

Computer Schau

Heft Nr. 225
Preis 28,- DM
sfr 28,-, öS 210,-

Kompakt-
wissen

Klar sehen beim Schneider CPC 464/664/6128

Teil 2

Grundlagen, Anwendungen, Programmlistings



Auf Profis programmiert:



Mit mc lernen Sie Computer von Grund auf verstehen. Ausführliche Funktionsbeschreibungen von Rechner-Hardware und gut kommentierte Programm-Listings bieten Ihnen den richtigen Einstieg ins ernsthafte Computern.



Durch Programme in mc werden Sie manches Problem überhaupt nicht mehr als Problem betrachten.



Nach mc-Bauanleitungen löten Sie vom einfachen Interface bis zum kompletten System, was an Hardware nur schwer zu kaufen ist.

mc

Die Mikrocomputer-Zeitschrift

6.50 DM · 55 öS · 7 sfr. · September 1985

9

68008-Platine für Apple-II

Geknackter Macintosh

Kommunikation mit dem mc-68000-Computer

UCSD-Pascal unter MS-DOS

Erweitertes C-64-Grafikpaket



In mc-Fachaufsätzen geht's um neue Entwicklungen, um professionelle Hardware und Peripherie.



Natürlich testet mc Geräte und Programme. Die Ergebnisse werden aus der Sicht des professionellen Anwenders interpretiert.

Aktuelles aus der Branche zu Unternehmen, Produkten, Kongressen, Tagungen und Messen finden Sie jeden Monat in mc.

mc bringt Profis weiter.

Für DM 6,50 bekommen Sie mc an jeder größeren Zeitschriften-Verkaufsstelle.

mc können Sie aber auch auf andere Art kennenlernen.

Kostenlos und unverbindlich.

Die Abrufkarte dafür finden Sie an der Umschlagklappe.

mc

Die Mikrocomputer-Zeitschrift

Lieber Leser,



mit diesem Heft des ComputerSchau-Kompaktwissens halten Sie die zweite Ausgabe in Händen, die sich ausschließlich mit den Schneider-Computern der CPC-Serie beschäftigt.

Sie haben damit eine weitere Informationsquelle, die Ihnen beim Umgang mit Ihrem Schneider-Computer behilflich ist.

In allen Programmlistings ist eine Fülle an Tips und Kniffen enthalten, die Sie auch bei der Erstellung Ihrer eigenen Programme verwerten können. Die im Heft enthaltenen Programme wurden – soweit es uns möglich war – so geschrieben, daß sie auf allen CPCs laufen.

Diesmal haben wir auch für die Besitzer von Diskettenstationen höchst interessante Beiträge.

Nicht nur die Grundlagenartikel gehen stark auf

die Disketten ein, in vielen weiteren Artikeln gibt es Ideen, Tips und Kniffe für verschiedenste Anwendungen. Für Anwender von CP/M 2.2 gibt es ein Programmlisting für eine DIN-Tastatur. Auch ein Programm zur kompletten Rettung von versehentlich gelöschten Files oder zur Abfrage des Disketten-Schreibschutzes ist enthalten.

Und wie im letzten Heft schon angedeutet, greifen wir auch auf die zweite Speicherbank beim CPC 6128 zu. Doch auch Eigentümer des CPC 464 mit Kassettenrecorder werden mit vielen interessanten Beiträgen bedient.

Ein weiterer „Knüller“ ist die im Heft enthaltene Gegenüberstellung der CPC-Adressen. Dieser Anhang soll es Ihnen ermöglichen, Programme, die „Peeks“ und „Pokes“ enthalten – oder sogar Maschinenprogramme –, die nicht für Ihren CPC-Typ geschrieben wurden, an Ihr System anzupassen.

Wer die im Heft enthaltenen Programme nicht abtippen will, hat die Möglichkeit, all diese (und auch die des ersten Heftes) vom Franzis-Software-Service auf Diskette zu beziehen. Diesen Service haben wir eingerichtet, damit Leser ohne die Mühe des Abtippens in den Genuß der Programme kommen, nachdem wir immer wieder darum gebeten wurden.

Viel Freude und Erfolg mit Ihrem CPC wünscht Ihnen

Th. Lorenz

Computer Schau KOMPAKTWISSEN

Verleger: Franzis-Verlag GmbH, München

Anschrift für Verlag, Redaktion, Anzeigenabteilung und alle Verantwortlichen (Anschrift für Kunden-Service siehe unter Vertrieb): Karlstraße 37–41, 8000 München 2, Postanschrift: Postfach 37 01 20, 8000 München 37, Telefon 0 89/51 17-1, Telex 5 22 301, Telefax 0 89/51 17-3 79, Btx-Leitseite *30503#, Tedas 0 89/59 84 23 und 59 64 22; Postgirokonto 57 58-807

Auslandsgesellschaft: Franzis Publishing Co., 504 Nino Avenue, Los Gatos, CA 95030, USA. Telefon (4 08) 3 58-21 51, Telex (00230) 171611

Geschäftsführung: Peter G. E. Mayer, Michael-Alexander Mayer

Verlagsleitung: Peter Habersetzer, Eugen Wintersberger

Redaktionsleitung: Günther Klasche

Anzeigenleitung: Johann Bylek

Vertriebsleitung: Benno Gaab

Redaktion:

Burkhard P. Bierschenck M. A. (verantw.), Dipl.-Ing. Gerd Heyer

Freier Mitarbeiter: Lothar Miedel

Assistentin: Ingrid Daschner, Telefon 0 89/51 17-2 29

Franzis-Labor: Dipl.-Ing. (FH) Hans Neumayr

Art Direction: Dipl.-Designer (FH) Volker Hilbel

Layout, Grafik, Herstellung: Richard Maier

Sonderdrucke: Jakob Wintersberger

Lizenzen: Siegfried Pruskil

Franzis-Software-Service: Dipl.-Inform. Jürgen Plate, Telefon 0 89/51 17-3 31

Anzeigen:

Verkaufsleitung: Norbert Mann, Telefon 0 89/51 17-3 68

Disposition: Edith Hufnagel, Telefon 0 89/51 17-2 36

Auslandsvertretungen für Anzeigen:

USA: International Media Marketing, 7330 Adams St. Paramount, CA 90723, phone (2 13) 4 08-09 99, telex No. (9 10) 321-3026. **Frankreich:** Agence Gustav Elm, 41, avenue Moutaigne, 75008 Paris, phone 01-7 23 32 67. **United Kingdom:** Martin Geerke, Friary Hall (Flat 3), Friary Road, South Ascot, Berks SL5 9HD, UK, phone (09 90) 2 86 49, telex: 8 58 328. **Schweiz:** Exportwerbung AG Zürich, Kirchgasse 50, CH-8024 Zürich, Tel. 01-47 46 90, Telex 812 765. **Japan:** International Media Rep. Ltd., 2-29, Toranomon 1-chome, Minato-ku, Tokyo 105, phone 5 02-06 56, telex 22 633. **Italien:** Rancati advertising, Milano San Felice Torre 5, I-20090 Segrate, phone 0 92-7 53 14 45, telex 3 11 250 PP MII

Vertrieb:

Anschrift: Franzis-Verlag, Kunden- u. Abonnement-Service, Postfach 37 02 80, 8000 München 37, Telefon: Abonnement-Service 0 89/51 17-2 40/-2 79, Handel 0 89/51 17-2 04/-2 83

ComputerSchau Kompaktwissen erscheint 2monatlich (6× im Jahr). Bestellungen nehmen jede Buchhandlung im In- und Ausland und der Verlag entgegen

Bezugspreis: Einzelheft 28,- DM (28,- sfr, 210 öS), Jahresabonnement 144,- DM. In den Preisen ist die gesetzliche Mehrwertsteuer in Höhe von 7% enthalten, in den Abonnementspreisen auch die Versandkosten

Auslandsvertretungen für Heftbezug:

Belgien: Office International des Périodiques (O.I.P.), Avenue Marnix 30, B-1050 Brüssel

Dänemark: Harck + Gjellerups Booksellers Ltd., Fiolstraede 31–33, DK-1171 Kopenhagen K

Frankreich: Librairie Parisienne de la Radio, 43, rue de Dunkerque, F-75010 Paris

Luxemburg: Messageries Paul Kraus, 5, rue de Hollerich, Luxembourg

Niederlande: De Muiderkring N. V., Nijverheidsweg 17–19–21, Bussum

Österreich: Erb-Verlag Ges.m.b.H. & Co. KG, Buch- und Zeitschriftenvertrieb, Amerlingstr. 1, A-1061 Wien

Schweiz: Verlag Thali AG, CH-6285 Hitzkirch/Luzern

Verantwortlich für den Textteil: Burkhard P. Bierschenck;

für den Anzeigenteil: Johann Bylek

Gesamtherstellung: Franzis-Druck GmbH, Karlstraße 35, 8000 München 2, Tel. 0 89/51 17-1

Imprimé en Allemagne. Printed in Germany

ISSN 0177-2139

© 1985 Franzis-Verlag, München.

Urheberrechte:

Die in ComputerSchau Kompaktwissen veröffentlichten Beiträge sind urheberrechtlich geschützt. Alle Rechte, insbesondere das der Übersetzung in fremde Sprachen, vorbehalten. Kein Teil dieser Zeitschrift darf ohne schriftliche Genehmigung des Verlages in irgendeiner Form – durch Fotokopie, Mikrofilm oder andere Verfahren – reproduziert oder in eine von Maschinen, insbesondere Datenverarbeitungsanlagen, verwendbare Sprache übertragen werden. Auch die Rechte der Wiedergabe durch Vortrag, Funk- oder Fernsehendung, im Magnettonverfahren oder ähnlichem Wege bleiben vorbehalten. Fotokopien für den persönlichen oder sonstigen eigenen Gebrauch dürfen nur von einzelnen Beiträgen oder Teilen daraus als Einzelkopien hergestellt werden.

Für eingesandte Manuskripte wird keine Haftung übernommen.

Die Schneider-Computer:

CPC 464
CPC 664
CPC 6128

**Computer
Schau**

Heft Nr. 225
 Preis 28,- DM
 sfr 28,-, öS 210,-

Klar sehen beim Schneider
CPC 464/664/6128 Teil 2
 Grundlagen, Anwendungen, Programmiersing



„Klar sehen“ heißt für die Computer-Schau-Redaktion, Ihnen mit diesem Kompaktwissen-Heft einen Wissensvorsprung zu garantieren. Deshalb haben wir dafür gesorgt, daß Schneider-Freunde mit unseren Tips und Programmen zu echten Micro-Profis werden, die sogar komplizierte Anwendungssoftware selbst erstellen und analysieren können.

Burkhard P. Bierschenck

GRUNDLAGEN

Arbeiten mit Diskette (1)	6
Arbeiten mit Diskette (2)	9
Arbeiten mit Diskette (3)	12
Arbeiten mit Diskette (4)	18
Arbeiten mit Diskette (5)	22

ANWENDUNGEN

TEXT-PRO – Ein Textverarbeitungsprogramm	28
Arbeiten mit dem 2. Speicher beim CPC 6128	38
Der CPC 464 programmiert sich selbst!	45
Uhrzeiten rund um die Welt	48
DATA-PRO – Ein komfortables Dateiverwaltungsprogramm	52
Eine Text-Hardcopy für die Schneider-CPCs	64
Speichern Sie Zeichensätze ab!	68
Dateien verschlüsselt	72
Rund um den Kalender – auf dem Schneider CPC 464	74

UTILITIES

Schreibschutz abgefragt	79
Disc-Backup auf Kassette	82
„Memory Check“ – Prüfen Sie Ihren Computer!	87
Anzeige der Funktionstasten	92
Der Kompaktor	96
DIN-Tastatur für Schneider CPC 464/664/6128 unter CP/M 2.2	102
Die zweite Möglichkeit für DIN.COM	106

ALLGEMEINES

MID\$, die zweite Möglichkeit mit CPC 464	109
DEC\$ auf dem CPC 464	112

SPIEL

Quint – Ein Spiel mit dem Joystick	114
------------------------------------	-----

ANHANG

Aufstellung: Vergleichstabelle der CPC-Systemadressen	119
---	-----

Arbeiten mit Diskette (1)

Einleitung

In dieser Rubrik konnten Sie im letzten Kompaktwissen vor allem Einzelheiten und Eigenarten über Ihren CPC kennenlernen. Diesmal geht es um ein externes Gerät, die Diskettenstation.

Der CPC 664 und auch der CPC 6128 sind mit diesem schnellen Massenspeicher von Hause aus versehen. Der CPC 464 kann damit nachgerüstet werden. Will man nun nicht nur mit käuflicher Software arbeiten, dann ist es gut, mehr über den Umgang mit dem Diskettenlaufwerk zu wissen.

Bevor ich nun auf das eigentliche Thema eingehe, vorab gleich ein „echter Hammer“: Demnächst werden von einem Fremdanbieter auch Harddisks für den CPC 464 angeboten. Weiterhin ist mit einer Zusatzplatine zu rechnen, die durch einen Coprozessor den CPC 464 auch MS-DOS-fähig und IBM-kompatibel machen soll. Dann sind für die CPCs weitere professionelle Einsatzmöglichkeiten offen. Bis zum Zeitpunkt der Niederschrift dieses Artikels konnte ich nur ein Mustergerät in Betrieb sehen. Es liegen außerdem noch keine detaillierten Angaben hierüber vor, deshalb kann ich auch noch keine genaueren Angaben machen. Geplant sind jedoch zwei Ausführungen mit zehn und zwanzig Megabyte.

Ein einziges Programm für viele Anwendungen

Doch zurück zum Umgang mit der Diskettenstation, ich will Ihnen ja einiges hierüber mitteilen. Für die Anwendung der in diesen Artikeln veröffentlichten Programme ist dieses Wissen zwar nicht unbedingt erforderlich aber sehr, sehr nützlich.

Lesen Sie deshalb, auch wenn Sie an Maschinensprache nicht interessiert sind, weiter. Ich bemühe mich, die Dinge so einfach wie nur möglich darzustellen. Auch weitere eigene Disketten-Programme können Sie dann durch den Einsatz von nur einem (!) Maschinenprogramm entwickeln. Und dies alles in Basic! Keine Angst, Sie sollen nun nichts über den komplizierten technischen Ablauf zu lesen bekommen, sondern nur die Dinge, die für alle Anwendungen, die nun durchgesprochen werden sollen, erforderlich sind.

Das Basiswissen über die Diskettenstation und über die Diskette selbst aber, sollten Sie den Handbüchern entnehmen.

Zur Auffrischung bzw. Erinnerung sei nur noch einmal darauf hingewiesen, daß eine Diskette bei der Formatierung ein magnetisches Muster „aufgedrückt“ bekommt und dadurch die Diskette in Spuren (tracks) und Sektoren (sectors) eingeteilt wird.

Diese Formatierung kann für drei Formate durchgeführt werden.

- IBM-Format
- Data-only-Format
- System Format (CP/M-Format und Vendor)

(Das Vendorformat ist nur eine Sonderform des CP/M-Formates.)

Aber auch andere Formate sind mit dem Wissen über die Arbeitsweise, innerhalb der Grenzen des Controllerbausteines möglich. Hierzu müssen dann eigene – anstelle der „eingebauten“ – Routinen geschrieben und die Diskettenparameter direkt in den „Extended Disk Parameter Block“ gepatcht werden. Softwareanbieter benutzen derartige Techniken als Schutz gegen Raubkopien. Das IBM-Format will ich derzeit größtenteils noch unberücksichtigt lassen, da wahrscheinlich nur weni-

ge Anwender unter dieser Formatierung arbeiten. Prinzipiell kann aber alles folgende auch analog für dieses Format „umgedacht“ werden.

Was Sie nun lesen, trifft also für beide behandelten Formate, nämlich Data-only und CP/M zu. Bei der Formatierung wird die Diskette in 40 Spuren (0 bis 39) mit je 9 Sektoren (1 bis 9) eingeteilt. Die Sektoren erhalten als Kennzeichen für das Format ein Identifikationszeichen, welches sowohl die Sektornummer, als auch den Code für das Format beinhaltet.

Das Formatkennzeichen für Data-only ist dabei der Wert &C0 und für CP/M &40. Diese Werte müssen zur jeweiligen Sektornummer addiert werden. Unter CP/M gibt es in jeder Spur also die Sektornummern &41 bis &49 und unter Data-only die Nummern &C1 bis &C9. (IBM-Format 01 bis 08!)

Ein weiterer Unterschied zwischen diesen beiden Formaten ist, daß unter CP/M zwei komplette Spuren für das System reserviert sind und dem Anwender normalerweise nicht zur Verfügung stehen. Diese beiden Spuren beinhalten den „Boot-Sector“, den „Configuration-Sector“, „CCP“ und „BDOS“.

Immer ans Copyright denken!

Beim Format Vendor werden diese beiden ersten Spuren nicht mit wirklichen Daten beschrieben, sondern freigehalten. Deshalb dient diese Formatierung auch zur Weitergabe von Disketten mit CP/M-Programmen. Bitte denken Sie immer daran, daß Sie aus Copyright-Gründen keine „echten“ CP/M-Disketten weitergeben dürfen!

Wenden wir uns nun weiteren Einzelheiten zu. Jeder Sektor auf der Diskette „faßt“ insgesamt 512 Bytes

an Daten. Eine einfache Multiplikation zeigt nun die Diskettenkapazität unter den beiden Formaten.

Formel: $VS \cdot AS \cdot BS$

Hierbei bedeuten:

VS = Anzahl der verfügbaren Spuren

AS = Anzahl der Sektoren pro Spur

BS = Bytes pro Sektor.

Mit eingesetzten Werten ergeben sich folgende Kapazitätswerte:

SYSTEM-Format:

$(40-2) \cdot 9 \cdot 512 = 175\,104 \text{ Bytes} = 171 \text{ KByte.}$

Data-only-Format:

$40 \cdot 9 \cdot 512 = 184\,320 \text{ Bytes} = 180 \text{ KByte.}$

Wer sich bei einer frisch formatierten (oder „gelöschten“) Diskette schon einmal das Inhaltsverzeichnis angesehen hat, wird nun bereits Differenzen zwischen meinen Angaben und der Mitteilung der „free-Meldung“ feststellen. Der Grund dieser Differenz ist aber schnell geklärt. Das Inhaltsverzeichnis (Directory) benötigt nämlich ebenfalls Platz und deswegen fehlen zwei Kilobyte bei der Mitteilung des freien Speicherplatzes. Zieht man diese von den berechneten Zahlen ab, dann stimmt alles wieder.

Nur 64 Directory-Einträge sind möglich

Damit sind wir aber bereits an einem weiteren Punkt angelangt. Das Inhaltsverzeichnis (Directory) belegt also 2048 Bytes. Da jeder Fileeintrag 32 Bytes belegt, sind aus diesem Grunde auch nur 64 Fileeinträge im Directory möglich. Das heißt, daß nicht mehr als 64 Programme (und seien sie auch noch so kurz) pro Diskettenseite gespeichert werden können. Der Kapazitätsumfang des Inhaltsverzeichnisses setzt dabei die Grenze! Hiervon und auch noch von anderen Dingen, können sich anhand der Programme, die ich Ihnen in diesen Grundlagen-Artikeln gebe, auch in der Praxis überzeugen. Doch bevor wir zur Entwicklung dieser Programme übergehen, müssen noch ein paar weitere Punkte geklärt werden.

Damit die Diskettenstation vom CPC „angesprochen“ werden kann, befindet sich im CPC (664 und

6128) oder im aufsteckbaren Interface beim CPC 464, AMSDOS (AM-Strad Disc Operating System), das Disketten-Betriebs-System. Dieses erlaubt es, Diskettenfiles in der gleichen Art und Weise wie Kassettenfiles anzusprechen. Es steckt in einem der sogenannten „Sideways-ROMs“, die vom CPC angesprochen werden können. Zusatzroms sind bei den CPC's auf hervorragend durchdachte Weise in das Betriebssystem der CPCs „einbindbar“. Während der Initialisierungsphase des Computers wird – aufgrund verschiedener Speicherstelleninhalte in den Zusatzroms – nämlich abgeprüft, welche ROMs im Vordergrund oder Hintergrund arbeiten, bzw. ob es sich um sogenannte „On-board“-ROMs handelt. Eine mir – in dieser Art – bisher nur von den CPCs her bekannte Technik, die äußerst leistungsstark ist und deshalb (bezüglich Betriebserweiterungen) vergleichbare Computer um Längen schlägt. Wer mehr über diese Methode erfahren will, kann einen Teil darüber im ersten Teil des ComputerSchau-Kompaktwissens nachschlagen oder – falls er voll in die Thematik einsteigen will – sich das Firmwarehandbuch zulegen. Zum augenblicklichen Verständnis dieser Grundlagenartikel reicht es aber, zu wissen daß der CPC ROMs, die in den Bereich ab &c000 „eingeklinkt“ werden können, ansprechen kann und daß diese ROMs eine Auswahlnummer haben müssen. Diese Auswahlnummer (Select-number) wird benötigt damit bei evtl. Zugriffen auf solche Erweiterungen, auch der richtige Chip angesprochen wird. Die Auswahlnummer des Disketten-ROMs (BIOS-ROM) ist mit sieben festgelegt. BIOS heißt Basic Input/Output System, hat aber mit der Programmiersprache Basic nichts zu tun. In diesem ROM steckt auch AMSDOS und ein Teil von Dr. LOGO.

Mit diesen Informationen können Sie nun bereits tiefer einsteigen. Mit z. B. dem Hexpeeker aus dem ersten Heft oder (natürlich viel sinnvoller!) mit einem leistungsfähigen Disassembler/Monitor kann man sich nach Auswahl der Selectadresse dieses DOS (Disk-operating-system) näher ansehen.

Neun sehr leistungsfähige Befehle

„Maschinenprogrammierer“ werden bei näherer Betrachtung schnell die „Sprungleiste“, also den (auf neudeutsch) „Jump block“ der Diskettencontrollerbefehle finden. Diese Sprungbefehle stehen von &C033 bis &C04D im – dem Bildschirm-RAM überlagerten – BIOS-ROM. Die Befehlscodes für diese (insgesamt neun) Befehle stehen von &C0B6 bis &C0BF und gehören zum CP/M-BIOS, stehen aber auch dem AMSDOS zur Verfügung. Wer hier noch tiefer „eintauchen“ will, dem kann – als Zusatzliteratur zum Handbuch – das von Data Becker herausgegebene Floppy-Buch empfohlen werden. Trotz der (in den Listings) leider enthaltenen Fehler ist es ein brauchbares Informationswerk. Wer einen „guten Draht“ nach England hat und auch noch der englischen Sprache mächtig ist, der findet im „Disc Firmware Supplement“ ebenfalls viel an tiefergehenden Informationen. Auch die in manchen Veröffentlichungen als „versteckt“ genannten Befehle und deren Handhabung sind dort ausführlich beschrieben. Vielleicht gibt es bis zum Erscheinen dieses Heftes aber auch bereits eine deutsche Übersetzung hiervon.

Für uns reicht es im Augenblick zu wissen, daß es im Disketten-Betriebssystem sowohl einen Befehl für das Lesen, als auch für das Schreiben eines Sektors gibt. Außerdem ist noch von Interesse, daß das Betriebssystem der CPCs eine Firmware-routine für das Auffinden von Befehlen und Routinen bereitstellt. Diese Routine findet auch Befehle von externen ROMs. Sie hat den Namen „KL FIND COMMAND“ und wird durch call &BCD4 aufgerufen. Beim Einsprung in diese Routine, ist nur eine Einsprungbedingung erforderlich, das „HL“-Register muß die Adresse des Kommandonamens, nach dem gesucht wird, enthalten. Der CPC prüft daraufhin, ob ein RSX-Befehl oder ein Hintergrund-ROM-Befehl, der diesen Spezifikationen entspricht, vorhanden ist. Durch „Carry-Flag“ an oder aus, wird der Verlauf der Suche signalisiert.

siert. Carry-Flag an heißt gefunden, im anderen Falle nicht gefunden. Da ich voraussetze, daß das CPC-System ordnungsgemäß arbeitet und die Programme, die Sie später einsetzen, korrekt abgeschrieben wurden, brauchen wir uns nur um die „erfolgreiche“ Suche zu kümmern. Nach der Rückkehr aus der Suchroutine enthält das Register „C“ die ROM-Auswahladresse (ROM-select) und „HL“ enthält die Adresse der gesuchten Routine. Die-

se Registerinhalte müssen in Speicherstellen geladen werden, auf die ein „Folgeprogramm“ zugreift. Damit haben wir nun die Voraussetzungen für den direkten Zugriff auf die Diskette.

Welche „Mächtigkeit“ Sie mit diesem Wissen haben, wird Ihnen nach Lektüre dieser Grundlagenartikel und deren Anwendungsbeispielen bestimmt klar werden. Ich kann Ihnen vorab nur mitteilen, daß Ihnen Möglichkeiten offenstehen, die Ihre

Erwartungen vermutlich weit übertreffen. So ist es möglich, sehr komfortabel gelöschte Files wieder zurückzuholen, sogenannte RANDOM-Dateien aufzubauen, „COM-Files“ in Binärfiles zu ändern und vieles mehr. Einzige Voraussetzung, daß auch Sie all dies programmieren können ist Ihr Wille. Wie es geht wird in diesem Heft durch einige Beispiele gezeigt. Im nächsten Kapitel führe ich Sie dann wirklich ins „Eingemachte“.

L. Miedel

```

! ! ! ! ! ! ! ! ! ! ! ! ! ! ! !
Wichtiger Hinweis
fuer die Benutzer
des Programmes :

      CPC-
Dateiverwaltung
(Version d)

      aus Heft 1 !
! ! ! ! ! ! ! ! ! ! ! ! ! ! ! !

```

Falls die CPC-Dateiverwaltung
Version d
im Kassettenrecorderbetrieb eingesetzt
werden soll, ist es anzuraten den
Basicteil durch folgende Zeilen
zu ergaenzen:

```

1731 if len(datein$)<8 then datein$=
      datein$+" ":goto 1731
1816 if len(datein$)<8 then datein$=
      datein$+" ":goto 1816

```

Sonst muss bei Dateinamen die kuerzer
als 8 Zeichen sind, beim Laden
mit Spaces (Leerzeichen) auf die Anzahl
von acht Zeichen aufgefuellt werden !

Arbeiten mit Diskette (2)

Maschinenprogramm für den Zugriff auf einen Sektor

Aufbauend auf den vorhergehenden Artikel will ich nun mit Ihnen gemeinsam eine Routine entwickeln, die den direkten Zugriff auf einen bestimmten Sektor einer Diskette ermöglicht. Dieser Routine dient auch für spätere Programme als Unteroutine. Deshalb will ich Ihnen die genaue Arbeitsweise erklären. Da ein sinnvoller Zugriff nur in Maschinensprache geschehen kann, werden wir uns diesem „Basisteil“ zuerst widmen.

Um ein derartiges Maschinenprogramm entwickeln zu können, bedient man sich sinnvollerweise eines Assemblerprogrammes. Im letzten Heft habe ich bereits einen Assembler/Disassembler/Editor getestet, mit dem ich auch heute noch arbeite. Zwischenzeitlich besitze ich nun zwei Versionen, da sich – wie im Testbericht bereits angedeutet – die Version 1.09 nicht auf dem 6128 oder 664 bewährte. (Das Eprom war nicht schnell genug!) Für Interessenten an diesem ROM-Programm sei erwähnt, daß – aufgrund der Unterschiede der englischen und deutschen 6128-Versionen – die Adaption nur mit einem selbstgebastelten Adapterstecker geschehen kann (Stand: 1. 11. 1985). Dies ist zwar für einen versierten Bastler nicht schwierig, für einen „elektronischen Linkshänder“ aber eine kaum überwindbare Hürde. (Hoffentlich nimmt sich demnächst

jemand dieser Sache an, sonst gibt es immer wieder Probleme mit aus England stammenden Erweiterungen, wenn diese an den CPC 6128 angeschlossen werden sollen.)

Das Programm wird erstellt

Ich schildere nun die Vorgehensweise mit dem schon erwähnten Assembler. Dieser muß natürlich erst aktiviert und in den Editormode gebracht werden. Als erstes teilen wir dem Assembler die Adresse mit, ab welcher das Maschinen-Programm abgelegt werden soll. Da es sich um eine sehr kurze Routine handelt, kann diese Adresse sehr weit oben, also beispielsweise auf &A600 gesetzt werden. Mit dem „MAXAM im ROM“ kann allerdings der Test der Routine an dieser Stelle nicht erfolgen, denn MAXAM ist dort selbst aktiv. Während der Austestphase kann dieses Programm beispielsweise auf &A000 gelegt werden, um es später, wenn man sich von der Fehlerfreiheit überzeugt hat, für den Bereich ab &A600 zu assemblieren. Betrachten Sie nun bitte die Zeilennummer 6 des Assemblerlistings. Dort wird durch `org &A600` bestimmt, daß das Maschinen-Programm ab dieser Adresse abgelegt werden soll. Als nächstes legen wir die Adresse für das aktive Laufwerk fest. Dies hat den Vorteil, daß wir später auch auf ein zweites Laufwerk zugreifen können. Die Spei-

cherstelle, in der bei den CPCs die Nummer des aktiven Laufwerks abgelegt wird, ist &A702. Sinnvollerweise nennen wir diese Speicherstelle „akt driv“. Die Einsprungsadresse für die Suche nach einem Kommando ist wie im letzten Abschnitt schon erwähnt, &BCD4. Der Name hierfür wurde mit `findco`, der Abkürzung für „Finde Kommando“ gewählt. Als nächstes muß die Tabelle zur Aufnahme bestimmter Werte aufgebaut werden. (Ein komfortabler Assembler ermöglicht jederzeit das „Zwischenschieben“ von Befehlen, Direktiven usw. in den bestehenden Sourcecode.) Da später auch eine Beeinflussung des Maschinenprogrammes von Basic aus erfolgen soll, werden in dieser Tabelle Speicherplätze für verschiedene „Merker“ reserviert. Für unser Programm benötigen wir Platz für den Befehl (`befehl = 1 Byte`), die ROM-Adresse (`romadr = 2 Byte`), für ROM-Select (`romsel = 1 Byte`), für die Laufwerksnummer (`driven = 1 Byte`), für die Spur-Nummer (`track = 1 Byte`), für Format und Sector-Nummer (`formse = 1 Byte`) und schließlich noch für den Buffer aus dem oder in den die Daten gelesen oder geschrieben werden (`buffer = 2 Bytes`).

Tabellen bringen Vorteile

Durch den Aufbau von Tabellen gewinnen wir den „Freiheitsgrad“, spä-

ter auf einfache Weise die gewünschten Werte – von Basic aus – in das Maschinenprogramm poken zu können. Außerdem werden beim Assemblerlauf (falls gewünscht) die Adressen mit den Namen ausgegeben, was bei der Basicprogrammerstellung mehr oder weniger komplizierte Berechnungen und Überlegungen überflüssig macht.

Nun folgt das eigentliche Maschinenprogramm, das wir nun zwischen den ersten Teil und der Tabelle einschieben.

Zeile 11 (also ab der Adresse &A600) bewirkt, daß der Akkumulator mit dem Inhalt der Speicherstelle für das aktive Laufwerk geladen wird. Dadurch kann beim späteren Programmlauf der automatische Zugriff auf das jeweils ausgewählte Laufwerk erfolgen. Dieser Wert wird dann sofort (Zeile 12) in die „Merker“-Speicherstelle der Drive-(Laufwerks-)Nummer übertragen. Anschließend wird die Adresse des Befehles geholt und die Suchroutine aufgerufen. Da bei intaktem CPC und korrekt angeschlossenem Laufwerk die Suche erfolgreich sein muß, können die in den Registern „HL“ und „C“ enthaltenen Werte der Suchroutine direkt in die reservierten Speicherstellen übertragen werden (Zeile 15 bis 17). Um über einen sogenannten Restart die eigentliche Ausführung des Befehles zu bewirken, sind noch weitere Voraussetzungen zu berücksichtigen. Die Laufwerksnummer muß im E-Register stehen, das D-Register muß die Spur-Nummer enthalten, im C-

Register muß die Sektornummer (inklusive der Formatkennzeichnung) stehen und zu guter Letzt muß „HL“ die Bufferadresse enthalten. Dies alles geschieht in den Zeilen 18 bis 24. Beim Restart selbst wird außerdem noch die ROM-Adresse der eigentlichen Routine „mitgenommen“.

Wird beim späteren Programmlauf die Routine abgearbeitet, dann steht im gewünschten RAM-Speicherbereich (Buffer) genau der gewünschte Sektorinhalt der Diskette.

Ein Byte mit großer Wirkung

Da wir in der Tabelle den Befehls-wert mit &84 festgelegt haben, handelt es sich im Moment um ein Programm zum Lesen eines Sektors.

Ändert man nur den Befehlscode in korrekter Weise, dann wird nicht ein Sektor gelesen, sondern geschrieben. Dabei können alle anderen Parameter in der gleichen Art übergeben werden. Um es nochmals anders auszudrücken, bei Änderung des Befehles von &84 in &85 werden 512 Bytes ab dem festgelegten Bufferanfang in den, in den anderen Registern festgelegten Sektor geschrieben. So einfach geht das! Der Rest des Assemblerlistings dürfte keine Verständnisschwierigkeiten bereiten. Die Bufferadresse und formse (Format und Sektor) wurden zwar im Assemblertext definiert, aber wir können von Basic aus ja jeden Tabellenwert verändern.

Damit haben wir nun eine Grundroutine erstellt, die uns bei allen

weiteren „direkten“ Arbeiten mit der Diskettenstation wertvolle Dienste leisten kann. Da nicht jeder CPC-Besitzer über einen Assembler verfügt, habe ich selbstverständlich auch einen „Dataloader“ erstellt, der das entsprechende „Binärfile“ automatisch auf der Diskette generiert. Die Basicprogramme laden dann jeweils dieses erzeugte Binärfile (Sector.bin) nach, also nicht den Dataloader!

Ich habe es im letzten Heft zwar schon einmal erwähnt, aber doppelt „genäht“ hält besser. Vergleichen Sie doch einmal die Daten des Dataloaders mit den Daten des Assemblerlistings. Wie Sie sehen, sind beide gleich. Der Dataloader ist also nur eine „Krücke“, um Maschinenprogramme in eine druckbare und direkt eingebbare Form zu bringen. Vor dem Lauf des Dataloader sollten Sie aber unbedingt den Maschinensprachebereich durch MEMORY &A5FF vor dem „Überschreiben“ schützen. Dies können Sie beispielsweise auch im Programm selbst tun. Wenn Sie es nämlich vergessen, kann es passieren, daß dort ganz andere Werte stehen, denn Stringvariable werden ja von HIMEM abwärts im Speicher des CPC abgelegt. Auch die späteren Basicprogramme müssen diesen Bereich „schützen“.

Alles was jetzt noch zu tun ist, kann mittels eines Basicprogrammes bestimmt werden.

