergoldet	6,25

Dilstecker für Flachkabel

14 pol	1,35
16 pol	1,55
24 pol	2,15
28 pol	2,35
40 pol	2,95

Centronics-Stecker

24 pol lötbar	9,20
36 pol lötbar	4,90
50 pol lötbar	12,90
36 pol Flachkabel	7,95

Centronics-Buchsen

24 pol lötbar	9,20
36 pol lötbar	5,85
50 pol lötbar	14,30
36 pol Flachkabel	8,95

Textool-Fassungen

16 pol	20,90
24 pol	21,00
28 pol	22,00
40 pol	31,00

Burn-In Fassungen

16 pol	7,85
24 pol	11,90
40 pol	13,85

Vielschicht Kondensatoren 0,1 µF Raster 2,5 oder 5 mm

10 Stück	à 0,25
100 Stück	à 0,20
500 Stück	à 0,18

Low-cost IC Sockel

8—40 pol p. Pin	0,02

Präzisions IC Sockel

8—40 pol p. Pin	0,05
64 pol	4,50

Flachbandleitung grau

pro Ader u. Meter	0,10

Weitere Steckverbinder auf Anfrage

IBM-Printerkabel 28,00

Liefer- und Zahlungsbedingungen: Die Lieferung erfolgt per Nachnahme + Porto und Versandspesen. Die Angebote sind freibleibend. Zwischenverkauf vorbehalten. Der Mindestbestellwert beträgt DM 30.00.

Aktuell ● Preiswert ● Schnell — Original- elektor -Bausätze mit Garantie

EC68K * * * EC68K * * * EC68K * * * EC68K

CPU-Karte	499,00
FDS-Karte (Floppy Disk Subsystem)	347,00
Floppy 3.5"	355,00
VDU-I/O-Karte	299,00
SRAM-Karte ohne Speicher	159,00
Speicher zu Tagespreisen	a.A.
BUS-Karte mit Leisten	198,00
Netzteil	342,00
Tastatur m. Sondertasten spez.	342,00
EPROM-Simulator	291,00
Midi-Interface	186,00
Zusatz-I/O-Karte (par./ser.)	236,00

* * * * Alle EC 68K-Bausätze inkl. Platine * * * *

EC 65 * * * EC 65 * * * EC 65 * * * EC 65

CPU-Karte kompl. inkl. Messerleiste	214,10
VDU-Karte kompl. inkl. Quarz/Spule	140,99
DRAM-Karte (64K) inkl. Messerleiste	169,10
FCU-Karte inkl. Messerleiste	77,99
EPROM-Karte mit Testfassung	84,80
Netzteil	61,10
passender Ringkerntrafo 225 VA	85,00
FUTABA-Profi-Tastatur parl./ser.	199,99
passendes Gehäuse FUTABA	99,88
RS232/Centronics-Schnittstelle	177,10
Kolorator * Farbgrafik-Karte	259,90
Kassetten-Interface	119,90
Echtzeituhr	80,96
Orig. Gehäuse EC65/Samson/BIG BOARD	298,99
Monitore, Drucker und Floppy's a. Anfrage	

EC 65 * * * EC 65 * * * EC 65 * * * EC 65

CPU-Karte inkl. Boot-Eprom	399,90
PSIO-RTC-Karte inkl. NiCd-Akku	189,90
SRAM-Karte (4 MHz./6264-100)	245,00
Z80-Karte (8 MHz./41256-120) inkl. Eprom	218,00
Mini-Modem inkl. Übertrager	221,90
Mini-Modem / Auto-Dial inkl. Relais	49,90
Spooler (Z80-CPU u. 64K Byte) inkl. Eprom	259,00
64-K-DRAM Zusatzspeicher (Kolorator usw.)	35,90

EC 68 * * * EC 68 * * * EC 68 * * * EC 68

CPU-DRAM-Karte inkl. EPROM's	255,50
CRT-FDC-Karte inkl. Eprom	199,90
SRAM-Karte (6264) 150 ns	148,90

Diesselhorst Elektronik
Biemker Straße 17
Tel. 0 57 34/32 08 4950 Minden

Bausätze, Spezialbauteile und Platinen auch zu älteren elektor-Projekten lieferbar!