Im nächsten Kapitel werden wir dies für einen ersten Anwendungsfall tun. Lothar Miedel

ARNOR Z80 ASSEMBLER version 1.10

Page 001

```

00002 ;*****
00003 ;*** SECTOR (c) LM ***
00004 ;*****
00005 A474 write "sector.bin"
00006 A600 (A600) org &a600
00007
00008 A600 (A702) aktdriv equ &a702
00009 A600 (BCD4) findco equ &bcd4
00010
00011 A600 3A 02 A7 ld a,(aktdriv) ;aktive drive# holen
00012 A603 32 26 A6 ld (driven),a ;und in tabelle
00013 A606 21 22 A6 ld hl,befehl ;Befehlsadresse
00014 A609 CD D4 BC call findco ;Adresse und Select holen
00015 A60C 22 23 A6 ld (romadr),hl ;Sprungadresse in Tabelle

```



```

00016 A60F 79          ld    a,c          ;Romsel uebergeben
00017 A610 32 25 A6    ld    (romsel),a    ;und in Tabelle ablegen
00018 A613 21 26 A6    ld    hl,driven    ;Adresse fuer drive holen
00019 A616 5E          ld    e,(hl)      ;drive# in's E-Register
00020 A617 23          inc    hl          ;Zeiger auf track
00021 A618 56          ld    d,(hl)      ;track in's D-Register
00022 A619 23          inc    hl          ;Zeiger auf formse
00023 A61A 4E          ld    c,(hl)      ;Format + Sector in C
00024 A61B 2A 29 A6    ld    hl,(buffer) ;Bufferadresse holen
00025 A61E DF 23 A6    rst    3,romadr    ;Befehl ausfuehren
00026 A621 C9          ret              ;zurueck nach Basic
00027                  ;*****
00028                  ;* Tabelle *
00029                  ;*****
00030 A622              .befehl
00030 A622 84          defb    &84          ;Lesebefehl
00031 A623              .romadr
00031 A623 00 00      defb    &00,&00      ; ROM-Einsprung
00032 A625              .romsel
00032 A625 00          defb    &00          ;hier Romselect
00033 A626              .driven
00033 A626 00          defb    &00          ;enthaelt Drive-#
00034 A627              .track
00034 A627 00          defb    &00          ;Track-#
00035 A628              .formse
00035 A628 C1          defb    &c1          ; Format + Sector
00036 A629              .buffer
00036 A629 00 90      defb    &00,&90      ;Bufferadresse
00037

```

Errors: 00000 Warnings: 00000

SYMBOL TABLE:

A702 AKTDRIV	A622 BEFEHL	A629 BUFFER	A626 DRIVEN
BCD4 FINDCO	A628 FORMSE	A623 ROMADR	A625 ROMSEL
A627 TRACK			

Listing: SECTOR (Dataloader)

```

100 'Dataloader Maschinenprogrammroutine ''Sector''
110 '
120 a=&A600:e=&A62A:zb=1000:e=e+1
130 FOR i =a TO e:READ d$:IF LEFT$(d$,1)="" THEN flag =1
140 IF (flag AND ps<>VAL(d$)) THEN PRINT"Fehler in Zeile "zb+1:END
150 IF (flag AND i=e) THEN 190
160 IF flag THEN i=i-1:zb=zb+1:ps=0:d$="":flag = 0:GOTO 180
170 d$=""&d$:POKE i,VAL(d$):ps=ps+VAL(d$):
180 IF i < e THEN NEXT i
190 SAVE"sector.bin",b,&A600,43
200 PRINT"M Maschinenprogramm ''Sector'' geladen und abgespeichert":END
210 '
1000 'Das fuer das Maschinenprogramm
1001 DATA 3A,02,A7,32,26,A6,21,22,A6,CD,D4,BC,22,23,A6,79,&068B
1002 DATA 32,25,A6,21,26,A6,5E,23,56,23,4E,2A,29,A6,DF,23,&052D
1003 DATA A6,C9,84,00,00,00,00,00,C1,00,90,&0344

```


Arbeiten mit Diskette (3)

Lesen eines Diskettensektors

Nun „steht“ also unser Maschinenprogramm zum Lesen und Schreiben von Diskettensektoren. Um dieses herum bauen wir nun unsere Basicprogramme auf.

Als erstes nun ein Programm, welches ich aufgrund einer Leser Anregung geschrieben habe. Ein Leser der ComputerSchau hatte mich gebeten, ihm bei seinem Problem zu helfen. Es ging darum, ein CP/M-COM-File in ein Binärfile umzuwandeln.

Dieser Leser hatte für einen anderen Computer (VZ200) eine Befehlserweiterung in Assembler geschrieben. Diese zusätzlichen Befehle sollten in ein Eprom vom Typ 27128 gebrannt werden, um danach ein eingebautes ROM zu ersetzen. Zur Erstellung des Sourcecodes benutzte der Leser den Macro-80-Assembler unter CP/M 80 auf dem Schneider CPC 464. Der Output (das Ergebnis) war eine Hex-Datei, bzw. ein lauffähiges Programm mit dem Extend .COM.

Zum Brennen des Eproms war ein EPROM-Programmer (Typ 4003) von der Firma Dobbertin vorhanden. Dieser Eprommer wird auch für die CPC's angeboten. Das Problem hierbei war nun, die COM-Datei in eine BIN-Datei umzuwandeln. Wäre es aber nur um den Extend gegangen, wäre dies gewiß kein Problem gewesen. Aber es ging genauer gesagt darum, daß die BIN-Datei ab einer bestimmten Speicheradresse im CPC stehen mußte, damit die Prommersoftware korrekt loslegen konnte. Der Leser konnte

bis zu diesem Zeitpunkt kein Programm finden, das diese Umsetzung hätte bewirken können. Dies hätte bedeutet, daß insgesamt 16 KByte Maschinenprogramm hätten von ihm abgetippt werden müssen. Wer jemals längere Maschinenprogramme abgetippt hat, kann sich vorstellen, welches mühselige und auch fehlerbehaftete Unterfangen eine derartige Unternehmung darstellt. Das von mir an den Leser übersandte Programm, half ihm aus diesem Dilemma. Demnächst will dieser Leser auch noch die ROMs des CPC in Angriff nehmen. Da seit kurzer Zeit für die CPC's Erweiterungskarten angeboten werden, ist deshalb, so glaube ich, das Problem für viele Leser, die sich eigene Erweiterungseproms brennen wollen, von großem Interesse. Deshalb ist hier eine Möglichkeit aufgezeigt, wie auch Sie derartige „Problemchen“ meistern können.

Etwas Gedankenakrobatik vorher

Es wäre von mir aber, so glaube ich, nicht fair, Ihnen einfach die Lösung anhand des Programmes vorzulegen. Deshalb will ich Ihnen meine Gedankengänge, die mich zur Lösung des Problems brachten aufzeigen. Auch hier erfahren Sie wieder einiges über Ihre Disketten. Bereits im ersten der Grundlagenartikel hatte ich erwähnt, daß jeder Fileeintrag im Inhaltsverzeichnis 32 Bytes belegt. Erinnern Sie sich noch? Noch nicht erfahren hatten Sie aber wo diese Informationen auf der Diskette stehen. Je nach Format

ist dies unterschiedlich. Trotzdem ist die Kurzantwort sehr einfach: Diese Informationen sind in den ersten vier verfügbaren Sektoren zu finden. Im Falle einer Data-only-Diskette heißt dies, Spur 0, Sektor 1 bis 4.

Beim Systemformat aber: Spur 2, Sektor 1 bis 4.

Damit Sie sich dies nun näher ansehen können, sollten Sie sich zunächst einmal die Maschinenroutine mittels eines Basicteiles so ausbauen, damit Sie sich dies näher betrachten können. Hierzu brauchen Sie lediglich das Basic-Programm „Readsec1“ abzutippen und zu starten. Berücksichtigen Sie dabei aber, daß beim Programmstart auf der in das Laufwerk eingelegten Diskette das Programm „SECTOR.BIN“ verfügbar sein muß. Außerdem klappt es bei diesem Programm nur, wenn Sie eine Diskette im Data-only-Format benutzen. Bei den weiteren Programmen sind dann alle beiden Formate les- und schreibbar. Readsec1 ist nur ein kleines Programm für die weiteren Erklärungen.

Nach dem Programmstart wird der Maschinencodeteil eingelesen. Danach werden Sie aufgefordert die zu untersuchende Diskette einzulegen. Haben Sie dies getan und dann eine Taste gedrückt, dann wird der erste verfügbare Sektor in den gewählten Buffer eingelesen. Anschließend wird der Inhalt des Buffers ausgelesen und sowohl im Hexformat als auch in ASCII auf dem Bildschirm dargestellt. Lassen Sie beim ersten Male ruhig das Programm ganz durchlaufen. Beim zweiten Mal

sollten Sie nach Ausgabe der ersten „Bufferinhaltszeilen“ aber mittels der ESC-Taste stoppen. Denn nun wollen wir uns dieses Ergebnis etwas näher betrachten, um weitere Schlüsse ziehen zu können. Die anderen Programme sind weit komfortabler, aber dieses Programm soll uns wirklich nur zur Veranschaulichung und zu Erklärungen dienen. Deswegen fehlt ihm auch jeglicher Bedienungskomfort. Da ein File, wie schon bekannt, 32 Bytes des Directoryeintrages benötigt, ist die in dem kleinen Programm gewählte Bildschirmdarstellung ideal, um Erkenntnisse zu erarbeiten. Zusätzlich wird durch das Programm nach jedem Fileeintrag eine Trennungslinie erzeugt. Dies dient nur dazu um alles noch übersichtlicher zu machen.

So ist ein Fileeintrag codiert

Die erste Hexzahl, die Sie auf dem Bildschirm sehen, ist vermutlich „00“ oder „E5“. Dieses erste Byte ist die User-Nummer. Falls Ihnen „Usernummer“ unbekannt ist, sehen Sie doch bitte im Handbuch nach. Bei „&E5“ handelt es sich um ein „gelöschtes“ File. Die wirklichen Usernummern können nur zwischen 0 und 15 liegen. Ist an dieser Stelle aber ein Wert ungleich E5 und größer 15, dann hat jemand an diesem Eintrag manipuliert! Durch derartige Spielereien können Einträge oder auch natürlich die zu diesen gehörenden Files selbst, auf die vielleicht direkt zugegriffen wird, vor den normalen Blicken und auch vor dem „Löschen“ geschützt werden.

Anschließend kommen 11 Hexzahlen welche die ASCII-Werte des File-Namens und der „Extension“ darstellen. Wie Sie sich überzeugen können, gibt es dazwischen keinen Punkt, sondern Name und Extend werden unmittelbar hintereinander abgespeichert.

Die nächste Hexzahl gibt Auskunft darüber, ob das File länger als 16 KB ist und sich aus diesem Grunde über mehrere Einträge erstreckt. Es ist die Directory-Eintragsnummer. Im Falle von „00“ ist es der erste Eintrag. Im Falle von „01“ der

zweite usw. Die nächsten beiden überspringen wir nun kurz. Die letzte Zahl der ersten Zeile gibt die Anzahl der für das File benötigten Records wieder. (Ein Record sind 128 Bytes, dies stammt noch aus den Urzeiten von CP/M!)

Wenn Sie diesen Wert mit 128 multiplizieren, dann haben Sie die Länge die das File auf der Diskette belegt. Steht an dieser Stelle der Wert &80, dann gibt es noch einen weiteren Teil dieses Files. In diesem Falle gibt es dann mindestens einen weiteren Fileeintrag mit dem gleichen Namen, dessen zwölftes Byte dann keine Null, sondern eine „01“ oder höhere Nummern, je nach Anzahl der Einträge, beinhaltet. Die Bytes 15 bis 31 enthalten die Blocknummern, über die das Programm auf der Diskette verteilt ist. Nun kommt etwas ganz wichtiges! Ein Block sind immer zwei aufeinanderfolgende Sektoren! Aufgrund dieser Informationen wissen Sie nun auch, weshalb ein Programm immer nur „ganze“ Kilobytwerte als Länge hat. Halbe Blöcke in diesen Einträgen (dies würde einem Sektor entsprechen) gibt es nicht im Inhaltsverzeichnis. Selbst wenn also ein Programm nur ein paar Bytes umfaßt, werden dadurch auf der Diskette zwei (!) Sektoren belegt. Dieses Wissen benötigen wir später nochmals. Nun wissen Sie prinzipiell aber eigentlich alles, was für die Erstellung eines Programmes, um den eingangs erwähnten Leser aus der Patsche helfen zu können, notwendig ist.

Da das in diesem Heft abgedruckte Programm „Reader“ aber noch weit mehr kann, als bisher erwähnt, ist es auch für Leser interessant, die keine Eproms brennen wollen. Denn es gibt noch einige weitere Informationen preis! Das im Heft abgedruckte Programm akzeptiert auch Systemformat und ist ausreichend kommentiert, deshalb unterbleibt die genaue Erklärung.

Die Bedienung von READER

Zur Bedienung aber noch ein paar Sätze. Nach dem Programmstart erhalten Sie einige Informationen über das Programm selbst. Da das

Programm feststellt, daß im ersten Byte des Maschinenprogrammbereiches noch keine 58 steht, wird der Maschinenteil erst geladen.

Daraufhin werden Sie aufgefordert, die zu untersuchende Diskette einzulegen und eine Taste zu drücken. Durch einen kleinen Trick erkennt das Programm das Diskettenformat und liest die vier entsprechenden Directorysektoren ein.

Danach wird Ihnen das komplette Verzeichnis aller Fileeinträge ausgegeben. Ein „+“ hinter dem jeweiligen Extend zeigt an, daß das File gültig ist. Ein „-“ signalisiert, daß dieses File gelöscht ist. Dabei ist nun folgendes von großer Wichtigkeit: Beim Löschen mit ERA wird nur die User-Nummer auf &E5 gesetzt und dadurch auch die zugehörigen Blöcke freigegeben. Das Programm selbst ist aber noch da, wenn noch nicht neu auf diese Diskette geschrieben wurde. Im Programm „FRETTER“ des nächsten Kapitels wird dies ausgenutzt.

Denken Sie deshalb bitte immer daran, wenn Sie Disketten weitergeben und nicht wollen, daß plötzlich jemand über Ihre Programme oder Daten verfügt. Wie schnell diese zurückgeholt sind, können Sie bald selbst sehen. Nach der Ausgabe aller Fileeinträge werden Sie aufgefordert, eine Filenummer einzugeben, wenn Sie mehr darüber wissen wollen. Damit ist die in Klammer stehende Hexadezimalzahl gemeint, die gleichzeitig auch die laufende Nummer im Directory-Eintrag darstellt.

Nachdem Sie eine entsprechende Nummer eingegeben haben, erfahren Sie Usernummer, die benutzten Blöcke, die Verteilung über die Sektoren usw. Nun können Sie auswählen, ob Sie andere Einträge ansehen wollen, oder ob dieses File geladen werden soll. Das Einladen eines Files kann nun z. B. zu Programmierzwecken geschehen. Aber auch andere Gründe für einen derartigen Ladevorgang kann es geben. Z. B. Retten von teilweise schon überschriebenen Files, wenn der FRETTER nicht mehr in der Lage ist alles korrekt zurückzuholen und noch weitere Gründe.

Probieren lohnt sich...

Lothar Miedel

Listing: Reader

```

100 'READER (c) by LM
110 '
120 MODE 2:MEMORY &2FFF:PRINT"Programm:  READER
130 '
140 PRINT"Dieses Programm liest alle File-Eintraege in einen Buffer ab &3000.
150 PRINT"Danach koennen weitere Informationen ueber die Eintraege abge-
160 PRINT"rufen werden. Dies betrifft z.B. die User-Nummer, die File-Namen
170 PRINT"und die Bloecke, ueber die das jeweilige File verteilt ist.
180 PRINT"Anschliessend kann das File eingelesen werden, um z. B. fuer
190 PRINT"Untersuchungen oder fuer einen 'Eprommer' die Daten im Speicher
200 PRINT"stehen zu haben.
210 PRINT"Die Bufferadresse wurde mit &3000 gewaehlt, da ein handelsueblicher
220 PRINT"EPROMMER diese Bufferadresse benoetigt.
230 PRINT"Die Maschinencoderoutine SECTOR muss als Binaerfile vorliegen!
240 PRINT:PRINT"Bitte eine Taste druecken":CALL &BB06:CLS
250 '
260 '
270 'Adressen fuer Maschinenprogramm
280 akt drivemc=&A702:befehlmc=&A622:buffermc=&A629:drivenmc=&A626
290 formsemc=&A628:trackmc=&A627:mcstart=&A600
300 '
310 buffer$="3000":'          Beginn des Einlesebereiches
320 '
330 DIM track(16),sector(16):' maximal 16 KB
340 '
350 IF PEEK(&A600) = 58 THEN 380
360 LOAD"sector.bin",&A600:'   Maschinencode einlesen
370 '
380 PRINT"Bitte die entsprechende Disk einlegen und Taste druecken":CALL &BB06
390 '
400 'Dieser kleine Trick hilft das Format zu erkennen
410 OPENOUT"dummy":CLOSEOUT:drive=PEEK(akt drivemc)
420 '
430 'je nach Drive das entsprechende Format abholen
440 IF drive = 0 THEN form = PEEK(&A89F)-1:' Diskformat drive #0 abholen
450 IF drive = 1 THEN form = PEEK(&A8DF)-1:' Diskformat drive #0 abholen
460 '
470 'Den durch das Format bedingten Offset festlegen
480 IF form =&C0 THEN offset =0:fo$="Data-Only-Format
490 IF form =&40 THEN offset =2:fo$="Systemformat
500 IF offset = 2 THEN track =2
510 '
520 'Werte fuer Bufferadresse berechnen
530 bufflow=VAL("&" +RIGHT$(buffer$,2)):buffhigh=VAL("&" +LEFT$(buffer$,2))
540 '
550 '*****
560 '* Komplettes Inhaltsverzeichnis einlesen *
570 '*****
580 sector = sector +1
590 GOSUB 1800:'              Sector einlesen
600 '
610 ' Falls Sektor = 4, dann ist komplettes Inhaltsverzeichnis eingelesen

```



```

620 IF sector=4 THEN 710
630 '
640 'Buffer um 2*256 = 512 Bytes erhoeuen
650 buffhigh=buffhigh+2:GOTO 580
660 '
670 '*****
680 '* Nun Auswertung und Ausgabe *
690 '*****
700 '
710 lesezeiger =VAL("&" +buffer$):CLS
720 PRINT"Verzeichnis der File-Eintraege (+ =gueltig/- =geloescht)":PRINT
730 blcnt=0:filename$=""
740 usernummer=PEEK(lesezeiger)
750 IF usernummer=&E5 THEN file$=" -":' File geloescht
760 IF usernummer<&16 THEN file$=" +"
770 '
780 ' File-Namen holen
790 FOR i = lesezeiger+1 TO lesezeiger+12
800 z$=CHR$(PEEK(i))
810 '
820 ' evtl. gesetzte Bits ausblenden
830 IF ASC(z$)>128 THEN z$=CHR$(ASC(z$)-64):GOTO 830
840 filename$=filename$+z$
850 NEXT i
860 filename$=filename$+file$
870 lesezeiger =lesezeiger+32
880 '
890 'Pruefung ob letzter File-Eintrag
900 IF LEFT$(filename$,1)=CHR$(101) THEN fz=65:GOTO 940
910 fz=fz+1:PRINT filename$;"(HEX$(fz,2)) ";
920 filename$=""
930 '
940 IF fz<64 THEN 740
950 '
960 '*****
970 '* Alle Eintraege ausgegeben *
980 '*****
990 '
1000 PRINT:PRINT:INPUT"Informationen ueber welches File ";f$
1010 f$="&" +f$:fz=VAL(f$):filename$=""
1020 '
1030 lesezeiger=VAL("&" +buffer$)+(fz-1)*32
1040 '
1050 usernummer=PEEK(lesezeiger)
1060 '
1070 'Namen holen
1080 FOR i = lesezeiger+1 TO lesezeiger+12
1090 z$=CHR$(PEEK(i))
1100 IF ASC(z$)>128 THEN z$=CHR$(ASC(z$)-64):GOTO 1100
1110 filename$=filename$+z$
1120 NEXT i
1130 '
1140 CLS:PRINT"Informationen ueber : "filename$,;"Eintrags-Nr.: "fz:PRINT
1150 FOR i =0 TO 16:track(i)=0:sector(i)=0:NEXT
1160 lesezeiger=lesezeiger+12:dirnum=PEEK(lesezeiger)

```



```
1170 lesezeiger=lesezeiger+3:records=PEEK(lesezeiger)
1180 lesezeiger=lesezeiger+1
1190 '
1200 PRINT"User-Nummer: "username;" Directory-Eintrags-Nummer: "dirnum:PRINT
1210 PRINT"Das File ist ueber folgende Bloecke verteilt: ":PRINT
1220 pw=PEEK(lesezeiger)
1230 IF pw=0 THEN PRINT: GOTO 1290
1240 IF pw=229 THEN PRINT: GOTO 1290
1250 PRINT HEX$(pw,2),;:blcnt=blcnt+1
1260 sector(blcnt)=pw
1270 lesezeiger=lesezeiger+1:GOTO 1220
1280 '
1290 PRINT:PRINT"Dies entspricht folgenden Track- u.-Sector-Nummern (T,S/T,S):"
1300 ges$="":FOR i = 1 TO 15:IF sector (i)=0 THEN i=16:GOTO 1390
1310 '
1320 '*****
1330 'Tracks und Sektoren aus Block-Nummer berechnen
1340 '*****
1350 b=sector(i)
1360 bn=(b*2+offset*9)+1:track=FIX(bn/9):sector=bn-track*9
1370 GOSUB 1490:PRINT track;","sector"/";GOSUB 1540:sector =sector+1
1380 GOSUB 1490:PRINT track;",";sector,;:GOSUB 1540
1390 NEXT i:PRINT
1400 '
1410 PRINT:PRINT"Bytes lt. Eintrag : "records*128:PRINT
1420 PRINT:PRINT"das sind "blcnt" *1 K, und entspricht "records" Records."
1430 PRINT:PRINT"Infos ueber anderes File (j/n) ?"
1440 a$=LOWER$(INKEY$):IF a$="" THEN 1440 ELSE IF a$="n" THEN 1560
1450 IF a$<>"j" THEN 1440
1460 fz=0:GOTO 710
1470 '
1480 'bei Berechnung Korrekturen vornehmen
1490 IF sector >9 THEN sector=1:track=track+1
1500 IF sector =0 THEN sector=9:track=track-1
1510 RETURN
1520 '
1530 'String fuer spaeteres einlesen zusammenbauen
1540 t$=HEX$(track,2):s$=HEX$(sector,2):ges$=ges$+t$+s$:RETURN
1550 '
1560 PRINT:PRINT"Soll dieses File geladen werden ?"
1570 a$=LOWER$(INKEY$):IF a$="" THEN 1570
1580 IF a$<>"j" THEN 1750
1590 '
1600 ' File wird ab Bufferbeginn -Sector fuer Sector- eingelesen
1610 PRINT:PRINT"Jetzt lade ich:
1620 bufflow=VAL("&" +RIGHT$(buffer$,2)):buffhigh=VAL("&" +LEFT$(buffer$,2))
1630 '
1640 '*****
1650 '* String wieder zerlegen und Einlesevorgang *
1660 '*****
1670 FOR i = 1 TO LEN(ges$) STEP 4
1680 g$=MID$(ges$,i,4):PRINT g$,
1690 track=VAL("&" +LEFT$(g$,2)):sector=VAL("&" +RIGHT$(g$,2))
1700 GOSUB 1800:buffhigh=buffhigh+2
1710 NEXT i:PRINT:PRINT
```



```

1720 '
1730 PRINT"Ihr Programm steht nun von "buffer$;
1740 PRINT" bis "HEX$(buffhigh-1)+"FF im Speicher !
1750 END
1760 '
1770 '*****
1780 '* Werte fuer Einlesevorgang uebergeben und Maschinenroutine aufrufen *
1790 '*****
1800 secform =form+sector:POKE (buffermc),bufflow:POKE (buffermc+1),buffhigh
1810 POKE drivenmc,drive:POKE trackmc,track:POKE formsemc,secform:CALL mcstart
1820 RETURN

```

Listing: READSEC1

```

100 '* READSEC1 *
110 '
120 MODE 2:MEMORY &8FFF:PRINT"Programm: READSEC1
130 '
140 'Adressen fuer Maschinenprogramm
150 akt drivemc=&A702:befehlmc=&A622:buffermc=&A629:drivenmc=&A626
160 formsemc=&A628:trackmc=&A627:mcstart=&A600
170 '
180 buffer$="9000":'           Beginn des Einlesebereiches
190 form=&C0:'                 Code fuer Format
200 '
210 LOAD"sector.bin",&A600:'   Maschinencode einlesen
220 '
230 PRINT"Bitte die entsprechende Disk einlegen und Taste druecken"
240 CALL &BB06
250 '
260 'Werte fuer Bufferadresse berechnen
270 bufflow=VAL("&"+RIGHT$(buffer$,2)):buffhigh=VAL("&"+LEFT$(buffer$,2))
280 '
290 sector = sector +1:'       Sektor 0 gibt es nicht !
300 GOSUB 440:'               1. Sector einlesen
310 '
320 CLS:PRINT STRING$(80,"-"): ' Schirm loeschen + Strich ausgeben
330 beginn=&9000:'           Start fuer Auslesen festlegen
340 '
350 'Ausleseschleife
360 FOR i = 0 TO 15:adr = i+beginn:wert=PEEK(adr): PRINT HEX$(wert,2);" ";
370 IF wert <&20 THEN wert=32
380 asci$=asci$+CHR$(wert):NEXT:PRINT" ";asci$:asci$=""
390 z=z+16:IF z=32 THEN PRINT STRING$(80,"-"):z=0
400 IF adr < &9000+511 THEN beginn=beginn+16:GOTO 360 ELSE END
410 '*****
420 '* Werte fuer Einlesevorgang uebergeben und Maschinenroutine aufrufen *
430 '*****
440 secform =form+sector:POKE (buffermc),bufflow:POKE (buffermc+1),buffhigh
450 POKE trackmc,track:POKE formsemc,secform:CALL mcstart:RETURN

```


Arbeiten mit Diskette (4)

Wieder bauen wir auf die schon erworbenen Kenntnisse auf und schreiben nun ein Programm, über dessen Sinn Sie sich wahrscheinlich erst dann, wenn Sie es einsetzen müssen, richtig freuen können. Ich habe dem Programm den Namen **FRETTER** (Abkürzung von **Fileretter**) gegeben, da er am deutlichsten zeigt, was mit dem Programm erreicht werden kann. Sie können damit versehentlich gelöschte Files wieder zurückholen, also retten. Voraussetzung ist allerdings, daß Sie nach dem Löschvorgang noch keine weiteren Files auf diese Diskette geschrieben haben, denn sonst ist die Rettung fraglich.

Schon in unserem ersten Heft haben wir ein Programm veröffentlicht, das diesen Zwecken dient. Hierbei durften die Files (Programme) aber nicht länger als 16 KB sein. Außerdem kann dieses Programm nur unter CP/M eingesetzt werden. Mit dem zu diesem Artikel gehörenden Programm ist diese Einschränkung nicht mehr gegeben, denn jeder Fileeintrag kann direkt bearbeitet werden.

Ein Pladoyer für BASIC

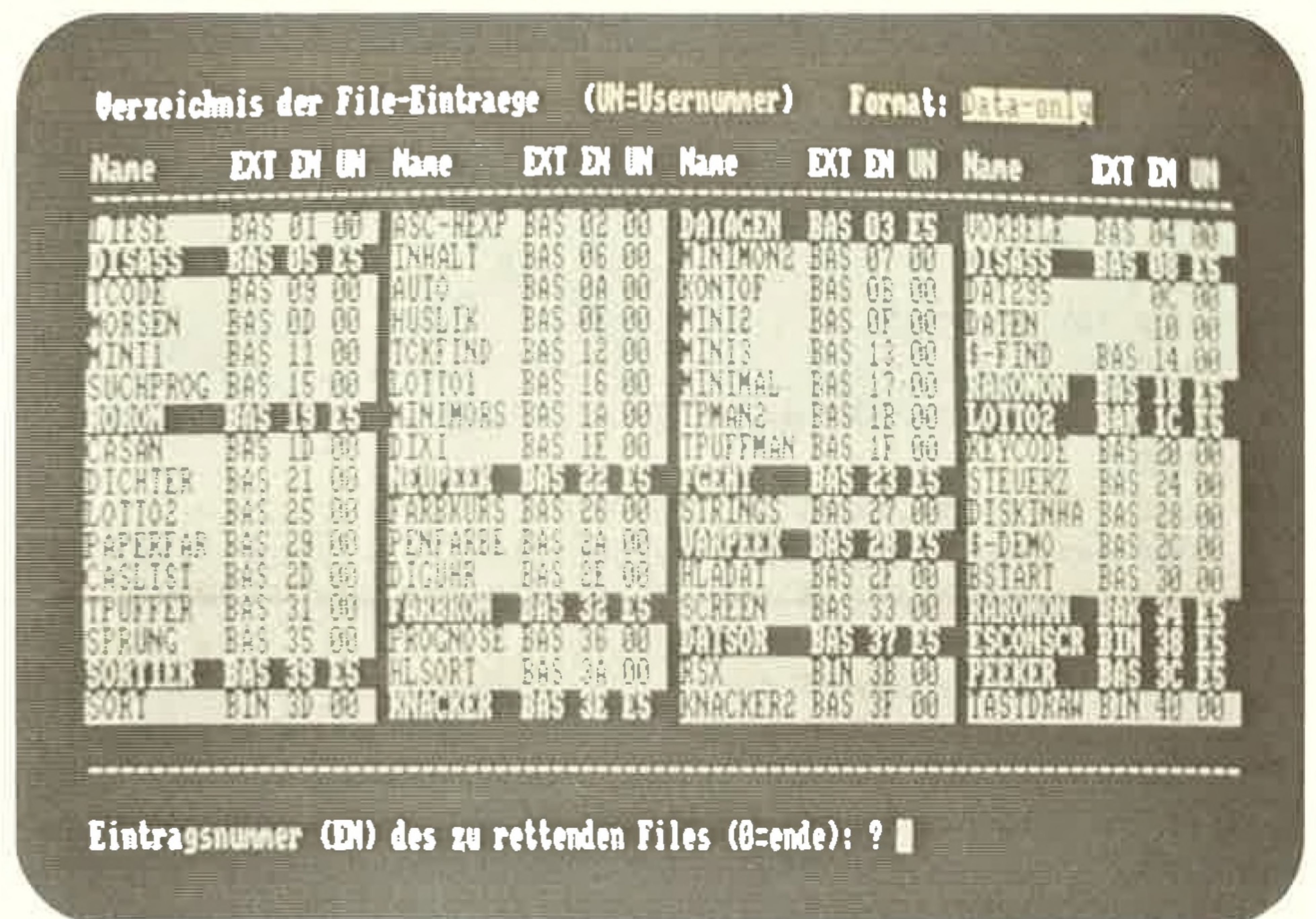
Außerdem wird mit diesem Programm (wie auch mit vielen anderen dieses Heftes) ganz klar aufgezeigt, was mit Basic, unterstützt durch Maschinensprache, möglich ist.

Mir darf niemand erzählen, daß Basic eine veraltete oder schlechte Programmiersprache sei. Egal ob „Turbo-Fortran“, „Quick C“, „Maxi-PL“, „XZ-Pascal“, „Forth XX“ oder wie auch immer eine Hochsprache heißt, derjenige, der seinen Rechner kennt und Maschinensprache beherrscht, programmiert jeden „Hochsprachenprogrammierer“ auch mit Basic in „Grund und Boden“, wenn er Maschinensprachemodule zur Basicunterstützung ein-

setzt. Dies soll Sie aber nicht davon abhalten, auch andere Programmiersprachen zu erlernen und einzusetzen, denn auch diese haben ihre Vorteile und Vorzüge.

Daß aber von „Sprachumsteigern“ immer mitleidig auf die etwas „zu-

Genug der „Frotzelei“ und Spaß beiseite, wer andere, weil sie nur in der einen oder anderen Programmiersprache programmieren können, nur schiefen Blickes würdigt, zeigt nur, daß er eine gehörige Portion an Intoleranz besitzt, aber nicht mehr.



rückgebliebenen“ Basicprogrammierer herabgesehen wird, das sollte eigentlich nicht sein! Schließlich war es vor allem Basic, das zu dieser enormen Verbreitung der Computer (in privaten Händen) geführt hat. Und nur dadurch war der Sieges- und Einzug der Computer in fast jeden Haushalt möglich.

Ich jedenfalls programmiere (neben anderen Hochsprachen) sehr gerne in Basic, vor allem weil es sehr schnell geht und trotz des verrufenen Spaghetticodes übersichtlich ist, wenn sich der Programmierer etwas Mühe gibt. Diejenigen die anderer Meinung sind, bitte ich um Tolerierung meiner Ansicht.

Beinahe hätte ich doch glatt vergessen, zu erwähnen, daß bei einer der modernen Hochsprachen, nämlich bei Dr. LOGO unter CP/M 2.2, glatt vergessen wurde, eine Druckerausgabe einzubauen.

Zurück zum Thema. Mittels des bereits bekannten (also unseres erarbeiteten) Maschinensprachemoduls und einem weiteren Basicteil haben Sie die Möglichkeit, jedes Ihrer versehentlich gelöschten Diskettenfiles wieder für gültig zu deklarieren.

Änderung der User-Nummer = Löschung?

Dies liegt ganz einfach daran, daß bei einer „Löschung“ durch ERA nicht wirklich gelöscht wird. Das einzige, was dabei verändert wird, ist die User-Nummer im Fileeintrag! Wie Ihnen ja bereits bekannt ist, können die User-Nummern zwischen 0 und 15 liegen. Bei Löschung eines Files geschieht nun nichts anderes als die Änderung dieser Nummer in &E5. Wird diese

Username wieder auf einen „gültigen“ Wert gesetzt, ist das File wieder da! Was geschieht, wenn Sie diesen Wert oberhalb 15 aber nicht mit &E5 belegen, können Sie durch eigene Experimente feststellen. Ich habe bisher nur feststellen können, daß dies äußerst interessante Aspekte eröffnet. Werden also die zusammengehörenden Fileeinträge auf eine „gültige“ Username gesetzt, dann ist eine Löschung rückgängig gemacht. Das Programm selbst ist so übersichtlich in der Darstellungsweise, daß sich größere Erklärungen hierzu erübrigen. Auf einen kleinen Trick in diesem Programm will ich Sie aber doch aufmerksam machen. Die CPCs haben eine Firmware-Routine, die es ermöglicht invertierte Zeichendarstellungen zu bewirken. Diese Routine steht in Zeile 1270 und wird zur Anzeige der gültigen Files benutzt. Interessant ist für Sie aber auch, daß

Sie bei Ihren bisher gekauften Programmdisketten Files sehen können, die Sie bisher nicht sahen, bzw. die gegen Löschen geschützt waren. Dies kann außer an einer manipulierten User-Nummer auch an gesetzten 7. Bits im File-Namen liegen. Gleiches gilt auch bei gegen Löschen geschützten Files. Sehen Sie sich dies ruhig etwas genauer an. Aber ändern Sie nichts an derartigen Originaldisketten, denn der Programmverlust kann die Folge sein! Daß Sie mit dem Programm FRETTER auch Files löschen können, dürfte auch klar sein, denn Sie brauchen die Username nur auf &E5 setzen, dann ist es schon passiert. Die Bedienung des Programmes selbst ist so einfach, daß sich dies aus dem Programmlauf selbst ergibt. Spielen Sie aber anfangs nicht unbedingt mit Ihren wertvollsten Disketten. Ein kleiner Fehler beim Abtippen des Programmes

kann unangenehme Folgen haben. Nehmen Sie gerade beim „Üben“ eine wirkliche Übungsdiskette.

Wichtig für 6128-Besitzer

Zum Schluß noch ein Hinweis für CPC-6128-Besitzer, die außerdem noch den MAXAM angesteckt haben. Bei mir gab es nur bei diesem Programm sporadisch auftretende Fehler bei einem zweiten Programmlauf. Da dieser Fehler nur sporadisch auftrat, konnte ich den Grund hierfür noch nicht finden. Der Fehler machte sich insofern bemerkbar, daß nach Rückkehr aus dem Maschinenprogramm ein Syntax-Error gemeldet wurde. Die ROM-Select-Nummer hatte nach dieser Fehlermeldung den Wert 10. Am 464 konnte ich diesen „komischen“ Effekt nicht feststellen.

Lothar Miedel

Listing: Fretter - Ein Programm zur Filerettung

```

100 '*****
110 '* FRETTER = Disk-File-Retter fuer die CPC's (c) by LM *
120 '*****
130 '
140 MODE 2:MEMORY &9DFF:GOSUB 1270:PRINT"Programm: FILE-Retter":GOSUB 1270
150 '
160 'Adressen und Werte fuer Maschinenprogramm
170 aktdrivemc=&A702:befehlmc=&A622:buffermc=&A629:drivenmc=&A626
180 formsemc=&A628:trackmc=&A627:befehl=&84:mcstart=&A600
190 '
200 'Spaeter benoetigte Variable festlegen
210 mi$=" -":pl$=" +":tr$=STRING$(79,"-")
220 '
230 'Maschinenprogramm einlesen falls Pruefbyte nicht ok!
240 adr=mcstart
250 c=PEEK(adr):'Pruefbyte holen (Maschinenprg vorhanden ?)
260 IF c=58 THEN 300:'Sprung da Pruefbyte stimmt
270 LOAD"sector.bin",&A600
280 '
290 'Beginn des Bufferbereiches
300 buffer$="9E00"
310 '
320 PRINT:PRINT"Bitte entsprechende Diskette einlegen und Taste druecken"
330 CALL &BB06

```



```

340 '
350 OPENOUT "dummy":CLOSEOUT:CLS:' pruefen, welches Diskformat
360 drive=PEEK(akt drivemc)
370 IF drive = 0 THEN form =PEEK(&A89F)-1:'Diskformat abholen und merken
380 IF drive = 1 THEN form =PEEK(&A8DF)-1:'Diskformat abholen und merken
390 IF form =&C0 THEN offset=0:fo$="Data-only"
400 IF form =&40 THEN offset=2:fo$="CP/M oder Vendor
410 '
420 '*****
430 '* eigentlicher Start und Wiedereinsprung *
440 '*****
450 'Werte fuer Bufferadresse berechnen
460 bufflow=VAL("&" +RIGHT$(buffer$,2)):buffhigh=VAL("&" +LEFT$(buffer$,2))
470 sector=0:IF flag=1 THEN befehl=&85:flag=0
480 '
490 '*****
500 '* Komplettes Inhaltsverzeichnis einlesen oder schreiben *
510 '*****
520 sector = sector +1
530 track=offset
540 GOSUB 1210:' Sector einlesen oder schreiben
550 '
560 ' Falls sector = 4, dann ist das Inhaltsverzeichnis komplett
570 IF sector=4 THEN 650
580 '
590 buffhigh=buffhigh+2:GOTO 520
600 '
610 '*****
620 '* Nun Auswertung und Ausgabe *
630 '*****
640 '
650 lesezeiger =VAL("&" +buffer$):CLS
660 PRINT"Verzeichnis der File-Eintraege (UN=Usernummer) ";
670 PRINT" Format: ";GOSUB 1270:PRINT fo$:GOSUB 1270:PRINT
680 PRINT"Name EXT EN UN Name EXT EN UN Name EXT EN UN";
690 PRINT" Name EXT EN UN":PRINT tr$
700 blcnt=0:filename$=""
710 usernummer=PEEK(lesezeiger)
720 IF usernummer=&E5 THEN file$=mi$:' File geloescht
730 IF usernummer<&16 THEN file$=pl$:' File nicht geloescht
740 '
750 ' File-Namen holen
760 FOR i = lesezeiger+1 TO lesezeiger+12
770 z$=CHR$(PEEK(i))
780 filename$=filename$+z$
790 NEXT i
800 lesezeiger =lesezeiger+32
810 '
820 'Pruefung ob letzter File-Eintrag
830 IF LEFT$(filename$,1)=CHR$(&E5) THEN fz=65:GOTO 910
840 fz=fz+1

```



```

850 IF file$=pl$ THEN GOSUB 1270:' File gueltig deshalb Text invertieren
860 PRINT LEFT$(filename$,8);" ";;PRINT RIGHT$(filename$,4);
870 PRINT " "HEX$(fz,2);:PRINT " "HEX$(usernummer,2)" ";
880 IF file$=pl$ THEN GOSUB 1270:' File gueltig deshalb Text invertieren
890 PRINT" ";;filename$=""
900 '
910 IF fz<64 THEN 710: 'noch keine 64 Directory-Eintraege
920 PRINT:PRINT tr$
930 '
940 '*****
950 '* Alle Eintraege ausgegeben *
960 '*****
970 '
980 PRINT:INPUT"Eintragsnummer (EN) des zu rettenden Files (0=ende): ";f$
990 f$=LOWER$(f$):IF f$="" THEN END
1000 f$="&"+f$:fz=VAL(f$):filename$=""
1010 '
1020 lesezeiger=VAL("&"+buffer$)+(fz-1)*32
1030 '
1040 usernummer=PEEK(lesezeiger): merker =lesezeiger
1050 '
1060 'Namen holen
1070 FOR i = lesezeiger+1 TO lesezeiger+12
1080 z$=CHR$(PEEK(i))
1090 IF ASC(z$)>128 THEN z$=CHR$(ASC(z$)-64):GOTO 1090:' Gesetzte Bits raus
1100 filename$=filename$+z$
1110 NEXT i
1120 '
1130 CLS:PRINT"Gerettet werden soll : "filename$;
1140 PRINT"Eintrags-Nr.: "fz:PRINT
1150 INPUT"Usernummer";usernummer
1160 POKE merker,usernummer
1170 fz=0
1180 flag=1:GOTO 450
1190 '
1200 '*****
1210 secform =form+sector:POKE buffermc,bufflow:POKE (buffermc+1),buffhigh
1220 POKE drivenmc,drive:POKE trackmc,track:POKE formsemc,secform
1230 POKE befehlmc,befehl:CALL mcstart: RETURN
1240 '
1250 '*****
1260 ' Schrift invertieren
1270 CALL &BB9C:RETURN

```


Arbeiten mit Diskette (5)

In diesem Teil der Grundlagenartikel über die Floppy gehe ich auf ein besonders heikles Thema ein. Nämlich auf das Thema Programmschutz. Viele Hersteller von Software versuchen, ihre Produkte vor dem Einblick fremder Augen zu schützen. Die Gründe hierfür sind mir bestens bekannt.

Auch ich habe eine nicht gerade gute Meinung von Personen, die mit der Arbeit anderer versuchen, Geschäfte zu machen. Was ist aber mit demjenigen, der sein Programm mit dem CPC-Schutz versehen hat und vergaß, eine offene Kopie abzuspeichern? Diesen CPC-Besitzern soll mit dem zu diesem Artikel gehörenden Programm geholfen werden. Aber vor allem auch andere Punkte bezüglich Softwareschutz und Copyrights werden angesprochen. Ein wirklich „heißes“ Thema.

Und deshalb zählen auch solche Informationen zu den Grundlagen.

Einmal abgesehen von der Computerei, zeigt die Geschichte der Evolution, daß nur durch Kopie guter Eigenschaften ein Weiter- oder Fortbestand gesichert ist. All dies trifft eigentlich für alle Lebensbereiche zu. Es wäre ja auch unsinnig, jedesmal das „Rad“ neu erfinden zu wollen. Andererseits aber ist es sicherlich das gute Recht der Hersteller und Ersteller von Produkten, für die geleistete „Arbeit“ auch gebührend entlohnt zu werden. Dies gilt auch für die Hersteller von Soft- und Hardware.

Eine unbefriedigende Situation

Durch Copyrights wird versucht, Mitbewerbern den Weg zu fremden „Früchten“ zu verwehren. In manchen Fällen aber nimmt dies Formen an, die leider zum Nachteil der eigentlichen Kunden ist.

Wenn es so weitergeht, wie es im Augenblick aussieht, dann unterlie-

gen irgendwann sogar einzelne simple Programmroutinen einem Copyright, weil ein guter Sachverständiger in diesen irgendwelche außergewöhnliche geistige Leistung „entdeckt“. Kopiert wurde immer und wird es immer werden. Im Falle der Computer eben manchmal auch zum Schaden der Softwarehäuser. Andererseits aber gibt es das schon fast klassische Beispiel des Programmes WORDSTAR, dem wohl meistkopierten Anwenderprogramm für Mikrocomputer. Trotz aller Kopien, war und ist der Verkauf dieses Programmes ein „Bombengeschäft“.

Deswegen gleich einmal ein paar Takte zum Thema „Raubkopierer“. Ich persönlich zähle zu den Raubkopierern eigentlich nur diejenigen, die dies tun um Software an denen sie keine Rechte besitzen, weiterzuverbreiten. Egal ob mit oder ohne Gewinn. Solange jemand – für sich – nur „Arbeitskopien“ von Programmen macht, die er legal erworben hat, sehe ich eigentlich keine strafbare Handlung. Die rechtliche Situation ist aber viel komplizierter! An ein paar Beispielen will ich erläutern, was ich damit meine und weshalb Sie sich – zu Ihrem eigenen Schutz – auf dem neuesten Stand der Dinge halten sollten. Da die Fachpresse laufend hierüber berichtet, haben Sie die Möglichkeit hierzu.

Lesen Sie weiter, welche Auswüchse die ganze Copyrightsituation „treibt“.

Manche Softwarehersteller erlauben nur eine Arbeitskopie. Andere gehen soweit, eine Arbeitskopie durch Schutzmaßnahmen, die nicht jeder umgehen kann, überhaupt nicht zuzulassen. Schlimm finde ich auch, daß in manchen Verkaufsbedingungen steht, daß das erworbene Programm nur auf einem einzigen Rechner eingesetzt werden darf. Stellen Sie sich bitte einmal vor, Sie kaufen eine Schallplatte und diese dürfen Sie nur auf dem Plattenspieler abspielen, den Sie

zum Zeitpunkt des Kaufes besaßen. Falls Sie zwei Plattenspieler besitzen, was ist dann? Vielleicht haben Sie sogar in Ihrem Wochenend-Dominizil einen weiteren Plattenspieler und dort dürfen Sie sich Ihren Lieblingssong ebenfalls nicht anhören! Zurück zur Software. Was bedeutet dies eigentlich in dem Falle, daß Sie sich zu Ihrem CPC 464 auch noch einen CPC 6128 zulegen? Dürfen Sie dann...? Oder Sie wollen einem Bekannten, der sich wirklich nur informieren will, die Möglichkeit geben, sich einmal länger mit einem von ihm zum Kauf geplanten Programm zu beschäftigen? Das sind Punkte, die durch die Rechtsprechung nicht voll geklärt sind. Ich weiß, daß die Hintergründe für derartige „Bestimmungsblüten“ schon einen tieferen Sinn haben. Es soll vor allem sichergestellt werden, daß große Firmen, die mehrere Computersysteme haben, auch für jedes System ein eigenes Programm kaufen sollen. Aber es ist oft noch schlimmer, denn manchmal können Sie auch lesen, daß Sie nicht das Programm, sondern nur die Nutzungsrechte zum Betrieb auf einem Computer erworben haben. Können Sie nun diese Nutzungsrechte verschenken, weil Sie dieses Programm nicht mehr verwenden wollen?

Wie sieht es aus, wenn Sie Ihren „alten“ Computer durch einen „Neuen“ ersetzen, dürfen Sie diesen Nachfolger mit den alten Programmen beglücken? Fragen über Fragen und in jedem Falle wäre – beim Auftreten eines Klägers – eine eventuelle richterliche Entscheidung wahrscheinlich anders. Geht das nicht alles etwas zu weit? Sind hier nicht manche Bestimmungen völlig unsinnig? Betrachten wir das Ganze auch noch einmal von einer anderen Seite. Sie haben sich ein Programm gekauft, mit dem Sie Ihre Buchhaltungsaufgaben erledigen. „Fairerweise“ wurde Ihnen erlaubt, eine Arbeitskopie zu erstellen. Durch einen unglücklichen Zufall

wurde die Arbeitskopie unbrauchbar. Da Sie aber das Original besitzen, kein Problem. Original aus dem feuer-, einbruchs- und erdbebensicheren Panzerschrank geholt, mit einem Doppellaufwerk die Erstellung einer neuen „Arbeitskopie“ eingeleitet. Wenn Sie nun Pech haben, kann Ihnen dasselbe passieren, wie ich es schon erlebt habe: Stromausfall! Dies muß nun nicht heißen, daß das Original zerstört wurde, aber ich habe es leider schon zweimal persönlich erlebt. Beide Lese-/Schreibköpfe waren im „Diskettenzugriff“, durch den Stromausfall ging die Elektronik in einen undefinierten Zustand. Der Erfolg, es wurde kurzfristig auf beide Disketten geschrieben. Original und die halbfertige Kopie waren unbrauchbar. Da es sich bei mir um eigene Datendisketten handelte, hatte ich nach dem Vater/Sohn/Enkel-Prinzip gearbeitet. Deswegen war es nicht so tragisch. Als der Strom wieder da war und ich festgestellt hatte, daß beide Disketten unbrauchbar waren, nahm ich die „Sohn“-Diskette. Den Kopiervorgang wieder gestartet..., und nun dürfen Sie schmunzeln... Seitdem bleiben meine Computer an gewittrigen Sommerabenden ausgeschaltet. Anmerken möchte ich aber noch, daß es kein CPC war, mit dem ich dieses Erlebnis hatte. Falls Ihnen aber ein derartiges Mißgeschick mit Programmdisketten passiert, müssen Sie nun, nur weil mehrere Kopien nicht erlaubt waren, das gleiche Programm nochmals kaufen? Ich hoffe nur, daß in der nächsten Zukunft eine Lösung im Sinne der Programm-Kunden gefunden wird. Vielleicht kann dies auf ähnliche Weise wie bei den Musikträgern geschehen (GEMA-Gebühren o. ä.). Bis dies aber soweit ist, kann ich Ihnen nur empfehlen, passen Sie auf, daß Sie nicht durch Verstoß gegen Vorschriften oder Bestimmungen in die Mühlen der Justiz geraten.

Auch ich muß nun aufpassen mit dem Programm PCOPY, denn es gibt Abmahnfirmen, die sich darauf spezialisiert haben, zum Beispiel gegen Anbieter von Backupsoftware vorzugehen, nur weil diese vergessen haben, in der Anzeige darauf hinzuweisen, daß das angebotene

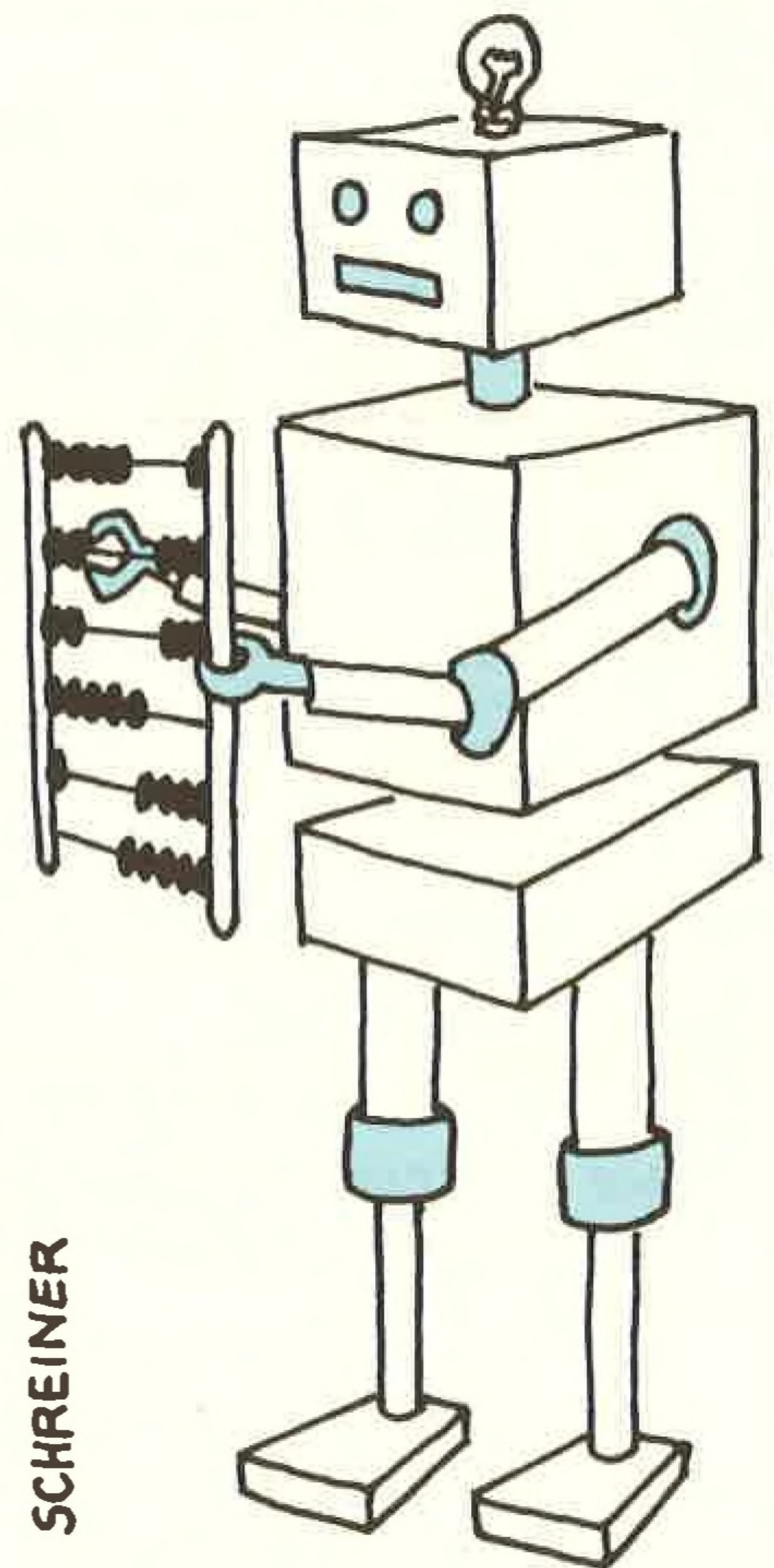
Programm nur für Backups von Software erlaubt ist, deren Kopierrechte man besitzt. „Beihilfe...“. Oft kommt es zwar nicht darüber dann zu einem Rechtsstreit, aber es sind seitens der Abmahnfirma Kosten angefallen, über die dann vielleicht ein Richter entscheiden muß. Deshalb weise ich nun ausdrücklich darauf hin, daß das Programm „PCOPY“ dieses Heftes nur zum Kopieren von Software benutzt werden darf, deren Rechte man besitzt.

Save„name“.p – ist dies sinnvoll

Doch nun wieder zum CPC. Was die Entwickler der CPC-Firmware dazu bewogen hat, die Möglichkeit vorzusehen, ein Basic-Programm zu „schützen“ wird mir vermutlich immer ein Rätsel bleiben. Vergessen diese in höchsten Regionen schwebenden „Koryphäen“ tatsächlich, daß alles was für ein Programm erforderlich ist, auch durch irgendwelche Programmteile lesbar sein muß? Oder glauben diese, daß sie soweit oberhalb des allgemeinen Wissens schweben, daß andere, die sich mit Computern beschäftigen nicht in der Lage sind innerhalb kürzester Zeit solche Schutzmechanismen zu beseitigen? Ich weiß es nicht, eines aber ist mir klar, die Programmierer des „protect“-Schutzes von Locomotive Software hatten sicherlich nicht bedacht, wie einfach beim CPC 464 dieser Schutz umgangen werden kann. Mein Aufruf deswegen an Programmierer von Firmware, laßt diesen Unsinn und implementiert dafür lieber eine Hardcopy-Routine. Gute Software braucht nicht geschützt zu werden, denn diese wird in ausreichenden Mengen gekauft. Ich erinnere nur an Wordstar. Auch gut gemachte Handbücher nützen mehr als solche Spielereien. Der einzige Effekt, der wirklich mit einer Schutz Einrichtung dieser Art erreicht wird, ist der, daß ein Anwender sich vor sich selbst schützt und irgendwann ein eigenes Programm vor sich selbst „versteckt“. Professionelle Software wird meist mit anderen Methoden geschützt.

Nun wieder zum Sinne des Programmes PCOPY. Auch mir ist es

schon passiert, daß ich bei Spielereien ein Programm mit dem Zusatz „P“ geschützt habe, ohne vorher eine offene Version abzuspeichern. Wollte ich nachträglich etwas am Programm ändern, gab es eigentlich keine „normale“ Möglichkeit mehr, dies zu tun. Nun ich habe ausreichende Kenntnisse um mir dann zu helfen. Was aber ist mit dem „armen“ Gelegenheitsprogrammierer, der den gleichen Fehler gemacht hat? Für den CPC 464 gibt es den wohl schon allgemein bekannten sogenannten „Readypatch“, der die Tatsache ausnutzt, daß der CPC 464 ab und zu über sogenannte Indirections arbeitet. Bei den Nachfolgern ist dies nicht ganz so leicht möglich.



SCHREINER

Aber auch diese Geräte haben diesen Unsinn eingebaut. Deshalb also das Programm PCOPY. Allerdings habe ich das Programm so aufgebaut, daß die „unprotected“ Version auf die „Originaldiskette“ zurückgeschrieben wird. Eine Maßnahme, die ich für sinnvoll hielt. Für weiterhin sinnvoll hielt ich es, das Programm

nur als Basicloader abzdrukken und kein dokumentiertes Assemblerlisting mitzuliefern. Bitte haben Sie Verständnis dafür.

Zur Bedienung gibt es nicht viel zu sagen. Sie brauchen nur das Programm zu laden und zu starten, dann führt es Sie selbst weiter. Er-

schrecken Sie aber nicht, wenn der CPC nach Erledigung seiner Arbeit plötzlich einen Reset durchführt. Geben Sie nach dem Reset CAT ein, dann sehen Sie, daß das geschützte Programm mit dem Extent .bak und das neue Programm mit .bas im Inhaltsverzeichnis steht. Ich habe al-

lerdings noch eine kleine „Sicherheitsmaßnahme“ eingebaut. Es können nur geschützte Basicprogramme mit dem Extent BAS bearbeitet werden.

Sie können es trotzdem mit anderen Extends versuchen, aber ich garantiere für nichts. Lothar Miedel

Listing: PCOPY

```

100 ' *****
110 ' *                               P C O P Y                               *
120 ' * Ein Programm um versehentlich mit 'p' abgespeicherte Programme *
130 ' * wieder als normale Basicprogramme auf Disc zurueckzuschreiben. *
140 ' *****
150 '
160 MODE 2:MEMORY &9FFF:a=&A000:e=&A03D:zb=1000:e=e+1
170 FOR i =a TO e:READ d$:IF LEFT$(d$,1)="/" THEN flag =1
180 IF (flag AND ps<>VAL(d$)) THEN PRINT"Fehler in Zeile "zb+1:END
190 IF (flag AND i=e) THEN 240
200 IF flag THEN i=i-1:zb=zb+1:ps=0:d$="":flag = 0:GOTO 220
210 d$="/" +d$:POKE i,VAL(d$):ps=ps+VAL(d$):
220 IF i < e THEN NEXT i
230 '
240 CLS:PRINT"PCOPY wurde korrekt geladen !":PRINT:PRINT
250 PRINT"Bitte immer daran denken, dass Sie nur Software kopieren duerfen,
260 PRINT"wenn Sie auch die Rechte hierzu besitzen !":PRINT
270 PRINT"Falls dies der Fall ist, geben Sie nun bitte
280 INPUT"den Filenamen (ohne Extend!!) ein ";filename$
290 IF LEN (filename$)>8 THEN PRINT:PRINT CHR$(7);"Fehler":PRINT:GOTO 270
300 IF LEN(filename$) < 8 THEN filename$=filename$+CHR$(32):GOTO 300
310 '
320 FOR i = 1 TO 8:POKE &A016+i,ASC(MID$(filename$,i,1)):NEXT
330 '
340 PRINT:PRINT"Bitte legen Sie nun die entsprechende Diskette ein !
350 PRINT"und druecken Sie eine Taste !":CALL &BB06:CALL &A000
360 '
1001 DATA 06,0C,11,00,C0,21,17,A0,CD,77,BC,21,70,01,CD,83,&059D
1002 DATA BC,CD,7A,BC,C3,23,A0,66,72,65,74,74,65,72,20,2E,&078F
1003 DATA 62,61,73,06,0C,21,17,A0,11,00,C0,CD,8C,BC,21,70,&0597
1004 DATA 01,ED,5B,6D,A7,3E,00,CD,98,BC,CD,8F,BC,C7,&079B

```


Die Gewinner unserer ComputerSchau-Preisausschreiben!

Verlosungsaktion „1 Jahr ComputerSchau“!

Taschencomputer PC 1500A
von Sharp:

Siegfried Baller
bei Zoeh
Ringslebenstraße 2
1000 Berlin 47

Kurt Dankwart
Reinhold Klügel Hof Nr. 37
5140 Erkelenz

Horst Werner Alfeis
Farnweg 1
3002 Wedemark

J. Bausewein
Bramsallee 39/IV
2000 Hamburg 13

Jochen Baumgärtner
Hindenburgstraße 24
8520 Erlangen

Wolfgang Domres
Heckenweg 25
3006 Burgwedel

Dieter Kasper
Dipl.-Ing.
Robert-Koch-Straße 36
2800 Bremen 61

Manfred Kretzschmar
Dipl.-Ing.
Sudewiesenstraße 50
3014 Laatzen

Eugen F. Breuninger
Karwendelstraße 5
8038 Gröbenzell

Joachim Baatz
Burkhardtstraße 4
8501 Feucht

Rolf Bräuker
Am Kindergarten 29
5840 Schwerte

Hans Bernhard
Langerstraße 2/I
8000 München 80

V. Süß
Wiederholdstraße 22
7000 Stuttgart 1

Friedhelm Will
Bahnstraße 30
4174 Issum 1

Oliver Volz
Dusestraße 13
7000 Stuttgart 80

Norbert Hettich
Gauchachstraße 14
7715 Bräunlingen

Ralf Schulze
II. Parallelstraße 9
4630 Bochum 1

Franz Jaquet
Am Buchet 10
8035 Gauting

Joachim Pöhlmann
Meisenweg 37
8526 Bubenreuth

Manfred Wilimek
Haimhauser Straße 7
8000 München 40

ComputerSchau 9/85

1. Preis: 1 Schachcomputer
der Fa. Hegener & Glaser AG

Roland Kehlenbach
Scherpenberger Straße 19
4130 Moers 1

Franzis-Buchgutscheine:

Peter Meusel
Alte Marburger Straße 5
6348 Herborn

Gerd Nolden
Albertstraße 90
4000 Düsseldorf 1

Bruno Marti
Neubrückstraße 28
CH-2555 Brugg (BE)

Günter Fischer
Feldstraße 13
2081 Alveslohe

Gerd Heydt
Alter Stadtweg 47
6602 Sbr.-Dudweiler

Klaus Sperling
Eversbuschstraße 188
8000 München 50

Wolfram Richter
Zuckerberg 51
5000 Köln 50

Helmut Ramstötter
Stutenanger 10
8042 Oberschleißheim

Gabriele Szklenar
Rathausstraße 28
7062 Rudersberg

Stefanie Heibel
Monumentenstraße 37
1000 Berlin 62

ComputerSchau 10/85

1. Preis: 1 Radiorecorder
der Fa. Sharp

Manfred Michels
Knorrstraße 45
8000 München

Franzis-Buchgutscheine:

Peter Hufnagl
Hospitalstraße 9
6086 Riedstadt 1

Christoph Windmüller
Uhlandstraße 96
4500 Osnabrück

Astley Andersson
Mattiasg 12B/I
S-81500 Tierp

Gerd Sommer
Kerner Straße 4
7120 Bietigheim-Bissingen

Willi Blum
Bohlstraße 9
7430 Metzingen

Max Bauer
Adlerstraße 86
8300 Landshut

Uwe Petsch
Billhorner Deich 50
2000 Hamburg 28

Axel Hengemühle
Erdelhofstraße 17
4600 Dortmund 15

Heinz Weigelt
Proschwitzer Straße 12
8950 Kaufbeuren 2

Christoph Nebel
Erlenweg 7
2358 Kaltenkirchen

Mit einem Sony Hit Bit kann man ein Zum Beispiel große Literatur und kle

Von Haus aus beste Daten.

Jetzt ist die Gelegenheit besonders günstig, die Computerwelt kennenzulernen und dabei verborgene Talente zu fördern. Mit dem Sony Hit Bit kriegen Ihre Texte plötzlich ganz einfach Format. Und farbige Grafiken und Tabellen entstehen wie von selbst. Der Sony Hit Bit bringt alles mit, was man dazu braucht.

HIT BIT

Denn er funktioniert nach dem international festgelegten MSX-Standard. Das bedeutet Zugang zu allen Programmen und Zusatzgeräten wie Druckern oder Datenrecordern gleichen Standards. Und eine beruhigende Zukunftssicherheit für Ihre Kaufentscheidung. Was der Sony Hit Bit sonst noch alles auf dem Kasten hat, ganz kurz für die Kenner der Materie: Der HB-75 D ist ein Z 80 A-Computer mit deutscher Schreibmaschinen-Tastatur. Sein 64 K-Byte Arbeitsspeicher erlaubt auch die Verwendung anderer Betriebssysteme wie z. B. MSX-Dos.

Für spezielle ROM-gestützte MSX-Programme kann die gesamte Speichertiefe von 64 K zur Aufnahme von Daten genutzt werden. Zusätzlich bietet der Sony Hit Bit einen 16 K-Byte großen Bildspeicher. Ein integriertes deutsches Dateiprogramm, das spielend leicht Adressenlisten und Terminplanungen erstellen und verwalten hilft. Anwenderfreundliche Sortier-, Such-, Transfer- und Druckroutinen. Und einen 32 K-Byte MSX-Basic-Interpreter, der das Erstellen eigener Programme mit Grafik und Sound auch ohne Kenntnisse des Betriebssystems ermöglicht. Grafiken gibt der Hit Bit in 256 x 192 Bildpunkten und 16 verschiedenen Farben wieder. Der eingebaute Tongenerator mit 3 Tonausgängen und 1 Geräuscheffektgenerator hat einen Tonumfang von 8 Oktaven. Eine 21polige RGB-Scartbuchse und eine 6polige DIN-AV-Buchse garantieren vielseitigen und besten Anschluß an Fernseher und Monitor. Schnitt-



stellen für Audio-Cassetten-Recorder und Joystick, zwei MSX-Standard I/O-Interfaces und ein Druckerinterface sorgen für problemlosen Anschluß an MSX-Peripheriegeräte. Das Hit Bit Zubehör umfaßt außer wichtigen Kabeln ein ausführliches MSX-Basic Programmierhandbuch, eine Einführung in MSX-Basic und Anleitungen zur Erstellung von Personenkarteien.

Der Bitcorder SDC-500.

Das kennen auch Computerneulinge. Eine ganz normale Audio-Compact-Cassette. Zusammen mit dem SDC-500 Bitcorder ergibt das einen von 3 Datenspeichern, die dem Sony Hit Bit als Langzeitgedächtnis dienen können. Der Bitcorder speichert große Datenmengen, ist besonders preiswert und leicht zu bedienen.



Menge machen: e Kunstwerke.



Die Sony ROM-Cartridges.

Auch die besonderen Talente des Sony Hit Bit stecken in Cartridges. Eine besonders pfiffige und praktische Lösung. Einfach eine dieser Cartridges in den dafür vorgesehenen Slot am Computer stecken und ab geht's im Programm. Die rein elektronisch gespeicherten Informationen stehen sofort zur Verfügung. Zum Beispiel das Programm HBS-H003C Home Writer. Sie schreiben einfach drauflos – korrigiert wird später. Sortieren, Radieren, Einfügen/Auslassen, alles geschieht auf Tastendruck. Verschiedene Papiergrößen, Schriftgrößen und sogar Formvordrucke können gespeichert und einfach mit dem Cursor, einem beweglichen Pfeilsymbol, ausgewählt werden. Sony's Home Writer nimmt Ihnen lästige Arbeiten ab und läßt mehr Zeit für das Wesentliche. Die Cartridge HBS-H001C ist besonders bei farbigen Grafiken gut in Form. Sie bietet eine Anzahl von phantasievollen Bildern, die im Format verändert und mit Schrift verschiedener Größe und Farbe versehen werden können. Das fertige Kunstwerk ist dann über den Sony Vierfarb-Plotter PRN-C 41 D reproduzierbar.

Die Sony 3,5 Zoll-Micro-Floppy-Diskette.

Noch schnellere Zugriffszeiten als der Bitcorder erlaubt ein besonders flexibles und handliches Speichermedium. Die millionenfach bewährte, 90 x 94 mm kleine 3,5 Zoll-Micro-Floppydisk.



Anwender schätzen ihre kompakten Abmessungen, die hervorragende Zuverlässigkeit und Benutzerfreundlichkeit. Trotz des begrenzten Formats bietet die beispielbare Micro-Floppy OM-D 3440 eine Speicherkapazität von 500 K-Byte – genug Platz für ca. 100 Seiten Text.



Der Micro-Floppydisk-Drive HBD-50 D.

Das Laufwerk zur Micro-Floppydisk. Angeschlossen wird es, wie bei Sony üblich, ganz einfach mit einem Griff. Das dazu notwendige Interface ist in eine Cartridge integriert, die wie ein Stecker in den dafür vorgesehenen Slot am Computer gesteckt wird. Mit

dem kompakten Micro-Floppydisk-Drive können Sie blitzschnell Programme, Texte usw. speichern oder laden. Eine weitere interessante Speichermöglichkeit bieten die Data-Cartridges HBI-55. Diese Halbleiterspeicher zeichnen sich durch besonders schnellen Datenzugriff aus.

Weil oft der erste Eindruck der entscheidende ist, hat Sony einige sinnvolle Hit Bit-Pakete zusammengestellt. Damit ist der Spaß am Computern programmiert. Beim Schreiben und Malen. Aber auch beim Spielen und Lernen. Und überhaupt.

SONY

TEXT-PRO – Ein Textverarbeitungsprogramm

Auch hier gilt, was wir schon beim Dateiverwaltungsprogramm geschrieben haben. Viele Leser- und -rückfragen zeigten uns wie groß das Interesse an solchen Programmen ist. Deshalb auch hier eine zweite Version eines Textverarbeiters.