Bauteilelisten gegen DM 1,80 in Bfm. Bausatz-Übersichtliste anfordern (Rückporto) Gehäuse-Sonderliste gegen DM 1,80 in Bfm.
Unsere Garantie-Bausätze enthalten nur Bauteile 1. Wahl (Keine Restposten) sowie grundsätzlich IC-Fassungen und Verschiedenes.
Nicht im Bausatz enthalten: Baubeschreibung, Platine, Schaltplan und Gehäuse. Diese können bei Bedarf mitbestellt werden.
Versandkosten: DM 7,50 Nachnahme Postscheck Hannover 121 007-305 DM 5,00 Vorkasse. Anfragenbeantwortung nur gg. Rückumschlag.

Inserenten-Verzeichnis

ACS 2
Aktuell Verlag 5, 66, 115

Böhm 2, 33

combit computer GmbH 18

Data-Media 119
Diesselhorst 114

Edicta 114
Eigenanzeigen 2, 5, 14, 66, 78, 120
Elektronikladen 114

Flesch 77

Habersetzer 48
Heck 18
Henz 115
Hüthig-Verlag 8

Kleinanzeigen 115

mp/c Datentechnik 115
Milde 115

Ratev 77, 114
Rieger 95

Schorstein 33

Wenzel 114

mp//c - Datentechnik

mp//c 703

ASCII-codierte Tastatur mit deutscher Tastenbelegung

Ultraflache Tastatur für EC-65(K). Ergonomische Formgebung und eine Höhe der 3. Tastenreihe von < 30 mm garantieren Bediensicherheit und leichte Handhabung.

- 7/8-bit ASCII-Code nach DIN 66003
- Tastencodes und Strings im EPROM änderbar
- Tastenverriegelung: n-key rollover oder 2-key rollover über DIL-Schalter einstellbar
- ALPHA-LOCK und SHIFT-LOCK mit LED-Anzeige
- RESET-Funktion über Cntrl-BREAK
- 19 Funktionstasten, Cursortasten, Tab- und Delete-Tasten können in drei Ebenen mit max. 6 Zeichen belegt werden.
- Betriebsarten: unshift, shift, control, shiftlock und alphalock
- Tasten (Marke MARQUARDT) mit Goldkontakten, Druckpunkt und 100% abriebfesten Kappen
- Autorepeat mit 0,6 bis 0,7 sec
- Serienmäßig belegte Funktions- und Cursortasten mit EC-65(K)-Befehlen
- Lieferumfang: Tastatur, Gehäuse u. Kabel fertig montiert und getestet inkl. Handbuch
- Preis **DM 488,– inkl. MwSt.**

Die Tastatur ist sowohl mit einer parallelen und einer seriellen Schnittstelle bestückt und somit auch an jedes andere Rechnersystem anschließbar. Für ausführliche Information bitte Prospekt anfordern.

mp//c Datentechnik

Tel. 0 22 37 6 10 01, Dipl.-Ing. J. Bornemann
5014 Kerpen 4, Postfach 4248 Heerstr. 392

Reinhard Milde
Postfach 70 13 44
8000 München 70
Telefon 0 89/7 69 46 31

Alle Preise im DM für 1 Stück zzgl. Versandspesen bei NN-Versand. Preise für größere Stückzahlen, OEM u. WV bitte anfragen! Angebot freibleibend!

Diskettenlaufwerke

3,5" und 5,25" verschiedener Hersteller
z.B. 5,25" 1MB SA 460 **299,–**
5,25" 0,5MB Qume 142 **289,–**
weitere von EPSON, PANASONIC, TEAC, NEC a. A.

Winchester-Laufwerke 10 . . 150 MB und Controller (PC u.a.) bitte anfragen!
z.B. 5,25" 12,9MB SA712 **1095,–**

Integrierte Schaltungen

	ab 1	ab 10	ab 25
4164-15	3,90	3,80	3,60
41256-15	8,90	8,70	8,50
41256-12	10,30	10,10	9,90
511000-12	285,–	265,–	255,–
61161p-3	4,90	4,80	4,65
6264-1p15	7,95	7,85	7,70
62256-12	215,–	198,–	190,–
2764-25	7,70	7,50	7,30
27128-25	8,90	8,80	8,70
27256-20	13,90	13,70	13,50
27C64-25	9,95	9,80	9,60
UMC 9216	16,80	16,30	15,90
UMC 9229B	57,90	55,50	52,95
µPD 765	17,90	17,50	16,90
NEC V20 - 8MHz	33,90	32,60	31,30
NEC V30 - 8MHz	38,50	37,20	35,50

i808 ×, i825 ×, Z80A bitte anfragen!