Eine der vielen Aufgaben, die mittels eines Computers erledigt werden können ist das „Ver-“ und „Bearbeiten“ von Texten jeglicher Art. Für fast jeden Computer werden mehr oder weniger gute und teure Programme hierfür angeboten. Nichts gegen Profi-Programme vom Typ „Wordstar Version xx“. Diese sind hervorragend für denjenigen, der beruflich „textet“. Sie bieten Möglichkeiten, die für „Berufsschreiber“ nicht nur komfortabel, sondern auch erforderlich sind. Aber für den „Gelegenheitsschreiber“ sind die meisten dieser Programme zu leistungsfähig und meist auch zu schwierig in der Bedie-

nung. Wer hat denn schon Zeit, sich in tagelanger Einarbeitungszeit mit den Features eines „Profitexters“ auseinanderzusetzen?

Selbst wenn man sich aber diese Mühe gemacht hat, eine Woche später hat man den größten Teil der Bedienung wieder vergessen. Um dann einen Brief zu schreiben, der in wenigen Minuten geschrieben sein sollte, steckt man mehr Zeit in die wiederholte Einarbeitung des Programmes. Deshalb haben Programme wie das nachfolgende durchaus ihren positiven Sinn. Aber auch für „Selbstprogrammierer“ bietet ein derartiges Programm viele Anregungen und Tips. Deshalb nachfolgend ein weiteres Textverarbeitungsprogramm für Ihre zukünftigen Schreiben.

Programm-Beschreibung zum Programm TEXT-PRO

Das Programm TEXT-PRO ist ein Textverarbeitungsprogramm mit

den Möglichkeiten des Erstellens, Ändern, Formatierens und Druckens von Texten. Weiterhin verfügt es über einige Editier-Hilfen, wie das Einfügen, Löschen und Kopieren von Zeilen.

Ein Systemmenü sorgt für die optimale Ausnutzung der Bildschirmfarben, der Disketten- sowie der Tapedfunktionen.

Gedruckt wird mit einem Schneider-Drucker NLQ401. Das Druckmenü ist aber leicht an andere Drucker anzupassen. Die Hilfsfunktionen sind über eine Menüsteuerung leicht anzuwählen. Hier die Funktionen im Zusammenhang:

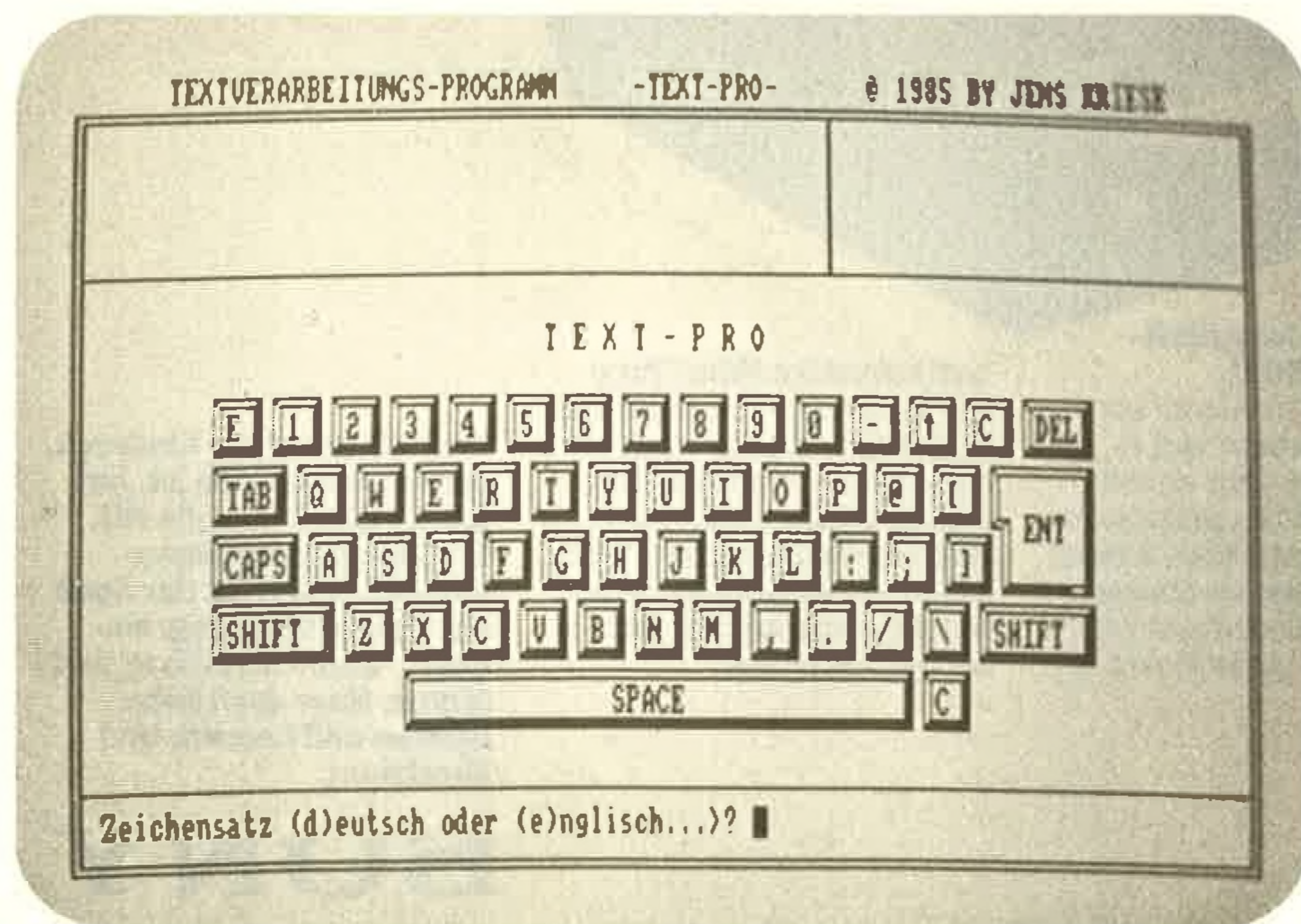
Nach dem Starten: Zunächst wird abgefragt, ob man mit dem englischen Standard-Zeichensatz oder mit dem deutschen Zeichensatz arbeiten möchte. Die Bildschirm- und Druckausgabe erfolgt dann mit dem gewählten Zeichensatz.

Die weitere Menüsteuerung erfolgt mit den Cursortasten (rechts-links). Die COPY-Taste ist zu drücken, wenn man die entsprechende Funktion ausführen möchte.

Im Editier-Mode erfolgt die Menüauswahl mit SHIFT- und den Cursor-Tasten. COPY hat die gleiche Funktion.

Das Hauptmenü: Im Hauptmenü sind neben den Menüsprüngen (mit SHIFT- und den Cursor-Tasten und COPY) noch einige Sonderfunktionen über die Tastatur abrufbar.

1. Die DEL-Taste behält ihre Funktion bei (löschen nach links).
2. Die CLR-Taste behält ihre Funktion auch bei (löschen nach rechts).
3. Die Cursor-Tasten behalten ihre normalen Funktionen.
4. SHIFT und Cursor up scrollt den Text um 20 Zeilen nach oben.
5. SHIFT und Cursor down scrollt den Text um 20 Zeilen nach unten.



6. Die TAB-Taste fügt ein Zeichen in die Textzeile ein (nach rechts über den Bildrand ragende Zeichen werden gelöscht).

7. Die RETURN-Taste geht mit dem Cursor in die nächste Zeile.

Das Menüangebot:

1. COPY – Die COPY-Funktion erlaubt es, Zeilen von einer Stelle des Textes zu einer anderen zu kopieren.

2. EINF – Hier wird eine Leerzeile nach der angegebenen Zeilennummer erzeugt.

3. LOES – Die Zeilen mit den angegebenen Nummern werden aus dem Text gelöscht.

4. SUCH –

a) Suchfunktion – Hier wird der Text nach einem bestimmten Begriff durchsucht. Wenn der

Begriff gefunden ist, springt der Cursor auf diesen.

b) Austauschfunktion – Im Text befindliche Worte können durch andere ersetzt werden.

5. FORM – Formatieren des aktuellen Textes auf eine bestimmte Breite. Worte werden nicht getrennt.

6. DRUC – Das Druckmenü enthält zunächst eine Abfrage nach der Schriftart: Normal, Komprimiert, Fettdruck, Schmaldruck oder NLQ-Druck sind möglich. Dann wird nach der Druckausführung gefragt. Diese kann Normal, mit doppeltem Anschlag oder hervorgehoben sein. Schließlich wird noch nach dem Zeilenabstand gefragt: Normal, Weit, Eng oder Sehr Eng sind möglich.

Bemerkung: TEXT-PRO druckt immer nur ab der Cursor-Position abwärts.

7. DISK – Unter diesem Menüpunkt kann ein bestehender Text geladen oder abgespeichert werden. Außerdem kann auch die Ausgabe des Disketteninhaltsverzeichnisses und die ERASE- und RENAME-Befehle aufgerufen werden. Aber auch die Bildschirmfarben und die Kassettengeschwindigkeit können an dieser Stelle beeinflusst werden. Der Pfeil kann angesprungen werden, um ins Hauptmenü zu gelangen.

8. NEU – Komplette Neu-Initialisierung des Programmes und Definition eines neuen Textes.

Jens Kriese/LM

Listing: TEXTPRO - ein Textverarbeitungsprogramm

```

1 '*****
2 '*
3 '*      (c) 1985 by Jens Kriese
4 '*
5 '* restart bei abbruch mit GOTO 830 *
6 '* oder RUN 35 (loescht alle Werte) *
7 '*      nach Ladefehler GOTO 30000 *
8 '*****
9 '
10 ON ERROR GOTO 1600
11 ON BREAK GOSUB 2310
12 SYMBOL AFTER 90
13 OPENOUT"dummy":MEMORY HIMEM-1:CLOSEOUT
14 GOSUB 3050
15 '-----variablen
16 aus=0:MODE 2:DIM t$(200):x=200
17 lin=1:name$="":cx=1:cy=1:cx1=1:cy1=1
18 INK 0,11:INK 1,1:BORDER 11
19 GOSUB 2340
20 '-----startmenue
21 CLS
22 GOSUB 1660:aus=1
23 INPUT#3,"Zeichensatz (d)eutsch oder (e)nglisch...";i$:i$=LOWER$(LEFT$(i$,1))
24 IF i$="d" THEN GOSUB 4360:GOTO 180:ELSE spr$="englisch":spr=1:GOTO 180
25 '
26 LOCATE#2,20,1:PRINT#2,spr$;
27 GOTO 270
28 '-----neuer text
29 CLS:CLS#1
30 LOCATE#2,2,3:PRINT#2,"Mode: Neuer Text"

```


Anwendungen

```
220 PRINT#3,"Name des neuen Textes...>";:LINE INPUT#3,i$
230 i$=i$+" " :name$=LEFT$(i$,8)
250 LOCATE #2,2,2:PRINT#2,"Text: "name$
260 x=200:ERASE t$:DIM t$(x):lin=1:FOR i=1 TO 25:t$(i)=STRING$(76," "):NEXT
265 GOTO 830
270 '-----menue
280 ON BREAK GOSUB 2310
290 ON ERROR GOTO 1600
300 CLS:CLS#1:PAPER 0:PEN 1
310 RESTORE 4530:GOSUB 3120
330 GOSUB 420
340 IF me=2 THEN GOTO 1390
350 IF me=8 THEN 200
360 GOTO 330
420 '-----eingabe bei menue
430 me=2
435 LOCATE#1,me,3:PRINT#1," ";CHR$(214)CHR$(215);
440 PRINT#3,"Bitte waehlen Sie mit den Cursortasten und Copy."
443 i$=INKEY$:FOR i=1 TO 100:NEXT
445 IF i$=CHR$(243) AND me<43 THEN mea=me:me=me+6:GOTO 470
446 IF i$=CHR$(242) AND me>7 THEN mea=me:me=me-6:GOTO 470
447 IF i$=CHR$(224) THEN mea=me:GOTO 480
450 GOTO 443
470 LOCATE#1,me,3:PRINT#1," ";:LOCATE#1,me,3:PRINT#1," ";CHR$(214)CHR$(215);
475 GOTO 443
480 CLS#3
485 LOCATE#1,me,3:PRINT#1," ";
490 RETURN
500 '-----text laden
520 CLS#2
530 LOCATE#2,2,3:PRINT#2,"Mode: Text laden"
540 PRINT#3,"Name des zu ladenden Textes...>";:LINE INPUT#3,la1$:la$=LEFT$(la1$,
8)+".tex"
550 ERASE t$
560 WINDOW SWAP 0,3
570 OPENIN la$
580 INPUT#9,x
590 DIM t$(x)
600 FOR i=1 TO x
610 INPUT#9,t$(i)
620 NEXT
630 CLOSEIN
631 CLS
640 WINDOW SWAP 0,3
650 name$=la1$
670 LOCATE#2,2,2:PRINT#2,"Text: "name$
680 GOTO 830
690 '-----text saven
700 CLS
710 LOCATE#2,2,3:PRINT#2,"Mode: Text abspeichern"
720 WINDOW SWAP 0,3
725 INPUT "Name des Textes...",ai$
730 ai$=LEFT$(ai$+" ",8)+".tex"
740 OPENOUT ai$
750 WRITE#9,x
```



```

760 FOR i=1 TO x
770 WRITE#9,t$(i)
780 NEXT
790 CLOSEOUT
791 CLS
800 WINDOW SWAP 0,3
820 GOTO 830
830 '-----text eingeben
840 CLS:CLS#3:CLS#1
850 '-----edit
860 lin=1
870 cx=1:cy=1
880 '-----print 1-15
900 CLS:CLS#3:FOR i=1 TO 15:LOCATE 1,i:PRINT LEFT$(t$(lin-1+i),76);:NEXT
920 '-----cursor setzen
935 CLS#1:RESTORE 4540:GOSUB 3120:LOCATE#2,2,3:PRINT#2,"Mode: Text editieren ":
LOCATE cx,cy:PRINT">";:GOSUB 1271
938 LOCATE cx,cy:PRINT">";
940 '-----inkey$
950 cx1=cx:cy1=cy
960 i$=INKEY$
970 LOCATE#3,49,1:PRINT#3,USING"ZEILE...>### SPALTE...>###";lin;cx
980 IF t$(lin)="" THEN T$(lin)=STRING$(76," ")
990 IF i$="" THEN 960
1000 IF i$=CHR$(240) THEN 2460
1010 IF i$=CHR$(241) THEN 2540
1020 IF i$=CHR$(243) THEN 2600
1030 IF i$=CHR$(242) THEN 2640
1040 IF i$=CHR$(244) THEN 2910
1050 IF i$=CHR$(245) THEN 2980
1060 IF i$=CHR$(127) THEN 2760
1061 IF i$=CHR$(247) OR i$=CHR$(246) THEN 1271
1063 IF i$=CHR$(224) THEN 1064 ELSE 1070
1064 IF me=2 THEN 3570 ELSE IF me=8 THEN 3750 ELSE IF me=14 THEN 3660 ELSE IF me
=20 THEN 3290 ELSE IF me=26 THEN 4090 ELSE IF me=32
THEN 2010 ELSE IF me=38 THEN 1390 ELSE IF me=44 THEN 200
1070 IF i$=CHR$(16) THEN 2800
1075 IF i$=CHR$(9) THEN 3850
1080 IF i$=CHR$(13) THEN 2840
1200 '-----eingabe in t$
1210 MID$(t$(lin),cx,1)=i$
1220 '-----print eingabe inkey$
1230 LOCATE cx1,cy1:PRINT MID$(t$(lin),cx1,1)
1240 '-----neue cursorposition
1250 IF cx<76 THEN cx=cx+1 ELSE GOTO 2840
1260 LOCATE cx,cy:PRINT">";
1270 GOTO 940
1271 '-----abfrage menue
1272 IF i$=CHR$(247) AND me<43 THEN mea=me:me=me+6:GOTO 1276
1273 IF i$=CHR$(246) AND me>7 THEN mea=me:me=me-6
1276 LOCATE#1,mea,3:PRINT#1," ";:LOCATE#1,me,3:PRINT#1," ";CHR$(214)CHR$(215)
;
1279 GOTO 960
1280 '-----input line
1290 LINE INPUT#3,i$

```



```

1300 IF i$<>" AND VAL(i$)<32 THEN f=VAL(i$):flag=1 ELSE flag=0
1310 i$=UPPER$(LEFT$(i$,1))
1320 RETURN
1330 '-----ende
1340 PRINT#3,"Wollen Sie den Text abspeichern...>";GOSUB 1280
1350 IF i$="J" THEN GOSUB 690
1360 PRINT#3,"Wollen Sie das Programm wirklich loeschen...>";GOSUB 1280
1370 IF i$="J" THEN CALL 0
1380 GOTO 935
1390 '-----systemmenue
1391 CLS#1:RESTORE 4580:GOSUB 3120
1392 LOCATE#2,2,3:PRINT#2,"Mode: System-Menue      "
1394 GOSUB 420
1395 WINDOW#4,58,79,8,22
1396 WINDOW#5,63,77,9,21
1400 IF ME=2 THEN GOTO 500
1410 IF ME=8 THEN GOTO 690
1420 IF ME=14 THEN 1421 ELSE 1430
1421 WINDOW SWAP 0,5:CLS#4:CLS:PLOT 470,280:DRAW 620,280:DRAW 620,60:DRAW 470,60
:DRAW 470,280:INPUT#3,"Wildcard (*.tex)";i$;!DIR,@i$
:WINDOW SWAP 0,5:GOTO 1394
1430 IF ME=20 THEN INPUT#3,"Welches Programm";o$:INPUT#3,"Neuer Name";n$;!REN,@n
$,@o$:GOTO 1394
1440 IF ME=26 THEN INPUT#3,"Welches Programm";o$;!ERA,@o$:GOTO 1394
1450 IF ME=32 THEN 1461 ELSE 1470
1461 INPUT#3,"Umschalten auf Kassette oder Disk";o$
1462 IF UPPER$(LEFT$(o$,1))="K" THEN !TAPE ELSE !DISC
1463 GOTO 1394
1470 IF ME=38 THEN 1550
1475 IF ME=44 THEN CLS:GOTO 870
1550 PRINT#3," Rahmenfarbe.....>";GOSUB 1280:IF flag=1 THEN BORDER f
1560 PRINT#3," Hintergrundfarbe....>";GOSUB 1280:IF flag=1 THEN INK 0,f
1570 PRINT#3," Schreibfarbe.....>";GOSUB 1280:IF flag=1 THEN INK 1,f
1580 PRINT#3," Speed Write (0/1)...>";GOSUB 1280:IF f=1 OR f=0 THEN SPEED WRITE
f
1585 CLS#3
1590 GOTO 1394
1600 '-----error
1610 CLS#1:PRINT#1:PRINT#1," Im Programm ist ein Fehler.", " Zeile      : "ERL
1620 PRINT#1," Fehlernummer: "ERR
1630 PRINT#3,"Programm abbrechen...>";GOSUB 1280
1640 IF i$="J" THEN 1330
1650 GOTO 830
1660 '-----tastatur
1670 CLS
1680 LOCATE 30,2:PRINT#2" T E X T - P R O
1690 a=25:b=a/6:z$="":RESTORE 4490
1700 y1=234:w=0:FOR i=73 TO 506 STEP 31:x1=i:GOSUB 1830:NEXT
1710 x1=x1+31:w=a/2:GOSUB 1830
1720 y1=y1-34:x1=73:w=a/2:GOSUB 1830
1730 w=0:FOR i=117 TO 459 STEP 31:x1=i:GOSUB 1830:NEXT
1740 x1=489:GOSUB 1910
1750 y1=y1-34:x1=73:w=a-a/4:GOSUB 1830
1760 w=0:FOR i=123 TO 465 STEP 31:x1=i:GOSUB 1830:NEXT
1770 y1=y1-34:x1=73:w=a+a/2:GOSUB 1830

```



```

1780 w=0:FOR i=142 TO 452 STEP 31:x1=i:GOSUB 1830:NEXT
1790 x1=483:w=a+a/2:GOSUB 1830
1800 y1=y1-34:x1=173:w=a*8+48:GOSUB 1830
1810 x1=452:w=0:GOSUB 1830
1820 RETURN
1830 READ z$
1840 PLOT x1,y1:DRAW x1+a+w,y1:DRAW x1+a+w,y1-a-a/10:DRAW x1,y1-a-a/10:DRAW x1,y
1
1850 PLOT x1+b,y1-b:DRAW x1+a+w-b,y1-b:DRAW x1+a+w-b,y1-a-a/10+b:DRAW x1+b,y1-a-
a/10+b:DRAW x1+b,y1-b
1860 PLOT x1,y1:DRAW x1+b,y1-b
1870 FOR t=x1+a+w TO x1+a+w-b STEP -1:PLOT t,y1+t-(x1+a+w):DRAW t,y1-a-a/10:NEXT

1880 FOR t=x1 TO x1+b:PLOT t,y1-a-a/10+t-x1:DRAW x1+a+w,y1-a:NEXT
1890 MOVE x1+b+b-1,y1-b-b:TAG #aus:PRINT#aus,Z$;:TAGOFF #aus
1900 RETURN
1910 READ Z$
1920 PLOT x1,y1:DRAW x1+a*2+6,y1:DRAW x1+a*2+6,y1-a*2-a/5-6:DRAW x1+a/4,y1-a*2-a
/5-6:DRAW x1+a/4,y1-a-a/5:DRAW x1,y1-a-a/5:DRAW x1,y
1
1940 PLOT x1+b,y1-b:DRAW x1+a*2+6-b,y1-b:DRAW x1+a*2+6-b,y1-a*2-a/5-6+b:DRAW x1+
a/4+b,y1-a*2-a/5-6+b:DRAW x1+a/4+b,y1-a-a/5+b:DRAW x
1+b,y1-a-a/5+b:DRAW x1+b,y1-b
1950 PLOT x1,y1:DRAW x1+b,y1-b:PLOT x1+a/4,y1-a-a/5:DRAW x1+a/4+b,y1-a-a/5+b:PLO
T x1,y1-a-a/5:DRAW x1+b,y1-a-a/5+b:PLOT x1,y1:DRAW x
1+b,y1-b
1960 FOR t=x1+a*2+6 TO x1+a*2+6-b STEP -1:PLOT t,y1+t-(x1+a*2+6):DRAW t,y1-a*2-a
/5-6:NEXT
1970 FOR t=x1+a/4 TO x1+a/4+b:PLOT t,y1-a*2-a/5-6+t-(x1+a/4)
1980 DRAW x1+a*2+a/4,y1-a*2-a/5-6:NEXT
1990 MOVE x1+b*2+10,y1-b*2-16:TAG:PRINT z$;:TAGOFF
2000 RETURN
2010 '-----ausdruck
2020 LOCATE#2,2,3:PRINT#2,"Mode: Drucken"
2050 WIDTH 80:PRINT#8,CHR$(27)"A"CHR$(12)CHR$(27)CHR$(84)CHR$(27)CHR$(72)CHR$(27
)CHR$(73)CHR$(1)CHR$(24)CHR$(18)CHR$(20);
2060 IF spr=0 THEN PRINT#8,CHR$(27);"6";
2070 IF spr=1 THEN PRINT#8,CHR$(27);"7";
2080 CLS#3:CLS#1:RESTORE 4550:GOSUB 3120
2090 GOSUB 420
2110 me1=me
2120 IF me=8 THEN PRINT#8,CHR$(15);:GOTO 2170
2121 IF me=2 THEN GOTO 2170
2130 IF me=14 THEN PRINT #8,CHR$(14);:GOTO 2170
2140 IF me=26 THEN PRINT #8,CHR$(27)CHR$(73)CHR$(3);:GOTO 2170
2150 IF me=20 THEN PRINT#8,CHR$(27)CHR$(83)CHR$(1);:GOTO 2170
2160 IF me<>2 THEN 2280
2170 IF me=26 THEN GOTO 2210 ELSE CLS#1:RESTORE 4560:GOSUB 3120:GOSUB 420
2190 IF ME1=8 AND ME=14 THEN ME=8
2200 IF ME=2 THEN PRINT#8,CHR$(27)CHR$(70); ELSE IF ME=14 THEN PRINT #8,CHR$(27)
CHR$(69); ELSE IF me=8 THEN PRINT#8,CHR$(27)CHR$(71)
; ELSE 2170
2210 CLS#1:RESTORE 4570:GOSUB 3120:GOSUB 420
2230 IF ME=2 THEN PRINT#8,CHR$(27);"3";CHR$(35); ELSE IF ME=8 THEN PRINT #8,CHR$
(27);"3";CHR$(70); ELSE IF ME=14 THEN PRINT#8,CHR$(2

```



```

7);"3";CHR$(24); ELSE IF me=20 THEN PRINT #8,CHR$(27);"3";CHR$(16);
2240 i=lin
2250 IF me1=14 THEN PRINT#8,CHR$(14);
2255 PRINT#3,"Drucken gestartet."
2260 IF (me1<>14 AND i<x+1 AND t$(i)<>"") THEN PRINT#8," ";LEFT$(t$(i),76);:i=i
+1:GOTO 2260
2270 IF (me1=14 AND i<x+1 AND t$(i)<>"") THEN PRINT#8,CHR$(14);" ";LEFT$(t$(i),
38):i=i+1:GOTO 2260
2280 GOTO 935
2310 '-----break
2320 ON BREAK GOSUB 2330
2330 GOTO 830
2340 '-----window normal
2350 CLS
2360 PRINT"          TEXTVERARBEITUNGS-PROGRAMM          -TEXT-PRO-          "CHR$(164)" 1985
      BY JENS KRIESE"
2370 FOR I=0 TO 1
2380 PLOT 1+I,380:DRAW 638+I,380:DRAW 638+I,1:DRAW 1+I,1:DRAW 1+I,380
2390 PLOT 5+I,376:DRAW 634+I,376:DRAW 634+I,4:DRAW 5+I,4:DRAW 5+I,376
2400 PLOT 5+I,296:DRAW 634+I,296
2410 PLOT 402+I,376:DRAW 402+I,296
2420 PLOT 5+I,40:DRAW 634+I,40
2430 NEXT
2440 WINDOW#0,3,79,8,22:WINDOW#1,3,50,3,6:WINDOW#2,52,79,3,6:WINDOW#3,3,79,24,24
2450 RETURN
2460 '-----cursor up
2470 IF lin>=1 THEN 2480 ELSE 938
2480 IF lin=1 THEN 938
2490 IF cy>1 THEN cy=cy-1 ELSE CALL 41301:cy1=cy1+1:LOCATE 1,1:PRINT LEFT$(t$(lin-1),76);
2500 IF cy=1 THEN LOCATE 1,1:PRINT LEFT$(t$(lin-1),76);
2510 LOCATE cx1,cy1:PRINT MID$(t$(lin),cx1,1)
2520 IF lin>1 THEN lin=lin-1
2530 GOTO 938
2540 '-----cursor down
2545 IF lin<x THEN 2550 ELSE GOTO 938
2550 IF cy<15 THEN cy=cy+1 ELSE CALL 41314:cy1=cy1-1
2560 IF cy=15 THEN LOCATE 1,15:PRINT LEFT$(t$(lin+1),76);
2570 LOCATE cx1,cy1:PRINT MID$(t$(lin),cx1,1)
2580 IF lin<x THEN lin=lin+1
2590 GOTO 938
2600 '-----cursor rechts
2610 IF cx<76 THEN cx=cx+1
2620 LOCATE cx1,cy1:PRINT MID$(t$(lin),cx1,1)
2630 GOTO 938
2640 '-----cursor links
2650 IF cx>1 THEN cx=cx-1
2660 LOCATE cx1,cy1:PRINT MID$(t$(lin),cx1,1)
2670 GOTO 938
2680 '-----copy cursor up
2690 IF cy>1 THEN cy=cy-1 ELSE CALL 41301:cy1=cy1+1:LOCATE 1,1:PRINT LEFT$(t$(lin-1),76);
2700 IF lin>1 THEN lin=lin-1
2710 RETURN
2720 '-----copy cursor down

```



```

2725 IF lin<x THEN 2730 ELSE RETURN
2730 IF cy<15 THEN cy=cy+1 ELSE CALL 41314:cy1=cy1-1:LOCATE 1,15:PRINT LEFT$(t$(lin+1),76);
2740 IF lin<x THEN lin=lin+1
2750 RETURN
2760 '-----del
2770 IF cx>1 THEN t$(lin)=LEFT$(t$(lin),cx-2)+RIGHT$(t$(lin),77-cx)+" "
2780 IF cx>1 THEN cx=cx-1:LOCATE 1,cy:PRINT t$(lin)
2790 GOTO 938
2800 '-----clr
2810 IF cx<76 THEN t$(lin)=LEFT$(t$(lin),cx-1)+RIGHT$(t$(lin),76-cx)+" "
2820 IF cx<76 THEN LOCATE 1,cy:PRINT t$(lin)
2830 GOTO 938
2840 '-----return
2850 cx=1
2860 IF cy<15 THEN cy=cy+1 ELSE CALL 41314:cy1=cy1-1
2870 IF cy=15 THEN LOCATE 1,15:PRINT t$(lin+1);
2880 LOCATE cx1,cy1:PRINT MID$(t$(lin),cx1,1)
2890 IF lin<x THEN lin=lin+1
2900 GOTO 938
2910 IF lin>1 THEN 2920 ELSE 2970
2920 IF lin-20>1 THEN aix=lin-20 ELSE aix=2
2930 LOCATE cx,cy:PRINT MID$(t$(lin),cx,1);
2940 FOR ii=lin TO aix STEP -1
2950 GOSUB 2680
2960 NEXT
2970 GOTO 938
2980 IF lin<256 THEN 2990 ELSE 3040
2990 IF lin+20<x+1 THEN aix=lin+20 ELSE aix=x
3000 LOCATE cx,cy:PRINT MID$(t$(lin),cx,1);
3010 FOR ii=lin TO aix
3020 GOSUB 2720
3030 NEXT
3040 GOTO 938
3050 '-----up mc laden
3060 RESTORE 4470
3070 FOR i=41301 TO 41326
3080 READ a
3090 POKE i,a
3100 NEXT
3110 RETURN
3120 '-----menue zeichnen
3130 CLS#1
3140 y1=366:a=25:b=a/6:w=a-a/4:FOR x1=16 TO 352 STEP 48:GOSUB 1830:NEXT
3280 RETURN
3290 '-----suchen/tauschen
3300 LOCATE#2,2,3:PRINT#2,"Mode: Suchen/Tauschen "
3310 PRINT#3," (S)uchen oder (A)ustauschen...>";:LINE INPUT#3,i$:i$=LEFT$(UPPER$(i$),1)
3320 IF i$="A" THEN 3350
3330 IF i$="S" THEN 3470
3340 GOTO 3310
3350 '-----austauschen
3360 PRINT#3," Auszutauschendes Wort...>";:LINE INPUT#3,u$:u=LEN(u$)

```



```
3370 PRINT#3," Zu ersetzen durch...>";:LINE INPUT#3,v$:v=LEN(v$):PRINT#3," Routine gestartet"
3380 FOR i=1 TO x:IF t$(i)="" THEN 3440
3390 FOR o=1 TO 77-u
3400 IF MID$(t$(i),o,u)<>u$ THEN 3430
3410 t$(i)=LEFT$(t$(i),o-1)+v$+MID$(t$(i),o+u,77-o-u)+"
3420 t$(i)=LEFT$(t$(i),76)
3430 NEXT o
3440 NEXT i
3460 cy=1:cy1=cy:CLS#3:GOTO 900
3470 '-----suchen
3480 PRINT#3," Zu suchende Sequenz...>";:LINE INPUT#3,u$:u=LEN(u$)
3490 PRINT#3," Routine gestartet"
3500 FOR i=1 TO x:IF t$(i)="" THEN 3550
3510 FOR o=1 TO 77-u
3520 IF MID$(t$(i),o,u)<>u$ THEN 3540
3530 lin=i:CLS#3:cy=1:cy1=cy:GOTO 870
3540 NEXT o
3550 NEXT i
3560 CLS#3:cy=1:cy1=cy:GOTO 900
3570 '-----kopieren
3580 LOCATE#2,2,3:PRINT#2,"Mode: Zeile kopieren "
3590 PRINT#3," Kopieren von Zeile...>";:LINE INPUT#3,i$:u=VAL(i$)
3600 PRINT#3," Kopieren bis Zeile...>";:LINE INPUT#3,i$:v=VAL(i$)
3610 PRINT#3," Kopieren nach Zeile...>";:LINE INPUT#3,i$:n=VAL(i$)
3620 FOR i=u TO v
3630 t$(n+i-u)=t$(i)
3640 NEXT i
3650 cy=1:cy1=cy:CLS#3:GOTO 900
3660 '-----loeschen
3670 LOCATE#2,2,3:PRINT#2,"Mode: Zeile loeschen "
3680 PRINT#3," Loeschen von Zeile...>";:LINE INPUT#3,i$:u=VAL(i$)
3690 PRINT#3," Loeschen bis Zeile...>";:LINE INPUT#3,i$:v=VAL(i$)
3700 FOR i=u TO x-1-(v-u)
3710 t$(i)=t$(i+(v-u)+1)
3720 NEXT i
3730 FOR i=x-1-(v-u) TO x:t$(i)="":NEXT
3740 cy=1:cy1=cy:CLS:CLS#3:GOTO 900
3750 '-----platz schaffen
3760 LOCATE#2,2,3:PRINT#2,"Mode: Zeile einfuegen "
3770 PRINT#3," Nach welcher Zeile einfuegen...>";:LINE INPUT#3,i$:u=VAL(i$)
3780 PRINT#3," Wieviele Zeilen einfuegen...>";:LINE INPUT#3,i$:v=VAL(i$)
3790 FOR i=x-v-1 TO u+1 STEP-1
3800 t$(i+v)=t$(i)
3810 NEXT i
3820 FOR i=u+1 TO u+v
3830 t$(i)=STRING$(76," "):NEXT
3840 cy=1:cy1=cy:CLS#3:GOTO 900
3850 '-----einfuegen
3860 IF cx<76 THEN t$(lin)=LEFT$(t$(lin),cx-1)+" "+LEFT$(RIGHT$(t$(lin),77-cx),76-cx)
3870 IF cx<76 THEN LOCATE 1,cy:PRINT t$(lin)
3880 GOTO 938
4090 '-----formatieren
4100 LOCATE#2,2,3:PRINT#2,"Mode: Text formatieren "
```



```

4110 CLS:PRINT#3," Wie breit soll der Text sein (20-76)...>";:LINE INPUT#3,i$:u=
VAL(i$):IF u<20 OR u>76 THEN 4110
4120 ii=1:e$="":w$="":DIM t1$(200)
4130 FOR i=1 TO x
4140 IF t$(i)=STRING$(76," ") THEN t1$(ii+1)="":t1$(ii)=LEFT$(t1$(ii)+t$(i),76):
ii=ii+2:GOTO 4200
4150 IF t$(i)=" " AND t1$(ii)=" " THEN ii=ii+1:GOTO 4200
4155 IF t$(i)=" " THEN t1$(ii+1)=t$(i):t1$(ii)=LEFT$(t1$(ii)+STRING$(76," "),76):
ii=ii+2:GOTO 4200
4160 FOR o=1 TO 76
4170 IF MID$(t$(i),o,1)<>" " THEN e$=e$+MID$(t$(i),o,1)
4180 IF MID$(t$(i),o,1)=" " AND MID$(t$(i),o+1,1)<>" " THEN e$=e$+" ":w$=e$:e$=""
":GOSUB 4220
4190 NEXT o
4200 IF ii<201 THEN NEXT i ELSE 4210
4210 GOTO 4240
4220 IF LEN(t1$(ii))+LEN(w$)<u THEN t1$(ii)=t1$(ii)+w$:w$="":ELSE t1$(ii)=LEFT$(
t1$(ii)+STRING$(76," "),76):ii=ii+1:GOTO 4220
4230 RETURN
4240 PRINT#3,"Soll der neu formatierte Text uebernommen werden...>";:LINE INPUT#
3,i$:i$=LEFT$(LOWER$(i$),1)
4250 IF UPPER$(i$)="J" THEN FOR i=1 TO X:t$(i)=t1$(i):NEXT
4255 ERASE t1$
4260 cy=1:cyl=cy:CLS#3:CLS:GOTO 850
4360 '-----tastatur
4370 spr$="deutsch":spr=0
4380 SYMBOL 123,198,0,120,12,124,204,118,0
4390 SYMBOL 125,198,0,102,102,102,102,62,0
4400 SYMBOL 124,198,0,60,102,102,102,60,0
4410 SYMBOL 126,120,198,198,252,198,198,248,192
4420 SYMBOL 91,219,60,102,102,126,102,102,0
4430 SYMBOL 93,198,0,198,198,198,198,60,0
4440 SYMBOL 92,198,56,198,198,198,108,56,0
4450 KEY DEF 17,1,123,91:KEY DEF 19,1,125,93:KEY DEF 26,1,124,92:KEY DEF 24,1,12
6:RETURN
4460 'datas fuer mc-code
4470 DATA &01,&0,&0,&11,&15,&4e,&21,&07,&02,&cd,&50,&bc,&c9
4480 DATA &01,&0,&01,&11,&15,&4e,&21,&07,&02,&cd,&50,&bc,&c9
4485 'datas fuer tastatur
4490 DATA E,1,2,3,4,5,6,7,8,9,0,"-", "^", C, DEL
4500 DATA TAB, Q, W, E, R, T, Y, U, I, O, P, "@", "[", ENT
4510 DATA CAPS, A, S, D, F, G, H, J, K, L, ":", ";", "]"
4520 DATA SHIFT, Z, X, C, V, B, N, M, ",", ".", "/", "\", SHIFT, " SPACE", C
4525 'datas fuer menues
4530 DATA DISK, NEU, , , , ,
4540 DATA COPY, EINF, LOES, SUCH, FORM, DRUC, DISK, NEU
4550 DATA NORM, KOMP, FETT, SCHM, NLQ, , ,
4560 DATA NORM, DOPP, HERV, , , , ,
4570 DATA NORM, WEIT, ENG, SEHR, , , , ,
4580 DATA LOAD, SAVE, DIR, REN, ERA, CASS, FARB, "->"
30000 CLOSEIN:WINDOW SWAP 3,0:DIM t$(200):GOTO 830

```


Arbeiten mit dem 2. Speicher beim CPC 6128

Der neue Schneider CPC 6128 wird von Haus aus mit einem Speicher von 128 KByte ausgerüstet. Dieser Speicher ist mit „normalen“ Basic-Befehlen nicht zu erreichen. Schneider liefert deshalb das Programm „BANKMAN“ mit. Mit dessen Hilfe können dem Rechner fünf RSX-Befehle mitgeteilt werden; dann stehen dem Anwender auch die zusätzlichen 64 KByte zur Verfügung.

„BANKMAN“ eignet sich hervorragend zum Arbeiten mit verschiedenen Bildschirmen. Vier Bildschirme können in den zweiten 64 KByte untergebracht werden. „SCREENCOPY“ und „SCREENSWAP“ ermöglichen schnelles und komfortables Arbeiten unter BASIC.

Auch über die anderen RSX-Befehle ist schon genügend geschrieben worden. „BANKREAD“ und „BANKWRITE“ erlauben die Speicherung von Stringvariablen in dem Zusatzspeicher; mit „BANKFIND“ ist sogar eine Suche mit Wildcards vorgesehen. Also, warum sich noch weiter mit den BANK-Operationen auseinandersetzen, wo doch dank „BANKMAN“ alles wunderbar funktioniert?

Etwas anderes: Haben Sie schon einmal probiert, den zweiten Speicher komplett zu löschen? Programm 1 soll das für uns machen. Die Aktion läuft relativ schnell, aber leider fehlt noch 1 Byte. Also erhöhen wir „l“ auf 256. Dies wiederum nimmt uns das BASIC übel, denn Stringvariablen dürfen nur maximal 255 Bytes lang sein. Deshalb soll „l“ 128 und „n“ 512 sein; nun werden

in 512 Durchgängen die gesamten 65 536 Bytes gelöscht. Aber, haben Sie schon einmal auf die Zeit geachtet? 4,53 Sekunden! Das kann ja noch nicht das Gelbe vom Ei sein. Grund genug, sich doch etwas mit dem Zusatzspeicher und seiner Ansteuerung zu beschäftigen. Um der Theorie auch etwas Würze zu geben, schreiben wir ein Programm, das den Zusatzspeicher ganz löscht und nicht soviel Zeit dafür benötigt. „BANKCLEAR“ heißt es, denn genau das soll es auch nachher machen. Für diejenigen, denen Maschinensprache im Moment noch nicht soviel sagt, haben wir auch einen BASIC-Lader als Programm 2 in diesen Artikel mit aufgenommen. Dieser BASIC-Lader erlaubt, in gewissen Grenzen, „BANKCLEAR“ an verschiedene Startadressen im Speicher anzusiedeln; unerlaubte Adressen werden von ihm ignoriert. Warum sind bestimmte Speicherbereiche für die Ansiedlung von „BANKCLEAR“ nicht erlaubt? Die Antwort wird schon im Handbuch mit einem kurzen Satz gestreift: „Beachten Sie, daß jeder Block aus den zweiten 64 KByte an derselben Adresse abgelegt wird.“ Der Bereich liegt zwischen &4000 und &7FFF. Wird der zweite Speicher angesprochen, so wird im „Hauptspeicher“ ohne Rücksicht auf irgendwelche vorhandenen Daten der Bereich ausgeblendet; stattdessen finden wir an dieser Stelle nun einen Block aus dem zweiten Speicher.

Ein Block ist 16 KByte lang, folglich besteht der Zusatzspeicher aus vier Blöcken. Die einzelnen Blöcke haben eine Kennung von 4 bis 7. Diese Kennung muß dem Rechner mitge-

teilt werden; so weiß er, welcher Block gerade „up to date“ ist.

„CALL BANK“ ermöglicht diese Mitteilung, wobei BANK für die Adresse &BD5B steht. Hier wird eine Routine im ROM aufgerufen, die die Bankwahl durch einen „OUT(c),c“-Befehl vornimmt. Die Blockkennung muß sich dabei im Register „A“ befinden. Will man also eine Bank aus dem Zusatzspeicher benutzen, so wird die Kennung nach „A“ geladen und „CALL BANK“ aufgerufen. Der Speicherbereich &4000 bis &7FFF wird nun durch den angewählten Block ersetzt und kann beschrieben werden. Würde unser Programm „BANKCLEAR“ in diesem Bereich liegen, so würde es sich beim ersten „CALL BANK“ selbst ausblenden; ein Absturz des Rechners wäre unvermeidlich.

Was macht nun unser kleines Programm? Register „A“ wird mit der ersten Bankkennung geladen; danach wird der erste Block geladen und gelöscht. „A“ wird erhöht und der zweite Block wird gelöscht. Als Zähler fungiert übrigens das Register „C“. Wird in „C“ der Wert 8 erreicht, dann ist es höchste Zeit, das Programm zu verlassen. Zuvor wird der Hauptspeicher aber wieder in seinen Originalzustand versetzt. Dazu bekommt das Register „A“ den Wert 0; ein letztesmal wird „CALL BANK“ aufgerufen. Der Hauptspeicher ist wieder komplett, der Zweitspeicher gelöscht, und das Maschinenprogramm zu Ende.

Haben Sie noch Interesse an einem zweiten Programm, das die Bankbefehle erweitert? Dann sollten Sie weiter lesen. Stellen Sie sich vor, Sie haben einen Speicherbereich,

den Sie in den Zweitspeicher übertragen möchten. Natürlich können Sie dies mit Hilfe des „BANKMAN“ tun. Wie, das zeigt Programm 3. Übergeben an „BANKWRITE“ wird nun nicht mehr eine Variable, sondern eine Adresse; es ist also nicht nötig, den Speicherinhalt erst einer Variablen „aufzuhalsen“. Umgekehrt funktioniert es bei „BANKREAD“ natürlich genauso; die Daten aus dem zweiten Speicher werden direkt an die Adresse im Hauptspeicher geladen.

Interessanter wird es bei einem „BANKSWAP“-Befehl. Hierbei wird ein Speicherbereich mit einem anderen Speicher vertauscht. Im Programm 4 wird gezeigt, wie beim „BANKMAN“ so etwas realisiert werden kann. Getauscht werden sollen, der Einfachheit halber, zwei 16-KByte-Bereiche. Die Bereichsgrenzen sind frei wählbar. Leider können wir den eingebauten „SCREENSWAP“-Befehl nicht benutzen; wir müßten jedesmal die Daten in den Bildschirmspeicher laden und anschließend wieder zurückholen. Das ist nicht nur sehr zeitaufwendig, sondern es sieht auch merkwürdig aus.

Unter Maschinensprache geht der Vorgang wieder erheblich schneller. Das Programm benötigt zu allererst zwei Zwischenspeicher. Aus dem zuvor gelesenen wissen wir, daß diese Speicher oberhalb von &7FFF liegen müssen; daher werden Sie an das Ende des Programms gehängt. „BANKSWAP“ kann also wiederum nur im Bereich über &7FFF abgelegt werden; der BASIC-Lader berücksichtigt das.

Das Programm gliedert sich grob in drei Teile. Teil 1 beginnt bei

BANKSW. Hier wird kontrolliert, ob die angewählte Bank überhaupt zulässig ist. Dabei steht eine „0“ für die Bank im Hauptspeicher; „1“ bis „4“ bezeichnen die Banken im Zusatzspeicher. Das Subprogramm bei BANKS1 berechnet die korrekten Kennungen.

Bei BANKS2 beginnt dann der eigentliche SWAP-Vorgang. Zuerst wird ein Speicherteilbereich im ZWSP1 gesichert. Dann wird der Hauptspeicher ausgeblendet; die Daten aus dem Zusatzspeicher werden nach ZWSP geladen (entspricht „BANKREAD“). Anschließend wird der Inhalt von ZWSP1 an die entsprechende Adresse im Zusatzspeicher gesandt („BANKWRITE“). Der Hauptspeicher wird nun eingeblendet; der Inhalt von ZWSP kommt an seinen Platz. Dieser Vorgang wiederholt sich bei unserem Beispiel 64mal; danach sind zwei 16-KByte-Blöcke ausgetauscht.

Teil 3 beginnt bei BANKSO. Bisher wurde nur die Bank aus dem Hauptspeicher, z. B. Bank 3, wieder an ihren Platz im Zusatzspeicher geschickt. Jetzt soll aber auch eine neue Bank, z. B. Bank 4, in den Arbeitsspeicher geladen werden. Diese neue Bank muß bei dem Aufruf von „BANKSWAP“ mit übergeben werden. Der Vorgang beginnt von neuem: Der Arbeitsspeicher (Bank 0) wird in den Zusatzspeicher geladen; Bank 4 nimmt den Platz von Bank 0 ein.

Worin liegt der Vorteil bei dem doppelten Bankgeschiebe? Schneller ginge es, wenn man den 16-KByte-Block im Hauptspeicher als reinen Arbeitsspeicher betrachten würde. Dann bräuchten die Bänke aus dem Zusatzspeicher nur dorthin geladen

zu werden; einfache Blockladebefehle des Z80 würden genügen. Leider stehen dem Anwender dann aber nur 64 KByte zur Verfügung; mit unserem „BANKSWAP“ liegen wir aber bei 80 KByte. Grund genug, wie ich meine, dieses etwas aufwendigere Verfahren zu benutzen.

„BANKSWAP“ wird mit einem CALL-Befehl aufgerufen. Dabei muß zu Beginn die Adresse von Bank 0 im Hauptspeicher angegeben werden. Mit „CALL &9F00, 0, &3000“ wird der Beginn von Bank 0 auf &3000 festgelegt; Bank 0 umfaßt also in diesem Beispiel den Speicherbereich &5000 bis &6FFF. Gleichzeitig finden wir auch unser „BANKCLEAR“ wieder; es wird ausgeführt, wenn zwei Parameter an das Programm übergeben werden. So wird der gesamte Zusatzspeicher gelöscht.

„CALL &9F00,2“: hiermit wird in den Arbeitsspeicher die Bank 2 geladen. Die bisherige Bank 0 wird unter der Adresse von Bank 2 gespeichert. Ein nochmaliger Aufruf von Bank 2 würde außer Zeitverlust nichts ändern.

Programm 5 enthält den BASIC-Lader für BANKSWAP. Das korrekte Programm wird automatisch gesichert und dann aufgerufen. Denken Sie daran, daß BANKSWAP zuerst die Bereichsgrenzen des Speichers mitgeteilt werden müssen. Dabei wird immer der gesamte Speicher gelöscht. In Programm 5 wird diese Bedingung vor dem ersten Aufruf von BANKSWAP erfüllt.

Natürlich können Sie auch diesen Befehl wieder als RSX-Befehl in Ihre Programme einbinden. Das Assemblerlisting kann Ihnen da behilflich sein. F. BERG/LM

ARNOR Z80 ASSEMBLER version 1.10

Page 001

```

00002      ;      BANKSWAP      fuer CPC 6128
00003      ;                      (c) 11/85 f.berg
00004      ;
00005      ;      CALL bankswap,0,spa (Aufruf Bankswap Init)
00006      ;      CALL bankswap,bank      (Aufruf Bankswap)
00007      ;
00008
00009      9F00      (A474)          ORG      &9F00,5
00010
00011      9F00      (0100)      SLEN      EQU      &100      ;Satzlaenge

```


Anwendungen

```

00012 9F00 (0140) BANKNR EQU &140 ;Banknummer (intern)
00013 9F00 (0141) SEITE EQU &141 ;alte Banknummer
00014 9F00 (0142) SPBANK EQU &142 ;Adr. Bankspeicher
00015 9F00 (0144) SPA EQU &144 ;Hauptspeicheranfang
00016 9F00 (0146) SPASP EQU &146 ;Zwischenspeicher
00017
00018 ; Systemspruenge
00019
00020 9F00 (BD5B) BANK EQU &BD5B
00021
00022
00023 9F00 FE 02 BKINIT CP 2 ;Uebergabe 2 Parameter?
00024 9F02 20 27 JR NZ,BANKSW
00025
00026 9F04 ED 53 44 01 LD (SPA),DE ;Speicheranfang
00Q27
00028 9F08 0E 04 LD C,4 ;Die einzelnen Baenke
00029 9F0A 79 BANKCL LD A,C ; werden geloesch.
00030 9F0B CD 5B BD CALL BANK ;Ausblendung Hauptspeicher
00031 9F0E C5 PUSH BC ;C wird gerettet.
00032
00033 9F0F 01 FF 3F LD BC,&3FFF ;Die Baenke liegen immer
00034 9F12 21 00 40 LD HL,&4000 ; im Bereich zwischen
00035 9F15 36 20 LD (HL)," " ; &4000 und &7FFF.
00036 9F17 11 01 40 LD DE,&4001
00037 9F1A ED B0 LDIR
00038
00039 9F1C C1 POP BC ;C wird geholt
00040 9F1D 0C INC C ;Die naechste Bank wird
00041 9F1E 79 LD A,C ; ausgesucht.
00042 9F1F FE 08 CP 8 ;Ist C=8, dann Ende.
00Q43 9F21 20 E7 JR NZ,BANKCL
00044
00045 9F23 97 SUB A ;Hauptspeicher wird ein-
00046 9F24 CD 5B BD CALL BANK ; geblendet.
00047 9F27 32 41 01 LD (SEITE),A
00048 9F2A C9 RET
00049
00050 9F2B ED 5B 41 01 BANKSW LD DE,(SEITE) ;alte Banknummer
00051 9F2F CD 99 9F CALL BANKS1
00052 9F32 38 03 JR C,BANKSO ;Carry bei 0<Bank>4
00053 9F34 CD 44 9F CALL BANKS2
00054
00055 9F37 DD 5E 00 BANKSO LD E,(IX+0) ;neue Banknummer
00056 9F3A 16 00 LD D,0
00057 9F3C ED 53 41 01 LD (SEITE),DE
00058 9F40 CD 99 9F CALL BANKS1
00059 9F43 D8 RET C
00060
00061 ; *** BANKSWAP
00062
00063 9F44 06 40 BANKS2 LD B,64 ;Initialisierung
00064 9F46 21 00 40 LD HL,&4000 ; B ist Zaehler
00065 9F49 22 42 01 LD (SPBANK),HL
00066 9F4C 2A 44 01 LD HL,(SPA)
00067 9F4F 22 46 01 LD (SPASP),HL
00068
00069 9F52 C5 BANKS3 PUSH BC ;B wird gerettet.

```


00070						
00071	9F53	01 00 01		LD	BC, SLEN	;Daten aus Hauptspeicher
00072	9F56	11 A9 A0		LD	DE, ZWSP1	; werden nach ZWSP1 ge-
00073	9F59	E5		PUSH	HL	; laden.
00074	9F5A	C5		PUSH	BC	;Rette Adr. Hauptspeicher
00075	9F5B	ED B0		LDIR		; und SLEN.
00076						
00077	9F5D	3A 40 01		LD	A, (BANKNR)	;interne Banknummer
00078	9F60	4F		LD	C, A	
00079	9F61	CD 5B BD		CALL	BANK	;Hauptspeicher ausblenden
00080						
00081	9F64	C1	BANKRD	POP	BC	;SLEN
00082	9F65	2A 42 01		LD	HL, (SPBANK)	;Bankspeicher
00083	9F68	11 A8 9F		LD	DE, ZWSP	;Zwischenspeicher
00084	9F6B	D5		PUSH	DE	;Daten werden aus einer
00085	9F6C	E5		PUSH	HL	; Bank nach ZWSP gelesen.
00086	9F6D	C5		PUSH	BC	
00087	9F6E	ED B0		LDIR		
00088						
00089	9F70	C1	BANKWR	POP	BC	;SLEN
00090	9F71	21 A9 A0		LD	HL, ZWSP1	;SWAP-Speicher
00091	9F74	D1		POP	DE	;Adr. Bankspeicher
00092	9F75	C5		PUSH	BC	;Daten aus ZWSP1 werden in
00093	9F76	ED B0		LDIR		; eine Bank gelesen.
00094						
00095	9F78	97		SUB	A	
00096	9F79	CD 5B BD		CALL	BANK	;Hauptspeicher einblenden
00097						
00098	9F7C	C1		POP	BC	;SLEN
00099	9F7D	E1		POP	HL	;Zwischenspeicher ZWSP
00100	9F7E	D1		POP	DE	;Adr. im Hauptspeicher
00101	9F7F	ED B0		LDIR		;ZWSP -> Hauptspeicher
00102						
00103	9F81	C1		POP	BC	;B wird geholt
00104	9F82	10 01		DJNZ	BANKS5	;Ist B=0, dann Ende.
00105	9F84	C9		RET		
00106						
00107	9F85	11 00 01	BANKS5	LD	DE, SLEN	
00108	9F88	2A 42 01		LD	HL, (SPBANK)	;Adr. Banksp.+SLEN
00109	9F8B	19		ADD	HL, DE	
00110	9F8C	22 42 01		LD	(SPBANK), HL	
00111						
00112	9F8F	2A 46 01		LD	HL, (SPASP)	;Adr. Hauptsp.+SLEN
00113	9F92	19		ADD	HL, DE	
00114	9F93	22 46 01		LD	(SPASP), HL	
00115	9F96	C3 52 9F		JP	BANKS3	
00116						
00117	9F99	97	BANKS1	SUB	A	;Ist Banknr.0 oder
00118	9F9A	BB		CP	E	; >4, dann RET C.
00119	9F9B	37		SCF		
00120	9F9C	C8		RET	Z	
00121	9F9D	3E 05		LD	A, 5	
00122	9F9F	BB		CP	E	
00123	9FA0	D8		RET	C	
00124						
00125	9FA1	7B		LD	A, E	;interne Banknr. liegt
00126	9FA2	C6 03		ADD	&03	; zwischen 4 und 7.
00127	9FA4	32 40 01		LD	(BANKNR), A	

Anwendungen

```

00128 9FA7 C9 RET
00129
00130 9FA8 (9FA8) ZWSP EQU $ ;Zwischenspeicher
00131 9FA8 (A0A9) ZWSP1 EQU $+257 ;SWAP-Speicher
00132
00133 9FA8 (A1A9) ENDE EQU $+257+256

```

Errors: 00000 Warnings: 00000

SYMBOL TABLE:

0140 BANKNR	BD5B BANK	9F00 BKINIT	9F0A BANKCL
9F2B BANKSW	9F37 BANKSO	9F44 BANKS2	9F52 BANKS3
9F64 BANKRD	9F70 BANKWR	9F85 BANKS5	9F99 BANKS1
A1A9 ENDE	0100 SLEN	0141 SEITE	0142 SPBANK
0144 SPA	0146 SPASP	9FA8 ZWSP	A0A9 ZWSP1

Listing: Programm 1

```

100 REM *** Programm 1
110 REM Loeschen des Zusatzspeichers
120 '
130 IF HIMEM=37631 THEN 170
140 MEMORY &92FF
150 LOAD"bankman.bin",&9300:CALL &9300
160 '
170 n=257:l=255:DEFINT a:a=0: BANKOPEN,l:d$=STRING$(l," ")
180 t=TIME
190 FOR m=0 TO n
200 BANKWRITE,@a,d$,m
210 NEXT:PRINT"Zeit:";(TIME-t)/300;"s"

```

Listing: Programm 2

```

100 REM *** Programm 2
110 REM BANKCLEAR
120 '
130 MODE 1:LOCATE 12,5:PRINT"BANKCLEAR"
140 LOCATE 2,11:PRINT"Bei welcher Adresse soll BANKCLEAR"
150 LOCATE 2,13:INPUT "beginnen";adresse
160 IF adresse<0 THEN adresse=65536+adresse
170 IF adresse>32767 AND adresse<42587 THEN 200
180 LOCATE 2,18:PRINT"Die Adresse ist nicht erlaubt!":CALL &BB06:RUN
190 '
200 LOCATE 2,11:PRINT"BANKCLEAR wird nun gespeichert. "
210 LOCATE 2,13:PRINT STRING$(38," ")
220 LOCATE 2,15:PRINT"Sie koennen BANKCLEAR dann mit"
230 LOCATE 4,17:PRINT"SAVE 'BNKCLEAR',b,&" ;HEX$(adresse);",&20"
240 LOCATE 2,19:PRINT"auf Diskette sichern."
250 '
260 MEMORY adresse-1:RESTORE 320
270 FOR n=adresse TO adresse+31

```



```
280 READ a$:POKE n,VAL("&"a$)
290 NEXT
300 LOCATE 12,11:PRINT" ist":LOCATE 1,24:END
310 '
320 DATA 0E,04,79,CD,5B,BD,C5,01,FF,3F,21,00,40,36,20,11
330 DATA 01,40,ED,B0,C1,0C,79,FE,08,20,E7,97,CD,5B,BD,C9
```

Listing: Programm 3

```
100 REM *** Programm 3
110 REM Speichern/Laden mit Block 1 (Zusatzspeicher)
120 '
130 IF HIMEM=37631 THEN 170
140 MEMORY &92FF
150 LOAD"bankman.bin",&9300:CALL &9300
160 '
170 l=128:DEFINT a:a=0:BANKOPEN,l:d$=STRING$(l," "):POKE &15C,l
180 spa=&3000:spe=&6FFF
190 MODE 2:INPUT"1) Bereich speichern 2) Bereich laden ";a
200 POKE &15C,l:IF a=2 THEN GOTO 310
210 '
220 'BANKWRITE
230 t=TIME:z=0
240 FOR m=spa TO spe STEP 1
250 POKE &15E,INT(m/256):POKE &15D,m-PEEK(&15E)*256
260 BANKWRITE,@a,&15C,z:z=z+1
270 NEXT:PRINT"Zeit:";(TIME-t)/300;"s"
280 END
290 '
300 'BANKREAD
310 t=TIME:z=0
320 FOR m=spa TO spe STEP 1
330 POKE &15E,INT(m/256):POKE &15D,m-PEEK(&15E)*256
340 BANKREAD,@a,&15C,z:z=z+1
350 NEXT:PRINT"Zeit:";(TIME-t)/300;"s"
```

Listing: Programm 4

```
100 REM *** Programm 4
110 REM BANKSWAP in Basic
120 '
130 MODE 2:IF HIMEM=12287 THEN 170
140 MEMORY &2FFF
150 LOAD"bankman.bin",&9300:CALL &9300
160 '
170 l=128:DEFINT a:a=0:BANKOPEN,l:d$=STRING$(l," "):POKE &15C,l
180 spa=&3000:spe=&6FFF
190 '
200 INPUT"Bitte geben Sie die gewuenschte Bank an: ",b:PRINT
210 IF b<0 OR b>4 THEN PRINT"Falsche Eingabe!":PRINT:GOTO 200
220 '
```


Anwendungen

```
230 c$="speichert.":GOSUB 260:bank=b:c$="laden.":GOSUB 260
240 PRINT"BANKSWAP ist beendet.":END
250 '
260 IF bank=0 THEN RETURN
270 PRINT"Bank";bank;"wird ge";c$:z=(bank-1)*128
280 FOR m=spa TO spe STEP 1
290     n=@d$:GOSUB 340:d$=d$+" "
300     n=&15C:GOSUB 340
310     !BANKREAD,@a,&15C,z:!BANKWRITE,@a,@d$,z:z=z+1
320 NEXT:RETURN
330 '
340 POKE n+2,INT(m/256):POKE n+1,m-256*PEEK(n+2):RETURN
```

Listing: Programm 5

```
100 REM *** Programm 5
110 REM     BANKSWAP in M-Code
120 '
130 MODE 1:MEMORY &2FFF:adr=&9F00
140 LOCATE 5,9:PRINT"Die Werte von BANKSWAP werden"
150 LOCATE 5,10:PRINT"gespeichert."
160 LOCATE 5,15:PRINT"Bitte warten Sie."
170 zeile=490
180 '
190 zeile=zeile+10:IF zeile<>610 THEN 240
200 MODE 2:LOCATE 5,9:PRINT"BANKSWAP wird unter 'BANKSWAP.BIN' gespeichert."
210 LOCATE 5,11:PRINT"SAVE 'BANKSWAP.BIN',b,&9F00,&A7"
220 SAVE"BANKSWAP.BIN",b,&9F00,&A7
230 PRINT:GOTO 310
240 pruef=0:FOR n=adr TO adr+15
250     READ a$:POKE n,VAL("&"+a$):pruef=pruef+PEEK(n)
260 NEXT
270 READ a$:a=VAL(MID$(a$,3)):adr=n:IF a=pruef THEN 190
280 MODE 2:LOCATE 10,9:PRINT"Fehler in Zeile";zeile:PRINT
290 LIST 500-
300 '
310 spa=&3000:spe=&6FFF:CALL &9F00,0,spa
320 '
330 INPUT"Bitte geben Sie die gewuenschte Bank an: ",b
340 CALL &9F00,b
350 PRINT"BANKSWAP ist beendet.":PRINT
360 GOTO 330
370 '
500 DATA FE,02,20,27,ED,53,44,01,0E,04,79,CD,5B,BD,C5,01,= &602
510 DATA FF,3F,21,00,40,36,20,11,01,40,ED,B0,C1,0C,79,FE,= &628
520 DATA 08,20,E7,97,CD,5B,BD,32,41,01,C9,ED,5B,41,01,CD,= &71F
530 DATA 99,9F,38,03,CD,44,9F,DD,5E,00,16,00,ED,53,41,01,= &5F6
540 DATA CD,99,9F,D8,06,40,21,00,40,22,42,01,2A,44,01,22,= &47A
550 DATA 46,01,C5,01,00,01,11,A9,A0,E5,C5,ED,B0,3A,40,01,= &62A
560 DATA 4F,CD,5B,BD,C1,2A,42,01,11,A8,9F,D5,E5,C5,ED,B0,= &8D6
570 DATA C1,21,A9,A0,D1,C5,ED,B0,97,CD,5B,BD,C1,E1,D1,ED,= &B3A
580 DATA B0,C1,10,01,C9,11,00,01,2A,42,01,19,22,42,01,2A,= &372
590 DATA 46,01,19,22,46,01,C3,52,9F,97,BB,37,C8,3E,05,BB,= &5CC
600 DATA D8,7B,C6,03,32,40,01,C9,00,00,00,00,00,00,00,= &35B
```


Der CPC 464 programmiert sich selbst!

Mit der folgenden RSX-Erweiterung, die nur auf dem Schneider-CPC-464 lauffähig ist, können Sie den Computer sich selbst programmieren lassen. So lassen sich z. B. DEF-FN-Funktionen per INPUT eingeben oder sich selbst modifizierende Programme schreiben.

Sie erhalten ein lauffähiges Programm auf zwei Wegen: erstens durch Eingabe des Quellcodes und Assemblierung des Maschinenprogramms, oder zweitens durch Abtippen des Basic-Loaders. Achten Sie beim Basic-Loader aber sehr genau auf die Numerierung der Programmzeilen – Sie sehen später, warum!

Nach dem Programmstart steht Ihnen ein neuer RSX-Befehl zur Verfügung:

```
|LINE,@a$
```

Im angegebenen String muß eine vollständige Basic-Zeile stehen:

```
a$="100 PRINT"  
a$="150 FOR i=1  
TO 10:PRINT i:NEXT"
```

Die Stringvariable kann aufgrund des CPC-Betriebssystems also nicht direkt übergeben werden, sondern nur auf dem Umweg über den Variablenzeiger. Diese Funktion wird mit dem „Klammeraffen“ (ASCII-Code 64) aufgerufen. Diskettenbesitzer

kennen das sicher schon vom ERA- und REN-Befehl.

Ein Programm-Beispiel dafür:

```
10 b$="100 INPUT x$"  
20 |LINE,@b$  
30 LIST 100
```

Der Computer hat dann die Zeile 100 in den Speicher übernommen und kann sie ausführen. So lassen sich z. B. auf sehr einfache Weise Funktionsplotter programmieren, die die darzustellende Funktion mit einem INPUT abfragen:

```
10 INPUT "Welche Funktion",a$  
20 a$="40 DEF FN f(x)=" + a$  
30 |LINE,@a$  
50 FOR i=1 TO 3  
60 PRINT i;FN f(i)  
70 NEXT i
```

Nun noch einige Eigenheiten des Programms, die von Ihnen unbedingt beachtet werden müssen:

– Bei der Ausführung des LINE-Befehls wird der Basic-Stack vom Computer gelöscht. Dort befinden sich alle offenen FOR-NEXT- und WHILE-WEND-Schleifen und Rücksprungadressen aus GOSUB-Unterprogrammen. Sie müssen also darauf achten, daß der LINE-Befehl nicht innerhalb offener Schleifen oder in Unterprogrammen ausgeführt wird. Während Sie aufgrund letzterer Einschränkung eventuell Ihre Programmstruktur verändern

müssen, lassen sich Schleifen meist sehr einfach in IF-THEN-Konstruktionen verwandeln.

– Die zu programmierende Zeile muß eine höhere Zeilennummer besitzen als die Zeile, in der der LINE-Befehl steht. Das ist aber durchaus sinnvoll, da ja die neue Zeile erst noch ausgeführt werden soll.

Zurück zum Basic-Loader: Dieser ist selbst ein Beispiel für die Verwendung des LINE-Befehls. Die Zeilen 310–340 werden in Zeile 300 aus den DATA-Zeilen zwischen Zeile 530 und 560 gelesen und sofort danach ausgeführt.

Für Programmierer und sonstige Interessierte einige Bemerkungen zur Funktionsweise von LINE: Das Programm übernimmt den String, der beim RSX-Aufruf angegeben wird, in den Eingabepuffer des Computers. Dieser liegt beim CPC-464 zwischen den Adressen &ACA4 und &ADA3; er ist also 255 Zeichen lang, entsprechend der maximalen Eingabelänge des Editors. Dann schaltet LINE das Basic-ROM ein und ruft die Codierungs-Routine im Basic-Interpreter auf. Diese liegt beim CPC-464 an der Adresse &E6BC. Da der Aufbau der Codier-Routinen bei den beiden Nachfolgecomputern CPC-664 und 6128 verändert wurde, lassen sich auch keine neuen Einsprungstellen angeben. Damit ist LINE nur auf dem Erstlings-Computer von Schneider lauffähig.