Disketten

	ab 10	50	100
FUJI			
3,5" MF1D	6,50	6,40	6,20
3,5" MF2DD	10,50	10,30	10,10
5,25" MD2HD	11,95	11,35	10,80
PANASONIC			
5,25" MD2D	3,20	2,95	2,85
WHITE LABEL			
3,5" MF1D	5,50	5,40	5,20
3,5" MF2DD	8,90	8,70	8,50
5,25" MD2D	2,65	2,50	2,35
5,25" MD2D Wende	2,95	2,80	2,60
5,25" MD2D farbig	2,95	2,80	2,60

Sicherheit auf allen Wegen

Dazu: Roland Bauer, Industriemeister, Alzenau

Meine Meinung ist: Ich lasse die Finger vom Alkohol, wenn ich noch fahren muß. Denn bei einem Glas bleibt es ja meist nicht. Darum halte ich mich an den Grundsatz: NÜCHTERN FAHREN – SICHER ANKOMMEN.

BG – Hallo Partner danke schön

Ihre Berufsgenossenschaft und der Deutsche Verkehrssicherheitsrat

Gemeinsam helfen. Gemeinsam heilen.

Brot für die Welt

Postgiroamt Köln 500500-500

TI REVUE

Das Magazin für TI 99-4A

30 Seiten Listings für Ihren TI 99/4A

Assembler leicht gemacht

Neue Software im Test

Drucker - richtig behandelt!

4 Seiten Anzeigen rund um den TI 99/4A

Klein-anzeigen

Suche Kontakte zu Anwendern des Eurocom 2 mit Flex.
Terrence Barr
Dompfaffenweg 12h,
8192 Geretsried-1
Tel. 08171/51322

Die letzte Seite

Im nächsten Heft

haben wir uns eine Menge vorgenommen. Zum Beipiel die UDC-Karte - so sie bis dahin fertig ist, was wir aber hoffen. Für das EC-DOS gilt das Gleiche. Auf alle Fälle fertig sind vier Karten für den EC-68K, und die Artikel dazu dürften ebenso fertig werden wie ein Artikel zur Software des EC-68K, in dem es unter anderem um die wichtigen Systemadressen geht.

Jetzt schon fertig sind zwei Artikel zum Thema FLEX: Zum einen die Beschreibung der Programmiersprache PL-9, zum anderen ein Artikel zu "Copycat". Copycat ist ein spezielles Diskettenkopier-Programm, mit dem man unter anderem IBM-Formate lesen kann. Bei der neuen Speichererweiterung für den Big-Board II hoffen wir, daß die neue 1 MByte-Platine sich nicht so grundlegend von der 256 KByte-Platine unterscheidet: Dann können wir nämlich auch den schon fertigen Artikel zur 256 KByte-Platine entsprechend umschreiben, so daß auch dieses Thema in ec 6 "abgehakt" werden könnte.

Auf Seite des Autors auch schon fertig ist ein Farb-Grafik-Terminal. Es basiert auf unserer alten 6502-CPU-Karte und der Kolorator-Karte, ist also im Wesentlichen eine eigenständige Fortentwicklung bestehender Hard- und Software.

Und dann gilt es natürlich noch einige Artikel abzuschließen beziehungsweise fortzuführen: EDMO Teil 2, 65816 Teil 4, V30 Teil 2, der "Rest" des Plotters, die Platine zum Maus-Interface und - last but not least - das 6502-Modemprogramm. Und ein bißchen Platz für die eine oder andere Überraschung brauchen wir vielleicht auch noch...

Einen neuen Super-Computer mit absolut flimmerfreiem Bildschirm hat die Firma "Stonehenge Componists" auf der Zehbit im März dieses Jahres vorgestellt.

Die "Augenärztliche Vereinigung", die wir um Stellungnahme baten, rät jedoch von der Anschaffung des Rechners ab: Man befürchtet Augenverletzungen durch umherfliegende Steinsplitter.

Kommentar des Rechner-Herstellers: "Die haben ja nur Angst, ihre flimmergeschädigten Patienten zu verlieren".

IMPRESSUM

elektor computing (ec)

5. Sonderheft
3-Monatsangebot

HERAUSGEBER: Elektor Verlag GmbH,
Süsterfeldstraße 25, 5100 Aachen,
Tel.: 0241/81077, Telex: 8329371 elek d
Bank: Kreissparkasse Gangelt,
Konto 03001-294, Bankleitzahl 31251220, und
Postgirokonto Köln 229744-507
Geschäftszeiten: Montag — Freitag
8.30—12.30; 13.00—16.00 Uhr

Chefredaktion Elektor: E. Krempelsauer

Redaktionsleitung ec: P. v.d. Linden

Mitarbeiter und Autoren dieses Sonderheftes:
F. Aubert, F. Blaha, G. Diethert, B. Fasel, P. Hogenboom, P. Lavigne, D. Meyer, N. Pistorius, G. Raedersdorf, D. Reithinger, F. Schmidt, D. Solas, J. Steeman, P. Treytl

Grafische Gestaltung:
Grafisches Büro Sinke (GBS)

Redaktionssekretariat:
W. Wijnen, M. Lacroix

Marketing: D.K. Grimm

Verlagsleitung und Vertrieb: F.H. Petruch
Anzeigenleitung: B. Onnertz

Auslieferung in der Schweiz: Verlag und Versandbuchhandlung Thali AG, CH-6285 Hitzkirch, Tel.: 041/852828.
Vertretung für das Bundesgebiet und Berlin (West): Pressebüro Ferdinand C. Mentzen, Ubierstr. 36, 5300 Bonn-Bad Godesberg, Tel.: 0228/354781. Telex 885753.

Der Herausgeber ist nicht verpflichtet, unverlangt eingesandte Manuskripte oder Geräte zurückzusenden. Auch wird für diese Gegenstände keine Haftung übernommen. Nimmt der Herausgeber einen Beitrag zur Veröffentlichung an, so erwirbt er gleichzeitig das Nachdruckrecht für Holland, England, Frankreich, Italien, Spanien, Griechenland, Portugal, Indien.

Die in dieser Zeitschrift veröffentlichten Beiträge, insbesondere alle Aufsätze und Artikel sowie alle Entwürfe, Pläne, Zeichnungen einschließlich der Platinen sind urheberrechtlich geschützt. Ihre auch teilweise Vervielfältigung und Verbreitung ist grundsätzlich nur mit vorheriger schriftlicher Zustimmung des Herausgebers gestattet.
Die veröffentlichten Schaltungen können unter Patent- oder Gebrauchsmusterschutz stehen. Dann sind Herstellung, Feilhalten, Inverkehrbringen und gewerblicher Gebrauch nur mit Zustimmung des Schutzrechtinhabers zulässig. Nur der private Gebrauch ist frei.
Bei den benutzten Warenbezeichnungen kann es sich um geschützte Warenzeichen handeln, die nur mit Zustimmung ihrer Inhaber warenzeichengemäß benutzt werden dürfen.
Die geltenden gesetzlichen und postalischen Bestimmungen hinsichtlich Bau, Erwerb und Betrieb von Sende- und Empfangseinrichtungen sind unbedingt zu beachten.
Eine Haftung des Herausgebers für die Richtigkeit und Brauchbarkeit der veröffentlichten Schaltungen und sonstigen Anordnungen sowie für die Richtigkeit des technischen Inhalts der veröffentlichten Aufsätze und sonstigen Beiträge ist ausgeschlossen.

© = 1986 Elektor Verlag GmbH —
Printed in the Netherlands

elektor computing

Bestell-Karte

Absender
(Bitte deutlich schreiben)

Vorname/Name

Beruf

Straße/Nr.

PLZ Ort

Telefon Vorwahl/Rufnummer

Bitte mit 60 Pf. Brfm. freimachen

Postkarte

Elektor Verlag GmbH

Süsterfeldstr. 25

5100 Aachen

---

elektor computing

Service-Karte

Absender
(Bitte deutlich schreiben)

Vorname/Name

Beruf

Straße/Nr.

PLZ Ort

Telefon Vorwahl/Rufnummer

Bitte schicken Sie diese Karte direkt an die Firma, bei der Sie bestellen. ▶

Bitte mit 60 Pf. Brfm. freimachen

Postkarte

Firma

Straße/Postfach

PLZ Ort

---

elektor computing

Service-Karte

Absender
(Bitte deutlich schreiben)

Vorname/Name

Beruf

Straße/Nr.

PLZ Ort

Telefon Vorwahl/Rufnummer

Bitte schicken Sie diese Karte direkt an die Firma, bei der Sie bestellen. ▶

Bitte mit 60 Pf. Brfm. freimachen

Postkarte

Firma

Straße/Postfach

PLZ Ort

---

elektor computing

Service-Karte

Absender
(Bitte deutlich schreiben)

Vorname/Name

Beruf

Straße/Nr.