Martin Kotulla/LM

Listing: LINE

```
100 ' *****  
110 ' *  
120 ' * RSX-Erweiterung LINE *  
130 ' *  
140 ' *****
```



```

150 '
160 ' Bitte bei der Eingabe die Zeilennummern beibehalten!
170 '
180 SYMBOL AFTER 254:SYMBOL 255,&66,&0,&3C,&66,&66,&66,&3C,&0
190 MEMORY HIMEM-&50:start=HIMEM+1
200 DEF FNmsb(a)=INT(a/256) AND 255
210 DEF FNlsb(a)=UNT(a) AND 255
220 FOR i=start TO start+&4C:READ a:sum=sum+a:POKE i,a:NEXT i
230 IF sum=8381 THEN 250 ' Pruefsumme korrekt!
240 PRINT CHR$(7);" * Fehler in den DATA-Zeilen!":PRINT:LIST 410-
250 FOR i=1 TO 4:READ a:a=a+start
260   value=PEEK(a)+PEEK(a+1)*256-40960+start
270   POKE a,FNlsb(value):POKE a+1,FNmsb(value)
280 NEXT i
290 CALL start+14 ' RSX einbinden
300 READ a$:!LINE,@a$:x=x+1:IF x<4 THEN 300
350 PRINT CHR$(24)+"SELBSTMODIFIZIERENDE PROGRAMME MIT !LINE"+CHR$(24)
360 LOCATE 10,5:PRINT CHR$(164);" Martin Kotulla 1985"
370 PRINT:PRINT STRING$(40,210):PRINT:PRINT
380 PEN 1:PRINT " Diese RSX-Erweiterung erm"+CHR$(255)+"glicht"
390 PRINT:PRINT " die Entwicklung selbstmodifizierender"
400 PRINT:PRINT " Programme, z.B.:"
410 PRINT:PRINT:PRINT:PRINT
420 PRINT " A$="+CHR$(34)+"1000 DEF FN p(x)=PEEK(x)+"CHR$(34)+": "
430 PRINT:PRINT " !LINE,@A$"
440 PRINT:PRINT:PRINT
450 DATA &05,&A0,&C3,&18,&A0,&4C,&49,&4E,&C5,&00,&00,&00,&00,&00,&01,&00
460 DATA &A0,&21,&0A,&A0,&CD,&D1,&BC,&C9,&CD,&00,&B9,&21,&A4,&AC,&11,&A4
470 DATA &AC,&01,&FE,&00,&13,&97,&77,&ED,&B0,&DD,&66,&01,&DD,&6E,&00,&4E
480 DATA &06,&00,&23,&5E,&23,&56,&EB,&11,&A4,&AC,&ED,&B0,&21,&A4,&AC,&CD
490 DATA &BC,&E6,&C9,&CD,&00,&B9,&3A,&02,&C0,&32,&64,&01,&C9
500 ' Daten zum Relokatieren des Maschinencodes *****
510 DATA 0,3,15,18
520 ' Programmzeilen, die mit LINE programmiert werden *****
530 DATA "310 MODE 1:INK 1,0:INK 0,13:INK 2,1"
540 DATA "320 PAPER 0:BORDER 10:PEN 2:SPEED INK 30,20"
550 DATA "330 LOCATE 1,1:PRINT STRING$(40,210)"
560 DATA "340 LOCATE 1,2"
570 END ' -----

```

Listing: LINE

```

100 ' ; *****
110 ' ; *
120 ' ; *      LINE - Programmierte Eingabe von Programmzeilen
130 ' ; *
140 ' ; *      NUR FUER DEN SCHNEIDER CPC-464
150 ' ; *
160 ' ; *****
170 '
180 '      ORG &A000      ; Der Basic-Loader erzeugt relokatiiblen Code
190 '
200 ' INPBUF EQU &ACA4      ; Eingabe- und Editierpuffer beim CPC-464

```



```

210 ' UROMON EQU &B900      ; KL UPPER ROM ENABLE: Basic-ROM einschalten
220 ' ENCODE EQU &E6BC      ; ROM-Routine zur Codierung einer Zeile
230 ' LOGEXT EQU &BCD1      ; KL LOG EXT bindet RSX-Routinen ein
240 '
250 ' RSXTAB DEFW NAMTAB      ; Zeiger auf Namenstabelle
260 '      JP LINE          ; Sprungvektor nach LINE
270 '
280 ' NAMTAB DEFM "LIN"      ; RSX-Name "LINE"
290 '      DEFB &C5,&00      ; Wort- und Tabellenende
300 '
310 ' SPACE DEFS &04        ; Vier Bytes fuer das Betriebssystem
320 '
330 ' INIT LD BC,RSXTAB      ; Zeiger auf Sprungtabelle
340 '      LD HL,SPACE       ; Zeiger auf den Hilfsspeicher
350 '      CALL LOGEXT       ; RSX einbinden
360 '      RET              ; Ruecksprung nach Basic
370 '
380 ' LINE CALL UROMON       ; Basic-ROM einschalten
390 '
400 ' CLRBUF LD HL,INPBUF     ; Zeiger auf Eingabepuffer
410 '      LD DE,INPBUF      ; Ebenso
420 '      LD BC,254         ; Laenge des Puffers-1
430 '      INC DE            ; DE inkrementieren
440 '      SUB A             ; Akku loeschen
450 '      LD (HL),A         ; An die erste Pufferstelle Null schreiben
460 '      LDIR             ; LDIR zum Loeschen des Puffers "missbraucht"
470 '
480 ' DESCR LD H,(IX+1)      ; Highbyte des Stringdescriptors aus CALL
490 '      LD L,(IX+0)       ; Ebenso das Lowbyte
500 '
510 ' GETLEN LD C,(HL)       ; Stringlaenge ins C-Register
520 '      LD B,0            ; Das zugehoerige B-Register loeschen
530 '
540 ' GETADR INC HL          ; Zeiger auf Stringadresse im Descriptor
550 '      LD E,(HL)         ; Lowbyte der Stringadresse lesen
560 '      INC HL            ; Zeiger auf das Highbyte der Stringadresse
570 '      LD D,(HL)         ; Highbyte der Stringadresse lesen
580 '
590 ' MOVSTR EX DE,HL        ; Ersetzt LD HL,DE
600 '      LD DE,INPBUF      ; Zeiger auf Editierpuffer
610 '      LDIR             ; String in den Puffer kopieren
620 '
630 ' ENTER LD HL,INPBUF     ; Zeiger auf den Editierpuffer
640 '      CALL ENCODE       ; ROM-Routine zur Codierung aufrufen
650 '      RET              ; Ruecksprung nach Basic
660 '
670 ' ; Hilfsprogramm zur Feststellung des Computertyps (464/664/6128)
680 '
690 ' COMPUT CALL UROMON     ; Basic-ROM einschalten
700 '      LD A,&C002         ; Modifikationsbyte des ROM-Headers lesen
710 '      LD (&0164),A      ; Im PEEKbaren RAM ablegen
720 '      RET              ; Ruecksprung nach Basic
730 '
740 '      END              ; *****

```


Uhrzeiten rund um die Welt

Dieses Programm simuliert auf den Schneider-Computern eine Weltzeituhr. Neben der Greenwich Mean Time (GMT oder UTC) werden 20 weitere Uhrzeiten von Städten und Ländern rund um den Globus angezeigt.

Nach dem Start meldet sich das Programm mit einigen Bemerkungen zur Funktionsweise. Sie können dann die aktuelle Uhrzeit eingeben – wohlgemerkt als Greenwich-Zeit, das ist MEZ-Winterzeit minus 1 Stunde bzw. MESZ-Sommerzeit minus zwei Stunden. Die Eingabe erfolgt dabei getrennt nach Stunden, Minuten und Sekunden. Unsinnige Werte werden erkannt und zurückgewiesen; die Eingabe muß dann mit einem korrekten Wert wiederholt werden.

Nach einem Tastendruck wird der Bildschirm aufgebaut: In der obersten Zeile erscheint die Anzeige der GMT/UTC-Weltzeit, auf dem übrigen Bildschirm die Ortszeiten der 20 weiteren Städte. Folgende Orte werden angezeigt:

London, Mitteleuropa, Jerusalem (Israel), Moskau (Sowjetunion), Teheran (Iran), Karachi (Pakistan), Delhi (Indien), Bangkok (Thailand), Hongkong, Tokio (Japan), Sydney (Australien), Johannesburg (Südafrika), Wellington (Neuseeland), Honolulu (Hawaii), Los Angeles, Denver, Chicago, New York (alle USA), Santiago de Chile (Chile), Rio de Janeiro (Brasilien).

Alle Ortszeiten werden in der Schriftgröße des MODE 2 ausgegeben, nur die übergeordnete Weltzeit verwendet MODE-1-Lettern.

Aus Zeitgründen erfolgt die Darstellung der Ortszeiten nur jeweils in Minutenabständen, während für die Weltzeit auch die Sekunden angegeben werden. Martin Kotulla/LM

Weltzeit UTC/GMT:		20:25:30	
London (GMT)	20:25	Sydney	6:25
Mitteleuropa	21:25	Johannesburg	22:25
Jerusalem	22:25	Wellington	8:25
Moskau	23:25	Honolulu	10:25
Teheran	23:55	Los Angeles	12:25
Karachi	1:25	Denver	13:25
Delhi	1:55	Chicago	14:25
Bangkok	3:25	New York City	15:25
Hongkong	4:25	Santiago de Chile	16:25
Tokio	5:25	Rio de Janeiro	17:25



SCHREINER



Listing: CPC-Weltzeituhr

```

100 ' *****
110 ' *
120 ' *      CPC-Weltzeituhr      *
130 ' *
140 ' *****
150 '
160 ' (C) Martin Kotulla 8.10.1985
170 '
180 ' Weltzeit-Daten *****
190 DIM city$(20),offset(20)
200 FOR i=1 TO 20:READ city$(i),offset(i):NEXT i
210 DATA London (GMT),0,Mitteleuropa,1,Jerusalem,2
220 DATA Moskau,3,Teheran,3.5,Karachi,5,Delhi,5.5
230 DATA Bangkok,7,Hongkong,8,Tokio,9,Sydney,10,Johannesburg,2
240 DATA Wellington,12,Honolulu,-10,Los Angeles,-8
250 DATA Denver,-7,Chicago,-6,New York City,-5
260 DATA Santiago de Chile,-4,Rio de Janeiro,-3
270 ' Symbole und Umlaute *****
280 SYMBOL AFTER 32:RESTORE 370
290 FOR i=128 TO 147
300 FOR j=1 TO 7:READ a$:a(j)=VAL("&X"+a$):NEXT j
310 SYMBOL i+100,a(1),a(2),a(3),a(4),a(5),a(6),a(7),0:NEXT i
320 SYMBOL 160,0,0,15,15,0,15,15,0
330 SYMBOL 161,0,0,0,0,0,0,0,0
340 SYMBOL 252,&66,&0,&66,&66,&66,&66,&3E,0
350 SYMBOL 253,&66,&0,&3C,&66,&66,&66,&3C,&0
360 SYMBOL 254,0,0,3,3,3,0,0,0
370 DATA 00111111,11110000,11110000,11110011,11111100,11110000,00111111
380 DATA 11110000,00111100,11111100,00111100,00111100,00111100,11110000
390 DATA 00000011,00001111,00000011,00000011,00000011,00000011,00111111
400 DATA 11000000,11000000,11000000,11000000,11000000,11000000,11111100
410 DATA 00001111,00111100,00000000,00001111,00111100,00111100,00111111
420 DATA 11110000,00111100,00111100,11110000,00000000,00111100,11111100
430 DATA 00001111,00111100,00000000,00000011,00000000,00111100,00001111
440 DATA 11110000,00111100,00111100,11110000,00111100,00111100,11110000
450 DATA 00000011,00001111,00111100,11110000,11111111,00000000,00000011
460 DATA 11110000,11110000,11110000,11110000,11111100,11110000,11111100
470 DATA 00111111,00111100,00111100,00111111,00000000,00111100,00001111
480 DATA 11111100,00001100,00000000,11110000,00111100,00111100,11110000
490 DATA 00001111,00111100,00111100,00111111,00111100,00111100,00001111
500 DATA 11110000,00111100,00000000,11110000,00111100,00111100,11110000
510 DATA 00111111,00111100,00000000,00000000,00000011,00000011,00000011
520 DATA 11111100,00111100,00111100,11110000,11000000,11000000,11000000
530 DATA 00001111,00111100,00111100,00001111,00111100,00111100,00001111
540 DATA 11110000,00111100,00111100,11110000,00111100,00111100,11110000
550 DATA 00001111,00111100,00111100,00001111,00000000,00111100,00001111
560 DATA 11110000,00111100,00111100,11111100,00111100,00111100,11110000
570 ' Titel und Anleitung *****
580 CLS:MODE 1
590 INK 1,0:INK 0,13:INK 2,0,3:INK 3,1
600 PAPER 0:BORDER 10:PEN 3:SPEED INK 30,20
610 LOCATE 1,1:PRINT STRING$(40,210);

```



```
620 LOCATE 1,2:PRINT CHR$(24)+SPACE$(14)+"WELTZEITUHR"+SPACE$(15)+CHR$(24)
630 LOCATE 10,5:PRINT CHR$(164);" Martin Kotulla 1985"
640 PRINT STRING$(40,210):PRINT
650 PEN 1:LOCATE 1,10:PRINT " Dieses Programm stellt auf dem"
660 PRINT:PRINT " Bildschirm eine Weltzeituhr dar."
670 ' Eingabe der Uhrzeit *****
680 PEN 3:PRINT:PRINT:PRINT" Bitte geben Sie die GMT-Uhrzeit ein!"
690 PRINT:PRINT " Diese entspricht MEZ -1 Stunde."
700 LOCATE 1,19:INPUT" Stunden? ",hour:hour=INT(hour)
710 IF hour<0 OR hour>23 THEN PRINT CHR$(7);:GOTO 700
720 LOCATE 1,21:INPUT" Minuten? ",minute:minute=INT(minute)
730 IF minute<0 OR minute>59 THEN PRINT CHR$(7);:GOTO 720
740 LOCATE 1,23:INPUT" Sekunden? ",sec:sec=INT(sec)
750 IF sec<0 OR sec>59 THEN PRINT CHR$(7);:GOTO 740
760 PEN 2
770 LOCATE 25,19:PRINT CHR$(150);STRING$(13,154);CHR$(156)
780 LOCATE 25,20:PRINT CHR$(149);SPACE$(13);CHR$(149)
790 LOCATE 25,21:PRINT CHR$(149);" Start durch ";CHR$(149)
800 LOCATE 25,22:PRINT CHR$(149);SPACE$(13);CHR$(149)
810 LOCATE 25,23:PRINT CHR$(149);" Tastendruck ";CHR$(149)
820 LOCATE 25,24:PRINT CHR$(149);SPACE$(13);CHR$(149)
830 LOCATE 25,25:PRINT CHR$(147);STRING$(13,154);CHR$(153)
840 WHILE INKEY$="" :WEND
850 ' Stadtliste ausgeben *****
860 MODE 2:PEN 1
870 FOR i=1 TO 10
880 LOCATE 10,2*i+3:PRINT city$(i);
890 NEXT i
900 FOR i=11 TO 20
910 LOCATE 44,2*i-17:PRINT city$(i);
920 NEXT i
930 ' Umrandungen der Felder zeichnen *****
940 LOCATE 17,1:PRINT CHR$(150);STRING$(44,154);CHR$(156)
950 LOCATE 6,3
960 PRINT CHR$(150);STRING$(10,154);CHR$(155);STRING$(22,154);CHR$(158);
970 PRINT STRING$(21,154);CHR$(155);STRING$(12,154);CHR$(156)
980 FOR i=4 TO 24:LOCATE 6,i:PRINT CHR$(149);
990 LOCATE 40,i:PRINT CHR$(149);:LOCATE 75,i:PRINT CHR$(149);
1000 NEXT i
1010 LOCATE 6,25
1020 PRINT CHR$(147);STRING$(33,154);CHR$(155);STRING$(34,154);CHR$(153)
1030 LOCATE 17,2:PRINT CHR$(149);" Weltzeit UTC/GMT:";
1040 ' Weltzeiten erstmalig ausgeben und Timer starten *****
1050 GOSUB 1280
1060 EVERY 50 GOSUB 1090
1070 GOTO 1070
1080 ' Hauptzeit formatiert ausgeben *****
1090 LOCATE 45,2
1100 hourx$=MID$(STR$(hour),2)
1110 IF LEN(hourx$)=1 THEN hourx$=" "+hourx$
1120 minutex$=MID$(STR$(minute),2)
1130 IF LEN(minutex$)=1 THEN minutex$="0"+minutex$
1140 secx$=MID$(STR$(sec),2)
1150 IF LEN(secx$)=1 THEN secx$="0"+secx$
1160 pt$=hourx$:GOSUB 1590:PRINT CHR$(160);CHR$(161);
```



```

1170 pt$=minute$:GOSUB 1590:PRINT CHR$(160);CHR$(161);
1180 pt$=sec$:GOSUB 1590
1190 PRINT " ";CHR$(149);
1200 DI
1210 ' Uhrzeit weiterlaufen lassen *****
1220 sec=sec+1
1230 IF sec=60 THEN sec=0:minute=minute+1
1240 IF minute=60 THEN minute=0:hour=hour+1
1250 IF hour=24 THEN hour=0
1260 IF sec<>0 THEN EI:RETURN
1270 ' Uhrzeiten der ersten 10 Staedte fortzuschreiben *****
1280 FOR I=1 TO 10
1290 hour2=INT(hour+offset(I))
1300 minute2=minute
1310 IF hour2>23 THEN hour2=hour2-24
1320 IF INT(offset(I))=offset(I) THEN 1360
1330 minute2=minute2+30
1340 IF minute2>60 THEN minute2=minute2-60:hour2=hour2+1
1350 ' Uhrzeiten ausgeben *****
1360 LOCATE 30,2*I+3
1370 hour$=MID$(STR$(hour2),2)
1380 IF LEN(hour$)=1 THEN hour$=" "+hour$
1390 minute$=MID$(STR$(minute2),2)
1400 IF LEN(minute$)=1 THEN minute$="0"+minute$
1410 PRINT hour$;" " ; " ;minute$
1420 NEXT I
1430 ' Staedte 11-20: Zeiten fortzuschreiben *****
1440 FOR I=11 TO 20
1450 hour2=INT(hour+offset(I))
1460 IF hour2>24 THEN hour2=hour2-24
1470 IF hour2<0 THEN hour2=hour2+24
1480 ' Formatierte Zeitausgabe der uebrigen Staedte *****
1490 LOCATE 66,2*I-17
1500 hour$=MID$(STR$(hour2),2)
1510 IF LEN(hour$)=1 THEN hour$=" "+hour$
1520 minute$=MID$(STR$(minute),2)
1530 IF LEN(minute$)=1 THEN minute$="0"+minute$
1540 PRINT hour$;" " ; " ;minute$
1550 NEXT I
1560 EI
1570 RETURN
1580 ' Unterprogramm: Ausgabe von PT$ in MODE 1-Lettern *****
1590 FOR Y=1 TO 2
1600 IF Y=1 THEN IF ASC(pt$)=32 THEN PRINT SPACE$(2);:GOTO 1630
1610 char=228+2*(ASC(MID$(pt$,Y))-48)
1620 PRINT CHR$(char);CHR$(char+1);
1630 NEXT Y:RETURN

```


DATA-PRO – Ein komfortables Datei- verwaltungsprogramm

Schon in Heft 4/85 der Computer-Schau und auch der ersten Ausgabe unseres ComputerSchau-Kompaktwissens über die Schneider CPC's haben wir ein Programm zur Dateiverwaltung gebracht. Aufgrund des enorm hohen Leserrücklaufes und der ebenso hohen Nachfrage nach derartigen Programmen haben wir uns entschlossen auch in dieser Ausgabe damit zu dienen.

Das schon genannte Dateiverwaltungsprogramm hatte (wie wir vielen Zeitschriften und Anrufen entnehmen konnten) alle begeistert, die es korrekt abgetippt hatten. Andererseits aber wurden wir auch noch gebeten, ob wir nicht noch diese oder jene Eigenschaft mit einbauen könnten. Sicherlich wäre dies machbar, aber wir wollen ja keine Software verkaufen, sondern Ihnen Anregungen, Tips und Kniffe mitteilen, die Sie beim Umgang mit Ihren CPC's verwerten können. In diesem Programm sind nun einige der Wünsche realisiert worden. Welches der beiden Programme für Sie nun das richtige ist, müssen Sie selbst entscheiden. Vielleicht aber passen Sie eines der beiden Programme an Ihre speziellen Wünsche an und haben dann die genau für Sie passende Dateiverwaltung. Derjenige, der Probleme beim Abtippen hat, oder die Mühe der Eingabe scheut, kann alle größeren Programme dieses Heftes auch beim Franzis-Software-Service als „Abtipphilfe“ erhalten.

Programmbeschreibung:

DATA-PRO ist ein universelles Programm zur Verwaltung von Dateien jeglicher Art. Es besitzt einen Maskengenerator zur Erstellung der Eingabe- und der Druckmaske und ist voll menügesteuert. Als Besonderheit sollen auch gleich die komfortablen Such-, Sortier- und Druckmöglichkeiten angeführt werden, die selbst bei käuflicher Software oft nicht in diesem Maße gegeben sind. Weiterhin verfügt das Programm über ein Systemmenü das es erlaubt, Bildschirmfarben zu ändern oder auch Disketten- oder Kassettenfunktionen aufzurufen.

Die Funktionen im einzelnen:

1. Nach dem Programmstart erfolgt

im Menü die Abfrage nach „neuer“ oder „bestehender“ Datei. Neu bedeutet, daß die komplette Datei erst neu definiert werden soll. Bestehend heißt, daß diese Datei bereits existiert und von Band oder Diskette geladen werden soll.

2. Soll eine neue Datei erstellt werden, so stehen folgende Auswahlmöglichkeiten zur Verfügung:

– Datei und Maske erstellen

Die neue Datei muß definiert sein, bevor man mit ihr arbeiten kann.

– Daten eingeben

Der normale Eingabemodus.

– Datei abspeichern

Dient zum Abspeichern der eingegebenen Daten.

Der letzte Punkt dieses Untermenüs ermöglicht, wenn noch keine Daten

Die Maske ist frei definierbar

eingetragen wurden auch das Abspeichern der reinen Datei- und Maskendaten.

3. Eingeben der Basisdaten und definieren einer Datei:

- Abfrage des Dateinamens
- Angabe der maximalen Werte der Datenfelder und Datensätze
- Angabe der Bezeichnungen und Länge der einzelnen Datenfelder
- Definition der Eingabemaske

Die Bezeichnungen und Längen der einzelnen Datenfelder werden später auch in der Eingabemaske mit ausgegeben. Die Festlegung des Maskenaufbaues erfolgt mittels Cursorsteuerung und wird durch die COPY-Taste fixiert.

4. Das Hauptmenü. Es besteht aus folgenden Menüpunkten:

- Daten eingeben

Hier kann ein neuer Datensatz eingegeben und dann der bereits bestehenden Datei hinzugefügt werden.

- Daten suchen

Nach Angabe des zu durchsuchenden Datenfeldes erfolgt die Abfrage des Suchbegriffes. Der Suchbegriff kann aus einem kompletten Wort oder auch aus sogenannten „Wildcards“, also den sogenannten „Asterisks“ (* = Sternchen) bestehen. Das heißt bei Eingabe von *er werden alle Einträge welche mit „er“ enden, angezeigt.

- Drucken

Dies ist auf zwei Weisen möglich. Einmal ohne eine bestimmte Druckmaske in einer vorgegebenen Form. Hierbei kann angegeben werden, welcher Datensatz gewünscht wird (Einzeldruck). Ansonsten werden alle Datensätze „angedruckt“. Zum anderen ist aber nach Erstellen einer Druckmaske auch eine formatierte Ausgabe möglich.

Statusanzeigen geben Auskunft über Dateinamen, Anzahl der Datensätze usw.

- Anzeigen der gesamten Datei.

Diese erfolgt über Menüsteuerung. Über diesen Punkt sind auch bestehende Datensätze lösch- oder änderbar.

- Ändern der Eingabemaske

Auch nach längerem Arbeiten ist dies noch möglich.

- Datei laden

Eine von DATA-PRO erstellte Datei – inklusive der Maskendaten – kann geladen werden. (Im Speicher stehende Werte werden hierdurch gelöscht!)

- Datei abspeichern

„Saven“ der Datei auf Band oder Diskette.

- Datei sortieren

Nach entsprechender Eingabe des zu sortierenden Feldes und der Feldnummer bis zu der sortiert werden soll, wird die Umsortierung

durchgeführt. Zu beachten ist, daß der Sortiervorgang beispielsweise bei einer Datei von mehr als 100 Sätzen (programmbedingt) schon eine Zeitdauer von mehr als 30 Minuten beanspruchen kann. Deshalb sollte man wirklich nur soviel sortieren lassen, wie erforderlich ist.

- Programmende

Sicherheitshalber wird hier noch abgefragt, ob abgespeichert werden soll. Anschließend erfolgt ein RESET.

- Systemmenü

Wie schon angeführt, können die Bildschirmfarben den eigenen Vorstellungen angepaßt werden. Ebenfalls möglich bei diesem Programmpunkt ist das Arbeiten mit Disketten- und Kassettenbefehlen.

Viel Erfolg beim Verwalten Ihrer Daten.

Jens Krise/LM

Listing: Dateiverwaltungsprogramm DATA-PRO

```

100 '
110 ' (c) 1985 by Jens Krise
120 '
130 OPENOUT"dummy":MEMORY HIMEM-1:CLOSEOUT
140 ON ERROR GOTO 3820
150 ON BREAK GOSUB 5020
160 '-----variablen

```



```

170 MODE 2
180 DIM t$(20),ax(20),ay(20),d$(10,20),dd$(10,20),l(20),l$(20),sort(20),dz$(10),
dx(20),dy(20),g$(20),dxa(20),dya(20)
190 FOR i=1 TO 20:ax(i)=1:ay(i)=1:NEXT
200 df=5:x=10:name$="":u=0:INK 0,0:INK 1,24:BORDER 0
210 GOSUB 5120
220 '-----startmenue
230 CLS
240 LOCATE 10,10:PRINT"<1> Neue Datei erstellen"
250 LOCATE 10,12:PRINT"<2> Bestehende Datei bearbeiten"
260 GOSUB 670
270 IF men=1 THEN 300
280 IF men=2 THEN 400
290 GOTO 260
300 '-----menue neue datei
310 CLS:CLS#1
320 LOCATE 10,9:PRINT"<1> Datei und Maske definieren"
330 LOCATE 10,11:PRINT"<2> Daten eingeben"
340 LOCATE 10,13:PRINT"<3> Datei abspeichern"
350 GOSUB 670
360 IF men=1 THEN 1290
370 IF men=2 THEN 400
380 IF men=3 THEN GOSUB 1070
390 GOTO 310
400 '-----menue best. datei
410 ON BREAK GOSUB 5020
420 ON ERROR GOTO 3820
430 CLS:CLS#1:PEN 1
440 LOCATE#2,2,7:PRINT#2,"Mode: Hauptmenue"
450 LOCATE 10,3:PRINT"<0> Daten eingeben"
460 LOCATE 10,4:PRINT"<1> Daten suchen"
470 LOCATE 10,6:PRINT"<2> Drucken"
480 LOCATE 10,7:PRINT"<3> Anzeigen gesamte Datei"
490 LOCATE 10,9:PRINT"<4> Aendern Eingabemaske"
500 LOCATE 10,11:PRINT"<5> Datei laden"
510 LOCATE 10,12:PRINT"<6> Datei abspeichern"
520 LOCATE 10,14:PRINT"<7> Sortieren Daten"
530 LOCATE 10,16:PRINT"<8> Programm beenden"
540 LOCATE 10,17:PRINT"<9> Systemmenue"
550 GOSUB 670
560 IF men=0 THEN 2030
570 IF men=1 THEN 2610
580 IF men=2 THEN 3880
590 IF men=3 THEN 1790
600 IF men=4 THEN 1570
610 IF men=5 THEN GOSUB 750
620 IF men=6 THEN GOSUB 1070
630 IF men=7 THEN 2940
640 IF men=8 THEN 3550
650 IF men=9 THEN 3610
660 GOTO 400
670 '-----eingabe bei menue
680 i$=""
690 PRINT#3,"Bitte waehlen Sie eine der Menue-Moeglichkeiten aus...>";:LINE INPUT#3,i$

```



```

700 IF i$="" THEN 690
710 i$=LEFT$(i$,1):IF ASC(i$)<48 OR ASC(i$)>57 THEN 680
720 men=VAL(i$)
730 CLS#3
740 RETURN
750 '-----datei laden
760 CLS:CLS#1
770 name$="":u=0
780 CLS#2
790 LOCATE#2,2,7:PRINT#2,"Mode: Datei laden"
800 PRINT#3,"Name der zu ladenden Datei...>";:LINE INPUT#3,la1$:la$=LEFT$(la1$,8)
)+".dat"
810 ERASE d$,sort,dx,dy,dxa,dya,g$
820 WINDOW SWAP 0,3
830 OPENIN la$
840 INPUT#9,x
850 INPUT#9,df
860 FOR i=1 TO df
870 INPUT#9,t$(i)
880 INPUT#9,ax(i)
890 INPUT#9,ay(i)
900 INPUT#9,l(i)
910 INPUT#9,l$(i)
920 NEXT
930 DIM d$(x,df),dx(df),dy(df),g$(df),dxa(df),dya(df),sort(x)
940 FOR ii=1 TO x
950 FOR i= 1 TO df
960 INPUT#9,d$(ii,i)
970 NEXT:NEXT
980 CLOSEIN
990 WINDOW SWAP 0,3
1000 name$=la1$
1010 CLS#2:CLS#3
1020 PRINT#2:PRINT#2," Datei      : "name$
1030 PRINT#2," Datensaetze:"x
1040 PRINT#2," Datenfelder:"df
1050 PRINT#2," Gesamtbytes:";:FOR aa=1 TO df:u=u+1(aa):NEXT:PRINT#2,x*u
1060 RETURN
1070 '-----datei savein
1080 CLS:CLS#1
1090 LOCATE#2,2,7:PRINT#2,"Mode: Datei abspeichern"
1100 WINDOW SWAP 0,3
1110 OPENOUT LEFT$(name$,8)+".dat"
1120 PRINT#9,x
1130 PRINT#9,df
1140 FOR i=1 TO df
1150 PRINT#9,t$(i)
1160 PRINT#9,ax(i)
1170 PRINT#9,ay(i)
1180 PRINT#9,l(i)
1190 PRINT#9,l$(i)
1200 NEXT
1210 FOR ii=1 TO x
1220 FOR i= 1 TO df
1230 PRINT#9,d$(ii,i)

```



```
1240 NEXT:NEXT
1250 CLOSEOUT
1260 WINDOW SWAP 0,3
1270 CLS#3
1280 RETURN
1290 '-----basisdaten eingeben
1300 CLS:CLS#2
1310 LOCATE#2,2,7:PRINT#2,"Mode: Basisdaten"
1320 PRINT#3,"Name der neuen Datei...>";:LINE INPUT#3,i$:name$=LEFT$((i$+"
"),8)
1330 LOCATE#2,2,2:PRINT#2,"Datei      : "name$
1340 CLS#1:CLS#3
1350 PRINT#3,"Wieviele Datensaeetze(<2000)...>";:LINE INPUT #3,x$:x=VAL(x$)
1360 PRINT#2," Datensaeetze:"x
1370 CLS#1:CLS#3
1380 PRINT#3,"Wieviele Datenfelder(<21)...>";:LINE INPUT #3,df$:df=VAL(df$)
1390 PRINT#2," Datenfelder:"df
1400 ERASE d$,sort,dx,dy,g$,dxa,dya:DIM d$(x,df),g$(df),dxa(df),dya(df),dx(df),d
y(df),sort(x)
1410 CLS#1:CLS#3
1420 LOCATE 1,3
1430 FOR i=1 TO df
1440 PRINT#3,"Bezeichnung Datenfeld"i"...>";:LINE INPUT#3,t$(i)
1450 PRINT#3,"Laenge...>";:LINE INPUT#3,l$:l(i)=VAL(l$)
1460 l$(i)="" :FOR ii=1 TO l(i):l$(i)=l$(i)+".":NEXT
1470 PRINT" Nr."i": ";t$(i);" ";l$(i)
1480 NEXT
1490 PRINT#3,"Alle Eingaben OK ?...>";GOSUB 3500:IF i$="N" THEN 1500 ELSE 1570
1500 CLS#1:PRINT#1:PRINT#1," Welche Bezeichnung soll", " geaendert werden ?"
1510 PRINT#3,"Nr...>";:LINE INPUT#3,i$:i=VAL(i$)
1520 PRINT#3,"Neue Nr."i"...>";:LINE INPUT#3,t$(i)
1530 PRINT#3,"Neue Laenge...>";:LINE INPUT#3,l$:l(i)=VAL(l$)
1540 l$(i)="" :FOR ii=1 TO l(i):l$(i)=l$(i)+".":NEXT
1550 CLS:PRINT:PRINT:FOR i=1 TO df:PRINT" Nr."i": ";t$(i);" ";l$(i):NEXT
1560 GOTO 1490
1570 '-----positionierung
1580 PRINT#2," Gesamtbytes:";:FOR aa=1 TO df:u=u+1(aa):NEXT:PRINT#2,x*u
1590 CLS
1600 FOR i=1 TO df
1610 xx=ax(i):yy=ay(i):xx1=xx:yy1=yy
1620 CLS#3:CLS#1:PRINT#1:PRINT#1," Bitte positionieren Sie", " mit den Cursor-Tas
ten in      der Eingabemaske den Platz fuer das Daten
feld"i
1630 PRINT#1:PRINT#1," COPY = Fixieren"
1640 LOCATE xx1,yy1:PRINT STRING$(LEN(t$(i))+LEN(l$(i))+2," ")
1650 LOCATE xx,yy:PRINT t$(i);" ";l$(i)
1660 IF i>1 THEN FOR ai=1 TO i-1:LOCATE ax(ai),ay(ai):PRINT t$(ai);" ";l$(ai):N
EXT
1670 xx1=xx:yy1=yy
1680 i$=INKEY$
1690 IF i$=CHR$(240) AND yy>1 THEN yy=yy-1
1700 IF i$=CHR$(241) AND yy<20 THEN yy=yy+1
1710 IF i$=CHR$(242) AND xx>1 THEN xx=xx-1
1720 IF i$=CHR$(243) AND xx<50 THEN xx=xx+1
1730 IF i$=CHR$(224) THEN ax(i)=xx:ay(i)=yy:GOTO 1750
```



```

1740 GOTO 1640
1750 PRINT#3,"Alle Eingaben OK...>";:GOSUB 3500:IF i$="N" THEN 1620
1760 PRINT#3
1770 NEXT i
1780 GOTO 300
1790 '-----anzeigen gesamte datei
1800 i=1:ia=1:ai=1:CLS:CLS#1
1810 LOCATE#2,2,7:PRINT#2,"Mode: Anzeigen Datei"
1820 LOCATE#1,1,2:PRINT#1," < = Zurueckblaettern"
1830 PRINT#1," > = Vorblaettern"
1840 PRINT#1," S = Stop"
1850 PRINT#1," A = Aendern"
1860 PRINT#1," L = Loeschen"
1870 PRINT#1," M = Zurueck ins Menue"
1880 FOR ii=1 TO df:LOCATE ax(ii),ay(ii):PRINT t$(ii);" ";d$(i,ii);:FOR aa=LEN(
d$(i,ii)) TO l(ii)-1:PRINT " ";:NEXT aa:PRINT
1890 NEXT ii
1900 LOCATE#1,3,12:PRINT#1,"Datensatz:";i;" "
1910 dat=i
1920 i$=LOWER$(INKEY$)
1930 IF i$="s" THEN ia=1
1940 IF i$="." THEN ia=0:ai=1
1950 IF i$="," THEN ia=0:ai=-1
1960 IF i$="a" AND ia=1 THEN GOSUB 2410:GOTO 1810
1970 IF i$="l" AND ia=1 THEN GOSUB 2330:GOTO 1810
1980 IF i$="m" THEN 400
1990 IF ia=0 THEN i=i+ai
2000 IF i<l THEN i=x:ai=-1
2010 IF i>x THEN i=1
2020 GOTO 1880
2030 '-----daten eingehen
2040 ii=1
2050 CLS:CLS#3:CLS#1
2060 LOCATE#2,2,7:PRINT#2,"Mode: Daten eingeben"
2070 PRINT#1:PRINT#1," Bitte geben Sie die Daten ein. Eingaben mit <Enter> a
bschliessen."
2080 FOR ai=ii TO x:IF d$(ai,1)=" " THEN ii=ai:GOTO 2110
2090 NEXT
2100 PRINT#3,"Die Datei ";name$;" ist voll !!!":FOR i=1 TO 2000:NEXT:GOTO 400
2110 LOCATE#1,2,12:PRINT#1,"Datensatz:";ai
2120 dat=ai
2130 FOR i1=1 TO df:LOCATE ax(i1),ay(i1)
2140 PRINT t$(i1);" ";l$(i1):NEXT
2150 FOR i2=1 TO df
2160 LOCATE (ax(i2)+LEN(t$(i2))+2),ay(i2):PRINT l$(i2);"<"
2170 LOCATE ax(i2),ay(i2):PRINT t$(i2);" ";:LINE INPUT;i$
2180 d$(ai,i2)=LEFT$(i$,l(i2))
2190 GOSUB 2570
2200 NEXT i2
2210 PRINT#3,"Eingaben OK...>";:GOSUB 3500:IF i$="N" THEN d$(ai,1)=" ":GOTO 2050
2220 PRINT#3
2230 CLS#1:PRINT#1:PRINT#1," Daten angenommen..."
2240 PRINT#1:PRINT#1," <ENTER> = Weitere Daten"
2250 PRINT#1," A = Aendern"
2260 PRINT#1," L = Loeschen"

```



```

2270 PRINT#1, " M = Zurueck ins Menue"
2280 i$=LOWER$(INKEY$):IF i$="" THEN 2280
2290 IF i$="m" THEN 400
2300 IF i$="a" THEN GOSUB 2410:FOR i2=1 TO df:GOSUB 2570:NEXT
2310 IF i$="l" THEN GOSUB 2330:FOR i2=1 TO df:GOSUB 2570:NEXT
2320 GOTO 2030
2330 '-----up loeschen
2340 LOCATE#2,2,7:PRINT#2,"Mode: Loeschen Datei"
2350 PRINT#3,"Wirklich loeschen...>";GOSUB 3500
2360 IF i$="n" THEN PRINT#3:RETURN
2370 FOR aa=1 TO df:d$(dat,aa)="":NEXT
2380 PRINT#3,"Datei geloescht !"
2390 FOR aa=1 TO 1000:NEXT:CLS#1:PRINT#3
2400 RETURN
2410 '-----up aendern
2420 CLS:CLS#1
2430 PRINT#1:PRINT#1," Um Daten zu aendern bitte die Feldnummer eingeben."
2440 LOCATE#2,2,7:PRINT#2,"Mode: Aendern Daten "
2450 FOR i2=1 TO df:GOSUB 2600:NEXT
2460 PRINT#3,"Welche Nr. soll geaendert werden...>";LINE INPUT#3,i$:oo=VAL(i$)
2470 IF oo>df OR oo=0 THEN 2460
2480 PRINT#3,"Neuer Text...>";LINE INPUT#3,i$:i$=LEFT$(i$,l(oo))
2490 d$(dat,oo)=i$
2500 FOR i2=1 TO df:GOSUB 2570:NEXT
2510 PRINT#3
2520 CLS#1:PRINT#1:PRINT#1," A = Weiter aendern"
2530 PRINT#1," <ENTER> = Zurueck"
2540 i$=LOWER$(INKEY$):IF i$="" THEN 2540
2550 IF i$="a" THEN 2420
2560 CLS#1:RETURN
2570 '-----up printen datensatz
2580 LOCATE ax(i2),ay(i2):PRINT t$(i2); " ";d$(dat,i2);:FOR aa=LEN(d$(dat,i2)) TO 1(i2)-1:PRINT " ";:NEXT aa:PRINT
2590 RETURN
2600 LOCATE ax(i2)+LEN(t$(i2))-3,ay(i2):PRINT i2;"> ";d$(dat,i2);:FOR aa=LEN(d$(dat,i2)) TO 1(i2)-1:PRINT " ";:NEXT aa:PRINT:RETURN
2610 '-----daten suchen
2620 CLS:CLS#1:LOCATE#2,2,7:PRINT#2,"Mode: Suchen Daten "
2630 FOR i2=1 TO df:dat=1:GOSUB 2600:NEXT
2640 PRINT#1:PRINT#1," Eingabe des zu durchsuchenden Feldes. Bei 0 werden alle Datenfelder durchsucht"
2650 PRINT#3,"Feld Nr...>";LINE INPUT#3,i$:aa=VAL(i$):xs=aa
2660 CLS#1:PRINT#1:PRINT#1," Wonach soll gesucht werden. Z.B. St* = alles mit St ... *er = alles mit ...er"
2670 PRINT#3,"Eingabe...>";LINE INPUT#3,i$:su$=i$
2680 PRINT#3,"Suchroutine ist gestartet.":FOR o=1 TO 200:NEXT
2690 su=0:IF LEFT$(su$,1)="*" THEN su=1
2700 IF su=1 THEN 2720
2710 IF su=0 THEN 2790
2720 '-----* am anfang
2730 ii=xs:FOR i=1 TO x
2740 IF xs=0 THEN FOR ii=1 TO df
2750 IF RIGHT$(d$(i,ii),LEN(su$)-1)=RIGHT$(su$,LEN(su$)-1) THEN GOSUB 2860
2760 IF xs=0 THEN NEXT ii
2770 NEXT i

```



```

2780 GOTO 2850
2790 '-----* am ende
2800 ii=xs:FOR i=1 TO x
2810 IF xs=0 THEN FOR ii=1 TO df
2820 IF LEFT$(d$(i,ii),LEN(su$)-1)=LEFT$(su$,LEN(su$)-1) THEN GOSUB 2860
2830 IF xs=0 THEN NEXT ii
2840 NEXT i
2850 PRINT#3,"Keine Daten gefunden.":FOR o= 1 TO 1000:NEXT:GOTO 400
2860 '-----ausgabemenue suchen
2870 PRINT#3,"Daten gefunden.":FOR o=1 TO 500:NEXT
2880 dat=i:FOR i2=1 TO df:GOSUB 2570:NEXT
2890 CLS#1:LOCATE#1,2,7:PRINT#1,"<Enter> = Weitersuchen", " M = Zurueck zum Menue
"
2900 i$=INKEY$:IF i$="" THEN 2900
2910 IF i$="m" THEN 400
2920 PRINT#3,"Suchroutine ist gestartet.":FOR o=1 TO 200:NEXT
2930 RETURN
2940 '-----sortieren
2950 LOCATE#2,2,7:PRINT#2,"Mode: Sortieren      "
2960 CLS:CLS#1
2970 dat=1:FOR i2=1 TO df:GOSUB 2600:NEXT
2980 PRINT#3,"Sortieren nach Feld...>";LINE INPUT#3,i$:su=VAL(i$)
2990 PRINT#3,"Bis zu welchem Datensatz soll sortiert werden...>";LINE INPUT#3,i
$:xx=VAL(i$)
3000 PRINT#3,"(A)-Aufsteigend (Z)-Absteigend sortieren...>";GOSUB 3500
3010 IF i$="A" THEN von=1
3020 IF i$="Z" THEN von=2
3030 IF i$<>"A" AND i$<>"Z" THEN 3230
3040 CLS
3050 PRINT CHR$(7):PRINT#3,"Vorsortieren gestartet."
3060 '
3070 '
3080 FOR i=1 TO xx
3090 sort(i)=0
3100 NEXT i
3110 '
3120 FOR i= 1 TO xx
3130 IF MID$(d$(i,su),1,1)<>" " THEN sort(i)=sort(i)+((ASC(MID$(d$(i,su),1,1))-31
)*4569.77)
3140 IF MID$(d$(i,su),2,1)<>" " THEN sort(i)=sort(i)+((ASC(MID$(d$(i,su),2,1))-31
)*175.77)
3150 IF MID$(d$(i,su),3,1)<>" " THEN sort(i)=sort(i)+((ASC(MID$(d$(i,su),3,1))-31
)*6.77)
3160 IF MID$(d$(i,su),4,1)<>" " THEN sort(i)=sort(i)+((ASC(MID$(d$(i,su),4,1))-31
)*0.27)
3170 IF MID$(d$(i,su),5,1)<>" " THEN sort(i)=sort(i)+((ASC(MID$(d$(i,su),5,1))-31
)*0.01)
3180 NEXT
3190 PRINT CHR$(7):PRINT#3,"Sortieren gestartet."
3200 IF von=1 THEN 3340
3210 '-----z-a
3220 h=0
3230 FOR i=1 TO xx-1 STEP 2:IF sort(i)<sort(i+1) THEN 3240 ELSE 3270
3240 FOR ur=1 TO df:hilfe$=d$(i,ur):d$(i,ur)=d$(i+1,ur):d$(i+1,ur)=hilfe$
3250 NEXT ur

```



```

3260 sort1=sort(i):sort(i)=sort(i+1):sort(i+1)=sort1:h=1
3270 NEXT i
3280 FOR i=2 TO xx-1 STEP 2:IF sort(i)<sort(i+1) THEN 3290 ELSE 3320
3290 FOR ur=1 TO df:hilfe$=d$(i,ur):d$(i,ur)=d$(i+1,ur):d$(i+1,ur)=hilfe$
3300 NEXT ur
3310 sort1=sort(i):sort(i)=sort(i+1):sort(i+1)=sort1:h=1
3320 NEXT i
3330 IF h=1 THEN 3220 ELSE 3480
3340 '-----
3350 h=0
3360 FOR i=1 TO xx-1 STEP 2:IF sort(i)>sort(i+1) THEN 3370 ELSE 3400
3370 FOR ur=1 TO df:hilfe$=d$(i,ur):d$(i,ur)=d$(i+1,ur):d$(i+1,ur)=hilfe$
3380 NEXT ur
3390 sort1=sort(i):sort(i)=sort(i+1):sort(i+1)=sort1:h=1
3400 NEXT i
3410 FOR i=2 TO xx-1 STEP 2:IF (sort(i)>sort(i+1)) THEN 3420 ELSE 3450
3420 FOR ur=1 TO df:hilfe$=d$(i,ur):d$(i,ur)=d$(i+1,ur):d$(i+1,ur)=hilfe$
3430 NEXT ur
3440 sort1=sort(i):sort(i)=sort(i+1):sort(i+1)=sort1:h=1
3450 NEXT i
3460 IF h=1 THEN 3350
3470 '
3480 PRINT CHR$(7):PRINT#3:PRINT#3,"Datei neu sortiert..."
3490 FOR i=1 TO 1000:NEXT:GOTO 400
3500 '-----input line
3510 LINE INPUT#3,i$
3520 IF i$<>" "AND VAL(i$)<32 THEN f=VAL(i$):flag=1 ELSE flag=0
3530 i$=UPPER$(LEFT$(i$,1))
3540 RETURN
3550 '-----ende
3560 PRINT#3,"Wollen Sie die Datei abspeichern...>";:GOSUB 3500
3570 IF i$="J" THEN GOSUB 1070
3580 PRINT#3,"Wollen Sie das Programm wirklich loeschen...>";:GOSUB 3500
3590 IF i$="J" THEN CALL 64000
3600 GOTO 400
3610 '-----systemmenue
3620 WINDOW#4,2,50,6,18:CLS:CLS#4
3630 WINDOW SWAP 4,3
3640 LOCATE#3,1,6:PRINT#3," <1> Disk-Menue":PRINT#3:PRINT#3," <2> Grund-Menue"
3650 PRINT#4,"Auswahl...>";:LINE INPUT#4,i$
3660 i$=LOWER$(LEFT$(i$,1))
3670 IF i$="1" THEN CLS#3:GOTO 3700
3680 IF i$="2" THEN CLS#3:GOTO 3770
3690 GOTO 3650
3700 LOCATE#3,1,5:PRINT#3," <1> Disk-Inhalt":PRINT#3:PRINT#3," <2> Umbenennen"
:PRINT#3:PRINT#3," <3> Loeschen"
3710 PRINT#4,"Auswahl...>";:LINE INPUT#4,i$
3720 i$=LOWER$(LEFT$(i$,1))
3730 IF i$="1" THEN CLS#3:WINDOW SWAP 0,3:CAT:WINDOW SWAP 0,3:PRINT#4,"Bitte Tas
te druecken...":CALL &BB06
3740 IF i$="2" THEN CLS#3:INPUT#4,"Welches Programm";o$:INPUT#4,"Neuer Name";n$
:REN,@n$,@o$
3750 IF i$="3" THEN CLS#3:INPUT#4,"Welches Programm";o$:!ERA,@o$
3760 WINDOW SWAP 4,3:GOTO 400
3770 LOCATE#3,1,5:PRINT#3," Rahmenfarbe...>";:GOSUB 3500:IF flag=1 THEN BORDER f

```



```

3780 PRINT#3," Hintergrundfarbe...>";GOSUB 3500:IF flag=1 THEN INK 0,f
3790 PRINT#3," Schreibfarbe...>";GOSUB 3500:IF flag=1 THEN INK 1,f
3800 PRINT#3," Baud-Rate (0/1)...>";GOSUB 3500:IF f=1 OR f=0 THEN SPEED WRITE f

3810 WINDOW SWAP 3,4:GOTO 400
3820 '-----error
3830 CLS#1:PRINT#1:PRINT#1," Im Programm ist ein Fehler aufgetreten.",,,," Zeile
      : "ERL
3840 PRINT#1," Fehlernummer: "ERR
3850 PRINT#3,"Programm abbrechen...>";GOSUB 3500
3860 IF i$="J" THEN 3550
3870 GOTO 400
3880 '-----ausdruck
3890 LOCATE#2,2,7:PRINT#2,"Mode: Drucken      "
3900 CLS#1:CLS
3910 LOCATE 10,9:PRINT"<1> Druckmaske festlegen"
3920 LOCATE 10,11:PRINT"<2> Drucken ohne Maske"
3930 LOCATE 10,12:PRINT"<3> Drucken mit Maske"
3940 LOCATE 10,14:PRINT"<4> Zurueck ins Menue"
3950 GOSUB 670
3960 IF men=1 THEN 4250
3970 IF men=2 THEN 4010
3980 IF men=3 THEN 4650
3990 IF men=4 THEN 400
4000 GOTO 3950
4010 '-----drucken ohne maske
4020 LOCATE#2,2,7:PRINT#2,"Mode: Drucken o Maske "
4030 CLS#1:CLS
4040 LOCATE 10,10:PRINT"<1> Drucken einzelner Datensaeetze"
4050 LOCATE 10,11:PRINT"<2> Drucken aller Datensaeetze"
4060 LOCATE 10,13:PRINT"<3> Zurueck ins Menue"
4070 GOSUB 670
4080 IF men=1 THEN 4120
4090 IF men=2 THEN 4550
4100 IF men=3 THEN 3880
4110 GOTO 4070
4120 '-----d o m einzeln
4130 CLS:PRINT#3,"Welcher Datensatz soll gedruckt werden...>";:LINE INPUT#3,i$:i
=VAL(i$):IF i>x OR i<1 THEN 4130
4140 GOSUB 4160
4150 GOTO 4010
4160 '-----ausdruck datensatz i
4170 PRINT#3,"S fuer Stop..."
4180 PRINT#8:PRINT#8,"---> Datei:"name$:PRINT#8:PRINT#8,"---> Datensatz:"i
4190 FOR ia=1 TO df
4200 i$=INKEY$:IF i$="s" THEN 4010
4210 PRINT#8,t$(ia);": ";d$(i,ia)
4220 NEXT
4230 PRINT#8
4240 RETURN
4250 '-----druckmaske
4260 ERASE dz$
4270 CLS:CLS#1:LOCATE#2,2,7:PRINT#2,"Mode: Druckmaske      "
4280 PRINT#3,"Wieviele Druckzeilen soll ein Drucksatz haben...>";:LINE INPUT#3,i
$

```



```
4290 dz=VAL(i$)
4300 PRINT#3,"Wieviele Leerzeilen zwischen den Datensätzen...>";:LINE INPUT#3,i$
4310 ldz=VAL(i$)
4320 DIM dz$(dz)
4330 GOSUB 5050
4340 PRINT#3,"Steuerung>Cursor Fixieren>Copy Ausserhalb>kein Druck."
4350 FOR i=1 TO dz
4360 LOCATE 1,i:PRINT STRING$(78,233);:dz$(i)=STRING$(78," ")
4370 NEXT
4380 FOR ia=1 TO df:dx(ia)=1:dy(ia)=12:dxa(ia)=dx(ia):dya(ia)=dy(ia):g$(ia)=LEFT$(t$(ia),1)+STRING$(1(ia)-1,">"):NEXT
4390 FOR ia=1 TO df
4400 i$=INKEY$
4410 IF i$=CHR$(&F0) AND dy(ia)>1 THEN dy(ia)=dy(ia)-1
4420 IF i$=CHR$(&F1) AND dy(ia)<12 THEN dy(ia)=dy(ia)+1
4430 IF i$=CHR$(&F2) AND dx(ia)>1 THEN dx(ia)=dx(ia)-1
4440 IF i$=CHR$(&F3) AND dx(ia)<79 THEN dx(ia)=dx(ia)+1
4450 IF i$=CHR$(&E0) THEN 4490
4460 FOR aa=1 TO df:LOCATE dxa(aa),dya(aa):PRINT STRING$(LEN(g$(aa)), " "):LOCATE dx(aa),dy(aa):PRINT g$(aa):dxa(aa)=dx(aa):dya(aa)=dy(aa):NEXT
4470 GOTO 4400
4480 NEXT ia
4490 IF dy(ia)>dz THEN dx(ia)=1:dy(ia)=12
4500 IF ia=df THEN PRINT#3,"Alle Eingaben OK...>";:GOSUB 3500:ELSE 4530
4510 IF i$="N" THEN GOTO 4250 ELSE GOSUB 5120:GOTO 3880
4520 GOTO 4510
4530 GOTO 4480
4540 GOTO 3880
4550 '-----d o m a l l e
4560 PRINT#8,"---> Datei:"name$
4570 PRINT#8
4580 FOR i=1 TO x
4590 FOR ia=1 TO df:IF d$(i,ia)<>" THEN 4610
4600 NEXT ia:PRINT#8,"---> Datensatz"i"nicht vorhanden!":PRINT#8:GOTO 4630
4610 PRINT#8,"---> Datensatz:"i
4620 GOSUB 4190
4630 NEXT i
4640 RETURN
4650 '-----druck m maske
4660 LOCATE#2,2,7:PRINT#2,"Mode: Drucken m Maske "
4670 CLS#1:CLS
4680 LOCATE 10,10:PRINT"<1> Drucken einzelner Datensätze"
4690 LOCATE 10,11:PRINT"<2> Drucken aller Datensätze"
4700 LOCATE 10,13:PRINT"<3> Zurück ins Menü"
4710 GOSUB 670
4720 IF men=1 THEN 4760
4730 IF men=2 THEN 4940
4740 IF men=3 THEN 3880
4750 GOTO 4710
4760 PRINT#3,"Welcher Datensatz soll gedruckt werden...>";:LINE INPUT#3,i$
4770 i=VAL(i$)
4780 GOSUB 4790:GOTO 4650
4790 '-----ausdruck datensatz i
```



```

4800 PRINT#3,"S fuer Stop..."
4810 FOR ai=1 TO df
4820 IF dy(ai)<>12 THEN MID$(dz$(dy(ai)),dx(ai),l(ai))=d$(i,ai)
4830 NEXT
4840 FOR i1=1 TO dz
4850 i$=INKEY$:IF i$="s" THEN 4650
4860 PRINT#8," ";dz$(i1)
4870 NEXT
4880 IF ldz>0 THEN 4890 ELSE 4920
4890 FOR i1=1 TO ldz
4900 PRINT#8
4910 NEXT
4920 FOR aa=1 TO dz:dz$(aa)=STRING$(78," "):NEXT
4930 RETURN
4940 '-----d m m alle
4950 PRINT#8
4960 FOR i=1 TO x
4970 FOR ia=1 TO df:IF d$(i,ia)<>" THEN 4990
4980 NEXT ia:GOTO 5000
4990 GOSUB 4790
5000 NEXT i
5010 GOTO 4650
5020 '-----break
5030 ON BREAK GOSUB 5040
5040 GOTO 400
5050 '-----window f druckmaske
5060 WINDOW#0,1,80,1,25
5070 LOCATE 1,2:PRINT CHR$(150)STRING$(78,154)CHR$(156)
5080 FOR i=3 TO 14:LOCATE 1,i:PRINT CHR$(149):LOCATE 80,i:PRINT CHR$(149):NEXT
5090 LOCATE 1,15:PRINT CHR$(151)STRING$(78,154)CHR$(157)
5100 WINDOW#0,2,79,3,14:CLS
5110 RETURN
5120 '-----window normal
5130 WINDOW#0,1,80,1,25
5140 LOCATE 1,1
5150 PRINT"      DATEI VERWALTUNGS-PROGRAMM      -DATA-PRO-      "CHR$(164)" 1985
      BY JENS KRIESE"
5160 PRINT CHR$(150)STRING$(49,154)CHR$(158)STRING$(28,154)CHR$(156)
5170 FOR i=3 TO 22:LOCATE 1,i:PRINT CHR$(149):LOCATE 51,i:PRINT CHR$(149):NEXT
5180 FOR i=3 TO 16:LOCATE 80,i:PRINT CHR$(149):NEXT
5190 LOCATE 51,15:PRINT CHR$(151)STRING$(28,154)CHR$(157)
5200 FOR i=17 TO 22:LOCATE 51,i:PRINT CHR$(149):LOCATE 80,i:PRINT CHR$(149):NEXT

5210 LOCATE 1,23:PRINT CHR$(151)STRING$(49,154)CHR$(155)STRING$(28,154)CHR$(157)
5220 LOCATE 1,24:PRINT CHR$(149):LOCATE 80,24:PRINT CHR$(149)
5230 LOCATE 1,25:PRINT CHR$(147)STRING$(78,154)CHR$(153);
5240 WINDOW#0,2,50,3,22:WINDOW#1,52,79,3,14:WINDOW#2,52,79,16,22:WINDOW#3,2,79,2
      4,24
5250 RETURN

```


Eine Text-Hardcopy für die Schneider-CPC's

Mit diesem Maschinenprogramm können Sie auf dem Drucker NLQ401 von Schneider oder Brother M-1009 den Textbildschirm ausdrucken. Der Ausdruck erfolgt in NLQ-Briefqualität, und die Sonderzeichen der ASCII-Codes 128 bis 255 werden ebenfalls ausgegeben!

Das Programm ist unverändert auf allen Schneider-Computern lauffähig, da keine allzu spezifischen Systemadressen verwendet werden. Nach der Initialisierung der RSX-Extension verfügt das Schneider-Basic über einen neuen Befehl:

|PRINTSCREEN

Nach Eingabe dieses Befehls wird der gesamte Bildschirminhalt auf dem angeschlossenen Drucker ausgegeben. Sofern nicht schon geschehen, schaltet das Programm den Drucker in NLQ-Briefqualität um.

Die speziell angepaßten Schneider- und Brother-Drucker besitzen trotz des 7-Bit-Druckerausgangs eine Möglichkeit, die Sonderzeichen des Computers zu Papier zu bringen:

```
PRINT #8,CHR$(27);CHR$(61);
```

Danach werden alle folgenden Zeichen als mit gesetztem siebtem Bit interpretiert. Aus dem Zeichen 33 wird also $33+128=161$. Das entsprechende Grafikzeichen erscheint dann auf dem Papier. Zurückschalten in den normalen Druckmodus kann man mit:

```
PRINT#8,CHR$(0);
```

Diese Besonderheit macht sich das Programm zunutze und stellt damit die Zeichen mit den ASCII-Codes von 32 bis 255 richtig auf dem Drucker dar.

PRINTSCREEN funktioniert in allen

drei Bildschirm-MODEs. Sie sollten lediglich sicherstellen, daß das WINDOW #0 sich über den ganzen Bildschirm erstreckt. Andernfalls kann es zu Verzerrungen beim Ausdruck kommen.

Mit CHR\$(24) invers dargestellten Zeichen sowie die ASCII-Zeichen 0 bis 31 ersetzt PRINTSCREEN automatisch durch Leerzeichen, da sie auf dem Drucker nicht darstellbar sind.

Erwähnenswert ist, daß PRINTSCREEN den Zeilenabstand mit:

```
PRINT#8,CHR$(27);CHR$(51);CHR$(25);
```

herabsetzt, was bei Bedarf mit

```
PRINT#8,CHR$(27);CHR$(51);CHR$(35);
```

wieder korrigiert werden kann.

Martin Kotulla/LM

Listing: Text-HARDCOPY

```
100 ' *****
110 ' *
120 ' * PRINTSCREEN - Hardcopy *
130 ' *
140 ' *****
150 '
160 DEF FNmsb(a)=255 AND INT(a/256)
170 DEF FNlsb(a)=255 AND UNT(a)
180 MODE 1:INK 0,13:INK 1,0:INK 2,0,3:INK 3,1
190 PAPER 0:PEN 3:BORDER 10:SPEED INK 30,20
200 LOCATE 1,1:PRINT STRING$(40,210);
210 LOCATE 1,2
220 PRINT CHR$(24)+SPACE$(14)+"PRINTSCREEN"+SPACE$(15)+CHR$(24)
230 LOCATE 10,5:PRINT CHR$(164);" Martin Kotulla 1985"
240 PRINT:PRINT STRING$(40,210):PRINT
250 PEN 1:PRINT " Dieses Programm erweitert das Basic"
260 PRINT:PRINT " um einen RSX-Befehl, mit dem der"
270 PRINT:PRINT " Textbildschirm auf einem NLQ401- oder"
280 PRINT:PRINT " M-1009-Drucker ausgegeben werden kann:"
290 PEN 3:PRINT:PRINT STRING$(40,210):PRINT
300 PEN 2:PRINT TAB(15);":PRINTSCREEN"
```



```

310 FOR i=40960 TO 41147:READ a:sum=sum+a:NEXT i
320 IF sum=20589 THEN 350
330 PRINT:PRINT CHR$(7);" * Fehler in den DATAs!":PRINT
340 LIST 440-:END
350 RESTORE:MEMORY HIMEM-190:start=HIMEM+1
360 FOR i=start TO start+&BB:READ a:POKE i,a:NEXT i
370 FOR i=1 TO 22:READ a:a=a+start
380   value=PEEK(a)+PEEK(a+1)*256-40960+start
390   POKE a,FNlsb(value):POKE a+1,FNmshb(value)
400 NEXT i
410 CALL start+32 ' RSX-Erweiterung initialisieren
420 PEN 1:END
430 ' Maschinencode-Daten *****
440 DATA &00,&00,&00,&0D,&0A,&1B,&49,&03,&1B,&33,&19,&10,&A0,&C3,&2A,&A0
450 DATA &50,&52,&49,&4E,&54,&53,&43,&52,&45,&45,&CE,&00,&00,&00,&00,&00
460 DATA &01,&0B,&A0,&21,&1C,&A0,&CD,&D1,&BC,&C9,&CD,&78,&BB,&22,&01,&A0
470 DATA &21,&01,&01,&CD,&75,&BB,&06,&08,&21,&03,&A0,&7E,&CD,&B5,&A0,&23
480 DATA &10,&F9,&3E,&0D,&CD,&B5,&A0,&3E,&0A,&CD,&B5,&A0,&3E,&1B,&CD,&B5
490 DATA &A0,&3E,&49,&CD,&B5,&A0,&3E,&03,&CD,&B5,&A0,&CD,&69,&BB,&14,&7A
500 DATA &32,&00,&A0,&CD,&60,&BB,&FE,&20,&30,&04,&3E,&20,&18,&14,&FE,&7F
510 DATA &38,&10,&F5,&3E,&1B,&CD,&B5,&A0,&3E,&3D,&CD,&B5,&A0,&F1,&CD,&B5
520 DATA &A0,&97,&CD,&B5,&A0,&3E,&09,&CD,&5A,&BB,&3A,&00,&A0,&57,&1E,&19
530 DATA &CD,&78,&BB,&7A,&3C,&BC,&20,&0A,&3E,&0D,&CD,&B5,&A0,&3E,&0A,&CD
540 DATA &B5,&A0,&E5,&B7,&ED,&52,&E1,&20,&BA,&3E,&0C,&CD,&B5,&A0,&2A,&01
550 DATA &A0,&CD,&75,&BB,&C9,&CD,&2B,&BD,&30,&FB,&C9,&00
560 ' Relokatier-Daten *****
570 DATA &0B,&0E,&21,&24,&2E,&39,&3D,&45,&4A,&4F,&54,&59,&61,&76
580 DATA &7B,&7F,&83,&8B,&9B,&A0,&AC,&AF
590 END ' *****

```

ARNOR Z80 ASSEMBLER version 1.09

Page 001

```

00002          ; *****
00003          ; *
00004          ; * PRINTSCREEN - fuer den NLQ-401/Brother M-1009 *
00005          ; *
00006          ; *****
00007
00008 A000 (A000)          ORG    &A000
00009          ; Mit Basic-Loader verschiebbar
00010
00011 A000 (BB5A)          TXTOUT EQU    &BB5A          ; TXT OUTPUT
00012 A000 (BB60)          RDCHAR EQU    &BB60          ; TXT RD CHAR
00013 A000 (BB69)          GETWIN EQU    &BB69          ; TXT GET WINDOW
00014 A000 (BB75)          CURSET EQU    &BB75          ; TXT SET CURSOR
00015 A000 (BB78)          CURGET EQU    &BB78          ; TXT GET CURSOR
00016 A000 (BCD1)          LOGEXT EQU    &BCD1          ; KL LOG EXTERNAL
00017 A000 (BD2B)          MCCHAR EQU    &BD2B          ; MC PRINT CHAR
00018 A000 (0001)          .SCRMOD DEFS  1              ; Speicher Zeilenlaenge
00019 A001 (0002)          .CURHLD DEFS  2              ; Speicher Cursorposition
00020
00021 A003 0D 0A 1B 49      .STRINI DEFB  13,10,27,73,03
                                ; CR, LF, NLQ-an
                                A007 03
00022 A008 1B 33 19      DEFB  27,51,25          ; Zeilenabstand 25 Einh.
00023
00024 A00B 10 A0          .RSXTAB DEFW  NAMTAB          ; Vektor auf Namenstabelle
00025 A00D C3 2A A0      JP      PRSCRN          ; Sprung nach PRINTSCREEN
00026

```



```

00027 A010 50 52 49 4E .NAMTAB DEFM "PRINTSCREE"
; RSX-Name PRINTSCREEN
      A014 54 53 43 52
      A018 45 45
00028 A01A ÇE          DEFB &CE          ; "N"+&80, Wortende
00029 A01B 00          DEFB &00          ; &00 Tabellenende
00030
00031 A01C (0004)      .SPACE DEFS &04
00032                      ; Hilfsspeicher zur RSX-Verwaltung
00033
00034 A020 01 0B A0     .INIT  LD  BC,RSXTAB    ; Zeiger auf RSX-Tabelle
00035 A023 21 1C A0     LD  HL,SPACE    ; Zeiger auf Hilfsspeicher
00036 A026 CD D1 BC     CALL LOGEXT     ; RSX einbinden
00037 A029 C9          RET              ; Ruecksprung nach Basic
00038
00039 A02A CD 78 BB     .PRSCRN CALL CURGET    ; Cursorposition holen
00040 A02D 22 01 A0     LD  (CURHLD),HL    ; Und sichern
00041 A030 21 01 01     LD  HL,&0101    ; Zeile=1 & Spalte=1
00042 A033 CD 75 BB     CALL CURSET     ; Cursor setzen
00043
00044 A036 06 08     .LPINI1 LD  B,8
00045                      ; 8 Bytes zur Drucker-Initialisierung
00046 A038 21 03 A0     LD  HL,STRINI    ; Zeiger auf Init-String
00047
00048 A03B 7E          .INITLP LD  A,(HL)    ; Ein Byte lesen
00049 A03C CD B5 A0     CALL PRINTER     ; Auf dem Drucker ausgeben
00050 A03F 23          INC  HL              ; Zeig. auf naechstes Byte
00051 A040 10 F9     DJNZ INITLP
00052                      ; Weitermachen, bis alles uebertragen
00053
00054 A042 3E 0D     .LPINI2 LD  A,13        ; CR-Code
00055 A044 CD B5 A0     CALL PRINTER     ; ausdrucken
00056 A047 3E 0A     LD  A,10            ; LF-Code
00057 A049 CD B5 A0     CALL PRINTER     ; ausdrucken
00058 A04C 3E 1B     LD  A,27            ; ESC-Code
00059 A04E CD B5 A0     CALL PRINTER     ; ausdrucken
00060 A051 3E 49     LD  A,73            ; 1.Byte des NLQ-Codes
00061 A053 CD B5 A0     CALL PRINTER     ; ausdrucken
00062 A056 3E 03     LD  A,3            ; 2.Byte des NLQ-Codes
00063 A058 CD B5 A0     CALL PRINTER     ; ausdrucken
00064
00065 A05B CD 69 BB     CALL GETWIN      ; Window-Daten holen
00066 A05E 14          INC  D              ; D enthaelt Zeilenlaenge
00067 A05F 7A          LD  A,D            ; Wert in den Akku
00068 A060 32 00 A0     LD  (SCRMOD),A    ; Und in SCRMOD speichern
00069
00070 A063 CD 60 BB     .RDLOOP CALL RDCHAR
00071                      ; Ein Zeichen vom Bildschirm lesen
00072
00073 A066 FE 20     .LOW  CP  32            ; Kleiner als ASCII-32?
00074 A068 30 04     JR  NC,HIGH          ; Nein/Obergrenze pruefen
00075 A06A 3E 20     LD  A,32            ; durch Space ersetzen
00076 A06C 1B 14     JR  OUTPUT          ; Und Sprung zur Ausgabe
00077
00078 A06E FE 7F     .HIGH CP  127          ; Groesser als ASCII-127?
00079 A070 38 10     JR  C,OUTPUT          ; Nein/Sprung zur Ausgabe
00080 A072 F5          PUSH AF              ; Zeichen sichern
00081 A073 3E 1B     LD  A,27            ; ESC 61 schaltet auf
00082 A075 CD B5 A0     CALL PRINTER     ; alternativen
00083 A078 3E 3D     LD  A,61            ; Zeichensatz um
00084 A07A CD B5 A0     CALL PRINTER

```



```

00085 A07D F1          POP AF
00086                ; Zeichen im Akku wiederherstellen
00087 A07E CD B5 A0     CALL PRINTER      ; Und ebenfalls ausgeben
00088 A081 97           SUB A
00089                ; Mit CHR(0) zurueck zum Normzeichensatz
00090
00091 A082 CD B5 A0     .OUTPUT CALL PRINTER      ; Buchstaben ausgeben
00092 A085 3E 09        LD A,9
00093                ; ASCII-Code fuer CURSOR-RECHTS
00094 A087 CD 5A BB     CALL TXTOUT
00095                ; An die Text-VDU schicken
00096
00097 A08A 3A 00 A0     LD A,(SCRMOD)      ; Zeilenlaenge in den Akku
00098 A08D 57           LD D,A             ; Ins D-Register
00099 A08E 1E 19        LD E,25           ; Zeile 25
00100 A090 CD 78 BB     CALL CURGET
00101                ; Tatsaechliche Cursorposition holen
00102
00103 A093 7A           .LINCHK LD A,D      ; Zeilenlaenge in den Akku
00104 A094 3C           INC A
00105                ; Wert erhoehen (physikalisch -> logisch)
00106 A095 BC           CP H
00107                ; Mit der aktuellen Spalte vergleichen
00108 A096 20 0A        JR NZ,SCRCHK
00109                ; Wenn ungleich, weitermachen
00110 A098 3E 0D        LD A,13           ; Sonst CR ausgeben
00111 A09A CD B5 A0     CALL PRINTER
00112 A09D 3E 0A        LD A,10           ; Und LF
00113 A09F CD B5 A0     CALL PRINTER
00114
00115 A0A2 E5           .SCRCHK PUSH HL      ; Vergleicht DE mit HL
00116 A0A3 B7           OR A
00117 A0A4 ED 52        SBC HL,DE
00118 A0A6 E1           POP HL
00119 A0A7 20 BA        JR NZ,RDLOOP
00120                ; Wenn Bildschirmende nicht erreicht!
00121
00122 A0A9 3E 0C        .FFFEED LD A,12      ; Form-Feed-Code
00123 A0AB CD B5 A0     CALL PRINTER      ; Wirft das Blatt aus
00124
00125 A0AE 2A 01 A0     .RETURN LD HL,(CURHLD) ; Alte Cursorpos. laden
00126 A0B1 CD 75 BB     CALL CURSET      ; Und Cursor setzen
00127 A0B4 C9           RET              ; Ruecksprung nach Basic
00128
00129 A0B5 CD 2B BD     .PRINTER CALL MCCHAR ; Druckerausgabe versuchen
00130 A0B8 30 FB        JR NC,PRINTER
00131                ; Solange versuchen, bis es klappt
00132 A0BA C9           RET
00133 A0BB (A0BB)       END                ; *****

```

Errors: 00000 Warnings: 00000

SYMBOL TABLE:

BB75 CURSET	BB78 CURGET	A001 CURHLD	A0A9 FFEED
BB69 GETWIN	A06E HIGH	A020 INIT	A03B INITLP
BCD1 LOGEXT	A036 LPINI1	A042 LPINI2	A066 LOW
A093 LINCHK	BD2B MCCHAR	A010 NAMTAB	A082 OUTPUT
A02A PRSCRN	A0B5 PRINTER	BB60 RDCHAR	A00B RSXTAB
A063 RDLOOP	A0AE RETURN	A000 SCRMOD	A003 STRINI
A01C SPACE	A0A2 SCRCHK	BB5A TXTOUT	

Speichern Sie Zeichensätze ab!