PLZ Ort

Telefon Vorwahl/Rufnummer

Bitte schicken Sie diese Karte direkt an die Firma, bei der Sie bestellen. ▶

Bitte mit 60 Pf. Brfm. freimachen

Postkarte

Firma

Straße/Postfach

PLZ Ort

elektor computing Service-Karte

Ich beziehe mich auf die in elektor computing Nr. 5, Seite **erschienene**

- ☐ Anzeige
- ☐ bitte um weitere **Informationen** über Ihr Produkt _____
- ☐ Bestelle folgendes unter Anerkennung Ihrer Liefer- und Zahlungsbedingungen:

Menge	Produkt/Bestellnummer	à DM	gesamt DM

Absender nicht vergessen!

Datum, Unterschrift (fur Jugendliche unter 18 Jahren der Erziehungsberechtigte)

Ich bestelle hiermit aus dem Elektor-Programm:

Bücher

Elektronik Baubuch	DM 29,50
6502 Maschinencode fur schnelle Grafik	DM 23,
Modellenbahnsteuerung	DM 25,
Computer-Informationsverarbeitung	DM 48,
302 Schaltungen	DM 31,
Forth Programmiersprache der vierten Generation	DM 28,50
269 IC's	DM 33,50
So machen Sie mehr aus Ihrer Stereo anlage	DM 24,
Elektronik, gar nicht schwer	DM 17,75
Bascode Jenschi Kassettel	DM 58,
Wege in die Elektronik	DM 34,
Umwelt elektronisch messen	DM 24,
Die 68000er Serie, Buch 1	DM 42,50
Die 6800er Serie, Buch 2	DM 44,
Heimcomputer Bastelkiste	DM 29,80

Elektor-Zeitschriften

elektor plus 1 Hft Boxen im Selbstbau	DM 15,
elektor plus 2 Labor Meßgerate	DM 15,
elektor plus 3 Audio Elektronik	DM 15,
MSX -	DM 15,
elektor computing 1, 2, 3, 4	je DM 18,
elektor Ausgabe Nr	DM 6,
elex Ausgabe Nr	DM 4,50

Elektor-Platinen

Stückzahl	Bezeichnung	DM Preis

Zuzügl. für Porto und Verpackung DM 2,50 pro Sendung

☐ Senden Sie mir bitte Ihren kostenlosen Bücherprospekt

Zahlungsbedingungen: per Vorkasse.

Den Betrag von DM . . . . . habe ich auf

☐ Ihr Postscheckkonto Köln 229744-507

☐ Ihre Bank, Commerzbank Aachen, Konto 1201102, BLZ 39040013 überwiesen.

☐ Über den Betrag von DM . . . . . liegt ein Verrechnungsscheck bei.

(Zutreffendes bitte ankreuzen)

Diese Vereinbarung kann schriftlich innerhalb von 8 Tagen widerrufen werden.

Datum, Unterschrift

elektor computing Service-Karte

Ich beziehe mich auf die in elektor computing Nr. 5, Seite **erschienene**

- ☐ Anzeige
- ☐ bitte um weitere **Informationen** über Ihr Produkt _____
- ☐ Bestelle folgendes unter Anerkennung Ihrer Liefer- und Zahlungsbedingungen:

Menge	Produkt/Bestellnummer	à DM	gesamt DM

Absender nicht vergessen!

Datum, Unterschrift (fur Jugendliche unter 18 Jahren der Erziehungsberechtigte)

elektor computing Service-Karte

Ich beziehe mich auf die in elektor computing Nr. 5, Seite **erschienene**

- ☐ Anzeige
- ☐ bitte um weitere **Informationen** über Ihr Produkt _____
- ☐ Bestelle folgendes unter Anerkennung Ihrer Liefer- und Zahlungsbedingungen:

Menge	Produkt/Bestellnummer	à DM	gesamt DM

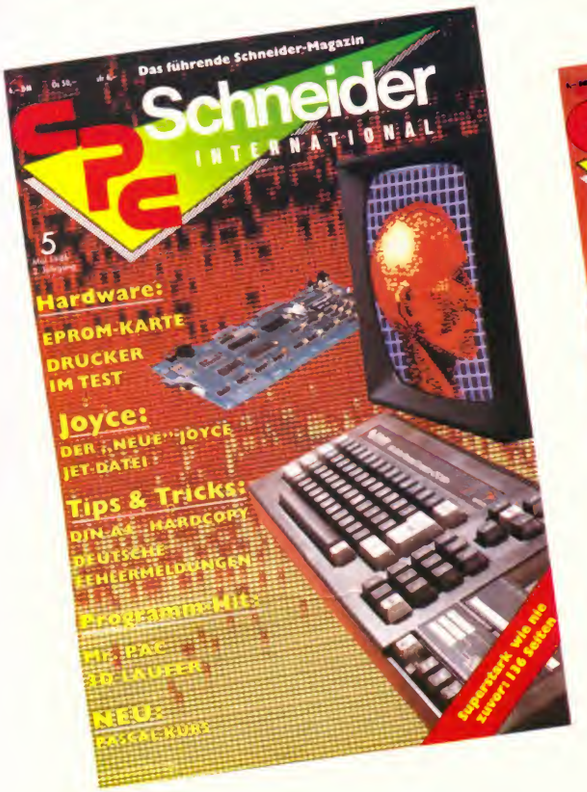
Absender nicht vergessen!

Datum, Unterschrift (fur Jugendliche unter 18 Jahren der Erziehungsberechtigte)

Das führende Schneider-Magazin

Schneider CPC International

Jeden Monat NEU und aktuell!



Jetzt mit neuer Rubrik: HARDWARE-BASTELEIEN mit den Schneider CPC's

DMV-Verlag, Fuldaer Str. 6, 3440 Eschwege, Tel.: (0 56 51) 87 02

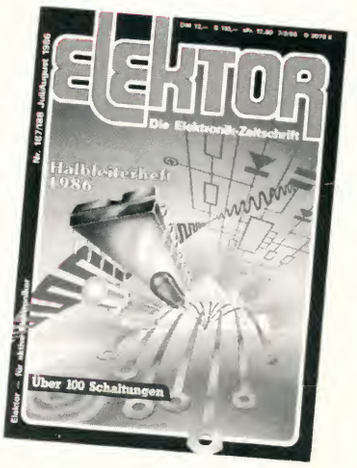
ELEKTOR

Zeitschriften
für alle Elektronik-Bereiche

ELEKTOR

Fachzeitschrift für Elektronik

* für Beruf, Studium und Hobby
* Bauprojekte aus allen Elektronikbereichen
* aktuelle Info-Artikel
* erscheint monatlich



elex

Zeitschrift für Freizeitelektronik

* für den leichten Einstieg in die Elektronik
* praxisnahes Wissen, keine graue Theorie
* aktuelle, nachbausichere Schaltungen
* marktorientierte Information, Tips und Tricks
* erscheint monatlich

elektor plus 1

Hifi-Boxen im Selbstbau

* Hifi-Boxen zum Selbstbau
* aktuelle Bauprojekte von "klein bis groß"
* vermittelt Grundlagenwissen
* Tips für den Selbstbau und für den Kauf fertiger Boxen

elektor plus 2

Labor-Meßgeräte

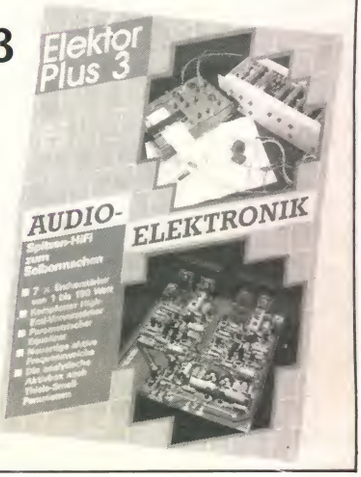
* Bauprojekte zu allen Bereichen des Meßgeräte-Sektors
* Marktübersichten: Digital-Multimeter und Oszilloskope
* Infos über Eigenschaften, Auswahlkriterien, Funktionen, Anwendungsbereiche

elektor plus 3

Audio-Elektronik

Spitzen-HiFi zum Selbermachen

* aktuelle Bauprojekte
* große und kleine Zusatzschaltungen für die HiFi-Anlage
* marktorientierte Info-Artikel



weitere Zeitschriften in Vorbereitung:
elektor plus 4 Hifi-Boxen im Selbstbau
elektor plus 5 Interface-Sonderheft

Bestellen Sie mit der Karte am Heftende.

Elektor Verlag, Aachen