Mit dem folgenden Maschinenprogramm wird das Schneider-Basic um zwei RSX-Befehle erweitert, mit denen der verwendete Zeichensatz auf Cassette oder Diskette gespeichert und später wieder geladen werden kann.

Bei der Programmeingabe können Sie auswählen, ob Sie lieber den Quellcode des Programms eintippen und diesen dann assemblieren, oder gleich den Basic-Loader abtippen. Der Basic-Loader verschiebt das Maschinenprogramm im Speicher und legt es immer unter HIMEM ab. Somit können keine Adressierungsprobleme bei gleichzeitiger Verwendung mehrerer Maschinenprogramme auftauchen. Der DATA-Loader erkennt auch automatisch den Computertyp (CPC-446, CPC-664 oder CPC-6128) und paßt das Maschinenprogramm an den verwendeten Computer an. Sobald die RSX-Erweiterung initia-

liert ist, stehen Ihnen zwei neue Befehle zur Verfügung:

```
|CHARSAVE,$file$  
|CHARLOAD,$file$
```

Mit |CHARSAVE wird der gerade verwendete Zeichensatz abgespeichert, mit |CHARLOAD ein Zeichensatz von Diskette oder Cassette geladen. In file\$ sollten Sie den zugehörigen Dateinamen angeben. Das Programm bestimmt dabei automatisch die Lage und Länge des RAM-Zeichensatzes, sofern nicht gerade mit SYMBOL AFTER 256 der gesamte Zeichensatz ins Betriebssystem-ROM verlegt wurde. Da die Lage der Zeichenmatrizen im RAM-Speicher dynamisch vergeben wird, sich also nicht immer an der gleichen Adresse befindet – denken Sie an die Veränderungen beim Anschluß einer Diskettenstation! – legt CHARLOAD unabhängig vom Ort, wo der Zeichensatz gespeichert wurde, die Zeichentabellen beim Laden an die richtige Stelle.

Das Programm verwendet übrigens einige Routinen im Basic-ROM, die bei allen drei Schneider-Computern an verschiedenen Stellen im Speicher stehen:

GETMEM reserviert 2K-Speicher für Dateien:

```
CPC-464: CALL &F637  
CPC-664: CALL &F72A  
CPC-6128: CALL &F72A
```

RELMEM gibt diesen Speicher wieder frei:

```
CPC-464: CALL &F671  
CPC-664: CALL &F75D  
CPC-6128: CALL &F761
```

BASERR gibt eine Fehlermeldung aus, deren Code im E-Register übergeben wird:

```
CPC-464: CALL &CA94  
CPC-664: CALL &CB50  
CPC-6128: CALL &CB4D
```

Martin Kotulla/LM

Listing: CHAR-SAVE/-LOAD (Abspeichern und Einlesen von Zeichensätzen)

```
100 ' *****  
110 ' *  
120 ' *      CHARSAVE & CHARLOAD      *  
130 ' *  
140 ' *****  
150 '  
160 DEF FNmsb(a)=255 AND INT(a/256)  
170 DEF FNlsb(a)=255 AND UNT(a)  
180 SYMBOL 255,&66,&0,&78,&C,&7C,&CC,&76,&0  
190 MODE 1:INK 0,13:INK 1,0:INK 2,0,3:INK 3,1  
200 PAPER 0:PEN 3:BORDER 10:SPEED INK 30,20  
210 LOCATE 1,1:PRINT STRING$(40,210);  
220 LOCATE 1,2  
230 PRINT CHR$(24)+SPACE$(10)+"CHARSAVE & CHARLOAD"+SPACE$(11)+CHR$(24)  
240 LOCATE 10,5:PRINT CHR$(164);" Martin Kotulla 1985"
```



```

250 PRINT:PRINT STRING$(40,210):PRINT
260 PEN 1:PRINT " Dieses Programm liefert Ihnen zwei"
270 PRINT:PRINT " neue RSX-Befehle zum Speichern und"
280 PRINT:PRINT " Laden kompletter Zeichensatzes"+CHR$(255)+"tze:"
290 PEN 2:PRINT:PRINT " !CHARSAVE,@FILE$    !CHARLOAD,@FILE$"
300 PRINT:PEN 1
310 FOR i=39168 TO 39311:READ a:sum=sum+a:NEXT i
320 IF sum=17223 THEN 350
330 PRINT:PRINT CHR$(7);"* Fehler in den DATAs!":PRINT
340 LIST 580-660:END
350 RESTORE:MEMORY HIMEM-150:start=HIMEM+1
360 FOR i=start TO start+&8F:READ a:POKE i,a:NEXT i
370 FOR i=1 TO 7:READ a:a=a+start
380   value=PEEK(a)+PEEK(a+1)*256-39168+start
390   POKE a,FNlsb(value):POKE a+1,FNmsb(value)
400 NEXT i
410 CALL start+&1D ' RSX initialisieren
420 CALL start+&3A ' Computerversion feststellen
430 version=PEEK(&160) ' Modifikationsbyte auslesen
440 IF version=0 THEN END ' VERSION=0: CPC-464
450 IF version<>1 THEN 500 ' VERSION=1: CPC-664
460 POKE start+&37,&2A:POKE start+&38,&F7
470 POKE start+&76,&5D:POKE start+&77,&F7
480 POKE start+&8D,&5D:POKE start+&8E,&F7
490 POKE start+&4A,&50:POKE start+&4B,&CB:END
500 IF version<>2 THEN 550 ' VERSION=2: CPC-6128
510 POKE start+&37,&2A:POKE start+&38,&F7
520 POKE start+&76,&61:POKE start+&77,&F7
530 POKE start+&8D,&61:POKE start+&8E,&F7
540 POKE start+&4A,&4D:POKE start+&4B,&CB:END
550 PRINT:PRINT " Ich kann mit dieser Version des"
560 PRINT:PRINT " Computers nichts anfangen!":END
570 ' Maschinencode-Daten *****
580 DATA &08,&99,&C3,&4C,&99,&C3,&79,&99,&43,&48,&41,&52,&53,&41,&56,&C5
590 DATA &43,&48,&41,&52,&4C,&4F,&41,&C4,&00,&00,&00,&00,&00,&01,&00,&99
600 DATA &21,&19,&99,&CD,&D1,&BC,&C9,&CD,&00,&B9,&DD,&66,&01,&DD,&6E,&00
610 DATA &46,&23,&5E,&23,&56,&EB,&CD,&37,&F6,&C9,&CD,&00,&B9,&3A,&02,&C0
620 DATA &32,&60,&01,&C9,&1E,&05,&CD,&00,&B9,&C3,&94,&CA,&FE,&01,&20,&F4
630 DATA &CD,&27,&99,&CD,&8C,&BC,&CD,&AE,&BB,&5F,&3E,&FF,&93,&E5,&6F,&26
640 DATA &00,&29,&29,&29,&11,&08,&00,&19,&EB,&E1,&3E,&2C,&01,&00,&00,&CD
650 DATA &98,&BC,&CD,&8F,&BC,&CD,&71,&F6,&C9,&FE,&01,&20,&C7,&CD,&27,&99
660 DATA &CD,&77,&BC,&CD,&AE,&BB,&CD,&83,&BC,&CD,&7A,&BC,&CD,&71,&F6,&C9
670 ' Daten zum Relokalisieren des Maschinencodes *****
680 DATA &00,&03,&06,&1E,&21,&51,&7E
690 END ' *****

```

```

1000 ' ; *****
1010 ' ; *
1020 ' ; *          CHARSAVER - Zeichensätze speichern und laden
1030 ' ; *
1040 ' ; *****
1050 '
1060 '          ORG    &9900          ; Mit dem Loader verschiebbar!

```



```

1070 '
1080 ' OPENIN EQU &BC77 ; CAS IN OPEN
1090 ' CLOSIN EQU &BC7A ; CAS IN CLOSE
1100 ' BLKIN EQU &BC83 ; CAS IN DIRECT
1110 '
1120 ' OPNOUT EQU &BC8C ; CAS OUT OPEN
1130 ' CLSOUT EQU &BC8F ; CAS OUT CLOSE
1140 ' BLKOUT EQU &BC98 ; CAS OUT DIRECT
1150 '
1160 ' UROMON EQU &B900 ; KL U ROM ENABLE
1170 ' GETMAT EQU &BBAE ; TXT GET MATRIX TABLE
1180 ' LOGEXT EQU &BCD1 ; KL LOG EXTERNAL
1190 ' GETMEM EQU &F637 ; CAS-Puffer anlegen, 664: &F72A
1200 ' RELMEM EQU &F671 ; CAS-Puffer freigeben, 664: &F75D
1210 ' BASERR EQU &CA94 ; Fehlermeldung ausgeben
1220 '
1230 ' ; ***** RSXen initialisieren *****
1240 '
1250 ' RSXTAB DEFW NAMTAB ; Vektor auf Namenstabelle
1260 ' JP CHSAVE ; Sprung nach CHARSAVE
1270 ' JP CHLOAD ; Sprung nach CHARLOAD
1280 '
1290 ' NAMTAB DEFM "CHARSAV" ; RSX-Befehl "CHARSAVE"
1300 ' DEFB &C5 ; Wortende durch gesetztes 7.Bit
1310 ' DEFM "CHARLOA" ; RSX-Befehl "CHARLOAD"
1320 ' DEFB &C4 ; Wortende durch gesetztes 7.Bit
1330 ' DEFB &00 ; Nullbyte als Tabellenende
1340 '
1350 ' SPACE DEFS &04 ; 4 Bytes zur Verkettung der RSXen
1360 '
1370 ' INIT LD BC,RSXTAB ; Zeiger auf RSX-Sprungtabelle
1380 ' LD HL,SPACE ; Zeiger auf Hilfsspeicher
1390 ' CALL LOGEXT ; RSXen ins Betriebssystem einbinden
1400 ' RET ; Ruecksprung nach Basic
1410 '
1420 ' ; ***** SUBINIT *****
1430 '
1440 ' SUBINI CALL UROMON ; KL U ROM ENABLE
1450 '
1460 ' LD H,(IX+1) ; Highbyte des String Descriptors holen
1470 ' LD L,(IX+0) ; Lowbyte des String Descriptors holen
1480 ' LD B,(HL) ; Stringlaenge ins B-Register
1490 ' INC HL ; Descriptor-Zeiger erhoehen
1500 ' LD E,(HL) ; Lowbyte der Stringadresse lesen
1510 ' INC HL ; Descriptor-Zeiger erhoehen
1520 ' LD D,(HL) ; Highbyte der Stringadresse lesen
1530 ' EX DE,HL
1540 '
1550 ' CALL GETMEM ; I/O-Buffer reservieren lassen
1560 ' RET ; Ruecksprung zum Hauptprogramm
1570 '
1580 ' ; ***** VERSION *****
1590 '
1600 ' VSN CALL UROMON ; Basic-ROM einschalten
1610 ' LD A,&C002 ; Modifikationsbyte im ROM-Header lesen

```



```

1620 '      LD    (&0160),A      ; In den Basic-lesbaren Speicher laden
1630 '      RET                  ; Ruecksprung nach Basic
1640 '
1650 ' ; ***** ERROR: Fehlermeldung ausgeben *****
1660 '
1670 ' ERROR LD    E,5            ; Fehlercode 5 = IMPROPER ARGUMENT
1680 '      CALL UROMON           ; Basic-ROM einschalten
1690 '      JP     BASERR         ; ROM-Routine zur Fehlerausgabe
1700 '
1710 ' ; ***** CHARSAVE *****
1720 '
1730 ' CHSAVE CP    1              ; Es muss genau 1 Parameter da sein!
1740 '      JR     NZ,ERROR        ; Sonst Fehlerausgabe
1750 '
1760 '      CALL SUBINI             ; holt Dateinamen, reserviert Speicher
1770 '      CALL OPNOUT            ; Datei oeffnen
1780 '
1790 ' SVMATR CALL GETMAT           ; Start der Zeichenmatrizen holen
1800 '      LD     E,A              ; CP vorbereiten
1810 '      LD     A,255            ; SUB vorbereiten
1820 '      SUB    E                ; Akku=255-erstes Zeichen
1830 '      PUSH   HL              ; Matrix-Zeiger sichern
1840 '      LD     L,A              ; Akku ins L-Register
1850 '      LD     H,0              ; Statt LD HL,A
1860 '      ADD    HL,HL            ; HL=HL*2
1870 '      ADD    HL,HL            ; HL=HL*2
1880 '      ADD    HL,HL            ; HL=HL*2
1890 '      LD     DE,8             ; ADD vorbereiten
1900 '      ADD    HL,DE            ; Auch den letzte Matrixeintrag!
1910 '      EX     DE,HL            ; Matrixlaenge ins DE-Register
1920 '      POP    HL              ; HL wiederherstellen
1930 '
1940 ' SVCHAR LD     A,&2C          ; Dateiart fuer Kopfsatz
1950 '      LD     BC,0             ; Einsprungsadresse 0
1960 '      CALL   BLKOUT           ; CAS OUT DIRECT
1970 '
1980 ' SVCLOS CALL CLSOUT           ; Datei schliessen
1990 '      CALL   RELMEM           ; Pufferspeicher wieder freigeben
2000 '      RET                    ; Ruecksprung nach Basic
2010 '
2020 ' ; ***** CHARLOAD *****
2030 '
2040 ' CHLOAD CP    1              ; Genau 1 Parameter zu uebernehmen!
2050 '      JR     NZ,ERROR        ; Sonst Fehlermeldung
2060 '
2070 '      CALL SUBINI             ; Dateinamen-Zeiger, Puffer reservieren
2080 '      CALL OPENIN            ; Datei zum Einlesen oeffnen
2090 '      CALL GETMAT           ; Adresse der Zeichenmatrix holen
2100 '      CALL BLKIN             ; CAS IN DIRECT
2110 '      CALL CLOSIN           ; Eingabedatei schliessen
2120 '      CALL RELMEM           ; Speicher wieder freigeben
2130 '      RET                    ; Ruecksprung nach Basic
2140 '      END
2150 '
2160 ' ; *****

```


Dateien verschlüsselt

Dieses Programm für die Schneider-Computer zeigt Ihnen, wie sich Daten und Dateien verschlüsseln lassen.

Zur Programmbedienung: Nach dem Start von CODEC (Codierer-Decodierer) werden Sie nach den Namen der Ein- und Ausgabedatei gefragt. Die Benutzung eines Diskettenlaufwerkes ist hier empfohlen, da Sie bei Kassettenbetrieb dauernd die Kassette mit der Einlesedatei und die Kassette mit der Ausgabedatei wechseln müßten. Die Eingabe-Datei muß im ASCII-Format vorliegen.

Außerdem wird von Ihnen der Codeschlüssel benötigt, mit dem die Dateien codiert und decodiert wer-

den sollen. Aufgrund des verwendeten Algorithmus dürfen im Code keine CHR\$(0) auftauchen. Dieses bereitet aber keine Probleme, da das Zeichen sowieso nicht von der Tastatur aus eingegeben werden kann. Der Codeschlüssel kann beliebig lang sein. Sie sollten aber daran denken, daß Sie sich ihn merken müssen! Das Programm unterscheidet übrigens nicht zwischen Klein- und Großbuchstaben.

Weiter werden Sie gefragt, ob Sie

- (1) Codieren oder
- (2) Decodieren

wollen.

Sofort nimmt das Programm seine Arbeit auf und meldet sich kurz danach mit READY wieder zurück.

Programmfunktion

Die Codierung erfolgt nach der XOR-Methode. Das heißt, daß alle Strings zeichenweise mit dem Codeschlüssel logisch exklusiv geteilt werden. Dies hat den Vorteil, daß beim Codieren und Decodieren der gleiche Schlüssel verwendet werden kann. Ein einfaches Basic-Beispiel dafür:

```
PRINT 89 XOR 3: 90
PRINT 90 XOR 3: 89
```

„3“ ist hier der Codierschlüssel, „89“ die zu codierende Zahl.

Nun taucht aber aufgrund der Programmierung des Schneider-Betriebssystems das Problem auf, daß der ASCII-Wert 26 keinesfalls in einer Datei enthalten sein darf. Er würde als EOF-(End-of-File-)Kriterium fehlinterpretiert und das Laden abgebrochen, obwohl die Datei noch längst nicht zu Ende ist.

In normalen Dateien taucht CHR\$(26) fast nie auf, sehr wohl aber u. U. in einer codierten Datei. CODEC verwendet einen einfachen Trick: es setzt einfach bei allen Zeichen das 7. Bit, so daß aus dem Zeichen 26 das Zeichen $26+128=154$ wird. Die Einschränkung besteht nun für Sie darin, daß Sie nur den 7-Bit-ASCII-Zeichensatz verwenden dürfen. Eine wirkliche Beschränkung der Möglichkeiten ist das aber nicht: Von 0 bis 127 stehen alle genormten Zeichen, von 128 bis 255 die Sonderzeichen, die z. B. bei Textverarbeitungs-Dateien kaum verwendet werden.

M. Kotulla/LM

DATEI-CODIERER & DECODIERER

© Martin Kotulla 1985

Mit diesem Programm können Sie Dateien nach einem beliebigen Schlüssel codieren und wieder decodieren.

Bitte drücken Sie eine Taste!

Listing: CODEC (Programm zum Ver- und Entschlüsseln von Dateien)

```
100 ' *****
110 ' *
120 ' * DATEI-CODIERER & DECODIERER *
130 ' *
140 ' *****
```



```

150 '
160 MODE 1:INK 0,13:INK 1,0:INK 2,0,3:INK 3,1
170 PAPER 0:PEN 3:BORDER 10:SPEED INK 30,20
180 SYMBOL AFTER 250
190 SYMBOL 254,&66,&0,&66,&66,&66,&66,&3E,&0
200 SYMBOL 255,&66,&0,&3C,&66,&66,&66,&3C,&0
210 LOCATE 1,1:PRINT STRING$(40,210);
220 LOCATE 1,2
230 PRINT CHR$(24)+"          DATEI-CODIERER & DECODIERER          "+CHR$(24)
240 LOCATE 10,5:PRINT CHR$(164);" Martin Kotulla 1985"
250 PRINT:PRINT STRING$(40,210):PRINT
260 PEN 1:PRINT " Mit diesem Programm k"+CHR$(255)+"nnen Sie Dateien"
270 PRINT:PRINT " nach einem beliebigen Schl"+CHR$(254)+"ssel"
280 PRINT:PRINT " codieren und wieder decodieren."
290 PEN 3:PRINT:PRINT STRING$(40,210):PRINT:PEN 1
300 LOCATE 1,20:PEN 2
310 WHILE INKEY$<>"":WEND ' Tastaturpuffer loeschen
320 PRINT " Bitte dr"+CHR$(254)+"cken Sie eine Taste!"
330 WHILE INKEY$="":WEND ' Auf Tastendruck warten
340 MODE 1:PEN 3
350 PRINT:PRINT " Bitte geben Sie den Code-Schl"+CHR$(254)+"ssel an!"
360 PEN 1:PRINT:PRINT STRING$(40,210):PRINT
370 PEN 3:PRINT " ";LINE INPUT code$
380 PEN 1:PRINT:PRINT STRING$(40,210):PRINT:PEN 3
390 PRINT " Name der Eingabedatei:"PEN 1:PRINT:LINE INPUT " ",infile$
400 PEN 3:PRINT:PRINT " Name der Ausgabedatei:"PEN 1
410 PEN 1:PRINT:LINE INPUT " ",outfile$
420 OPENIN infile$
430 OPENOUT outfile$
440 PEN 1:PRINT:PRINT STRING$(40,210):PEN 3
450 code$=UPPER$(code$)
460 IF LEN(codier$)>254 THEN 480
470 codier$=codier$+MID$(code$,1,255-LEN(codier$)):GOTO 460
480 PRINT " [1] Codieren"
490 PRINT:PRINT " [2] Decodieren"
500 cd$=UPPER$(INKEY$):IF cd$="" THEN 500
510 IF cd$="1" THEN 550
520 IF cd$="2" THEN 650
530 PRINT CHR$(7);:GOTO 500
540 ' CODIERUNG *****
550 WHILE NOT EOF
560   b$="":LINE INPUT #9,a$
570   FOR i=1 TO LEN(a$)
580     b$=b$+CHR$((ASC(MID$(a$,i)) XOR ASC(MID$(codier$,i))) OR 128)
590   NEXT i
600   PRINT #9,b$
610 WEND
620 CLOSEIN:CLOSEOUT:CLEAR
630 MODE 1:PEN 1:END
640 ' DECODIERUNG *****
650 WHILE NOT EOF
660   b$="":LINE INPUT #9,a$
670   FOR i=1 TO LEN(a$)
680     b$=b$+CHR$((ASC(MID$(a$,i)) XOR ASC(MID$(codier$,i))) AND 127)
690   NEXT i
700   PRINT #9,b$
710 WEND
720 CLOSEIN:CLOSEOUT:CLEAR
730 MODE 1:PEN 1:END
740 ' *****

```


Rund um den Kalender – auf dem Schneider CPC 464

Wollen Sie wissen, an welchem Wochentag Sie geboren sind oder Ihre Freunde Geburtstag feiern, brauchen Sie eine Planungshilfe für den Urlaub oder zur Gehaltsabrechnung? Dieses Programm für den Schneider CPC berechnet einzelne Wochentage für Sie und druckt auf Wunsch einen Kalender für jedes beliebige Jahr seit der Kalenderreform von Papst Gregor XIII aus – wahlweise auf dem Bildschirm oder einem beliebigen angeschlossenen Drucker.

Programmbedienung

Nach dem Programmstart erscheint ein kurzes Menü mit drei Unterpunkten:

- Wochentag berechnen
- Kalender ausdrucken
- Programm beenden

Mit den Cursor-Up- und Cursor-Down-Tasten können Sie einen der drei Menüpunkte anwählen, worauf dieser invertiert angezeigt wird. Sobald Sie die COPY-Taste betätigen, wird dieses Unterprogramm aufgerufen.

Wochentag berechnen

Hier können Sie einen bestimmten Tag eingeben, worauf der Computer den zugehörigen Wochentag ausrechnet. Es sind folgende Eingaben nötig:

Tag im Monat? 30
Welcher Monat? 6
Welches Jahr? 1985

Nach kurzer Zeit meldet sich der Computer wieder:

Der 30. 6. 1985 war ein Sonntag.

Statt der Nummer des Monats können Sie auch den Monatsnamen angeben. Dabei wird nicht zwischen Klein- und Großschreibung unterschieden, und nur die ersten drei Buchstaben sind signifikant. Alle folgenden Namen werden z. B. als Januar interpretiert:

Januar, JANUAR, januar, JAN, jAn, JaN, JANaaa, jAN***

Die Jahreseingabe kann vierstellig erfolgen, z. B. „1985“ oder auch in der verkürzten Form „85“.

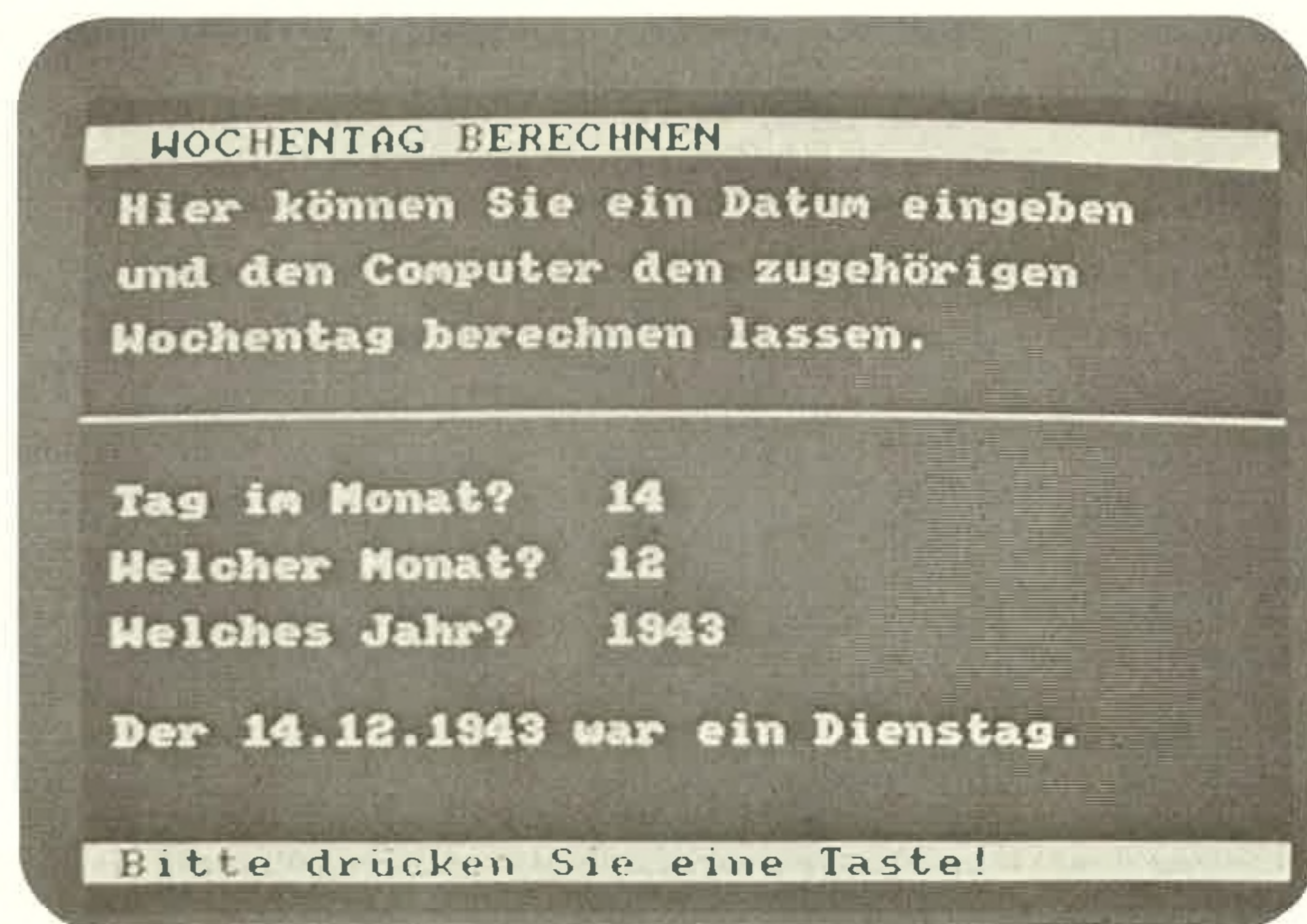
Kalender ausdrucken

Mit diesem Programmteil können Sie einen vollständigen Kalender für ein beliebiges Jahr ausdrucken lassen, was z. B. für Unternehmer bei der Buchführung recht nützlich ist.

Wahlweise erfolgt die Ausgabe auf dem Bildschirm oder auf einem Drucker. Der Druckertyp ist hierbei egal, da keinerlei spezifische Steuerzeichen verwendet werden. Der Dialog mit dem Computer kann z. B. so aussehen:

Für welches Jahr? 83
Druckerausgabe? j

Auch hier kann die Jahreszahl wieder auf zwei Stellen verkürzt werden. Bei der Frage nach dem Ausga-



begerät sind die Buchstaben „jInN“ gestattet.
Ein Kalenderausdruck für 1985 beginnt z. B. so:

Januar:
So-Mo-Di-Mi-Do-Fr-Sa
01 02 03 04 05
06 07 08 09 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30 31

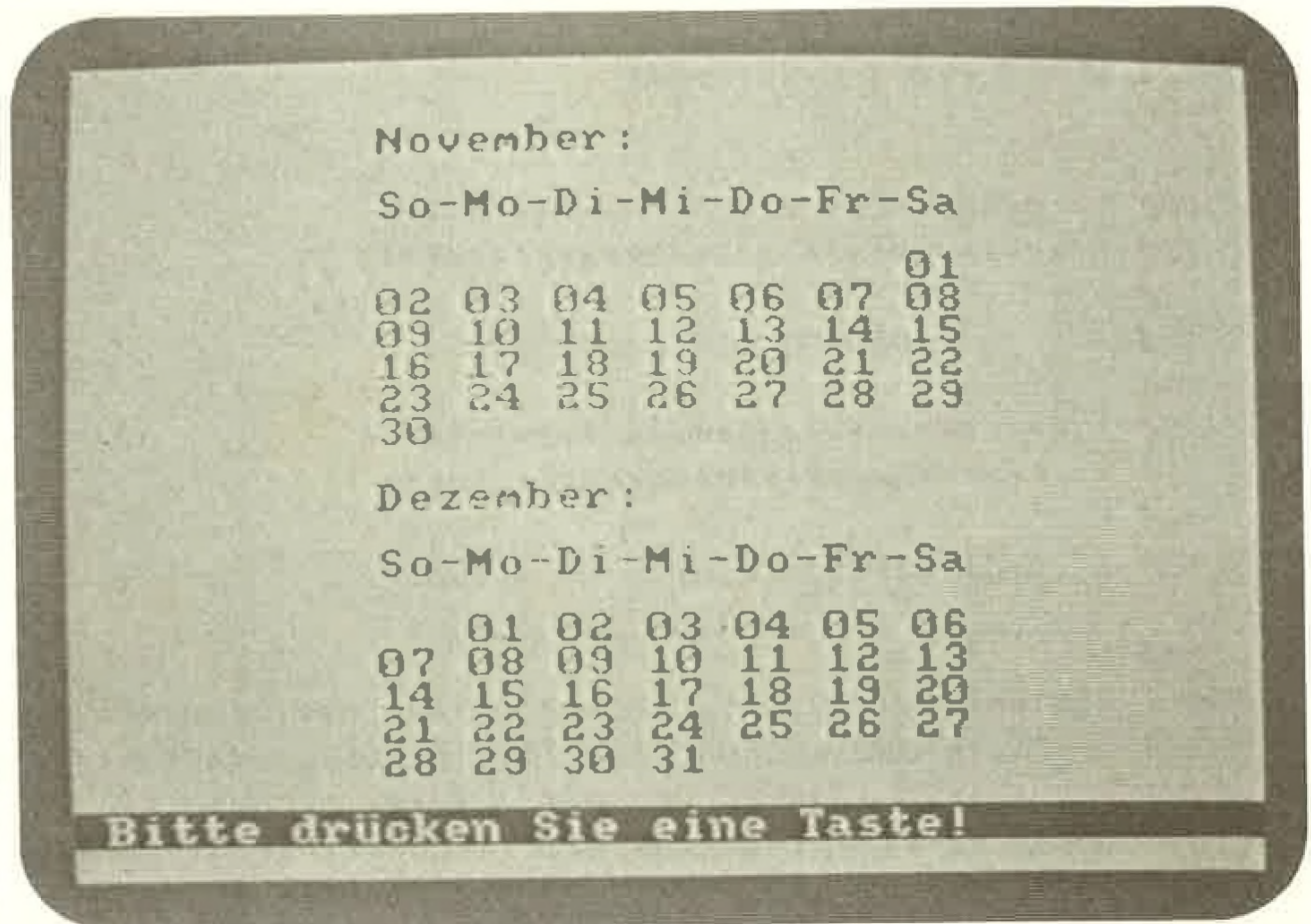
Selbstverständlich werden bei allen Datumsberechnungen eventuelle Schaltjahre berücksichtigt – genaueres dazu finden Sie unter dem Punkt „Programmaufbau“.

Programm beenden

Dazu gibt es nicht allzuviel zu sagen: Nach einer Sicherheitsabfrage wird das Programm gestoppt.

Programmaufbau

Das Programm ist modular gegliedert, so daß das Listing recht über-



sichtlich ist. Ab der Zeile 1280 beginnt das wichtigste Unterprogramm, denn hier wird aus den drei Variablen day, month und year der Wochentag berechnet und im String dayname\$ dem aufrufenden Pro-

gramm zurückgegeben. Allerdings muß nach dem Aufruf des Programms die Variable errflag geprüft werden, denn ein Wert ungleich Null zeigt an, daß die Eingangsdaten nicht korrekt waren.

Der genaue Programmaufbau:

- 1340 Korrektur der Jahreszahl: Aus 85 wird 1985
- 1350 Prüfung, ob das Jahr vor Beginn des gregorianischen Kalenders liegt.
- 1360 Prüfung auf unsinnige hohe Jahreszahlen (>4000!)
- 1370 Prüfung, ob der Monatswert gültig ist.
- 1380 Prüfung, ob der Tageswert gültig ist.
- 1400 Alle vier Jahre ist ein Schaltjahr.
- 1410 Aber nicht, wenn es ein Jahrhundert ist und nicht durch 400 teilbar.
- 1420 Prüfung auf den 27. Februar in einem Nicht-Schaltjahr.
- 1430 Prüfung auf den 28. Februar in einem Schaltjahr.
- 1450 Berechnung aller vergangenen Tage seit Christi Geburt bis Jahresanfang
- 1460 Hier werden die vergangenen Tage seit Jahresanfang addiert.
- 1470 Korrektur bei Schaltjahren um einen Tag.
- 1480 Die vergangenen Tage werden in eine Zahl von 0 bis 6 umgesetzt.
- 1490 Der Wert wird als Index des Stringfelds weekday\$() verwendet.
- Vorher findet in den Zeilen 210 bis 270 die Variablendefinition statt:
 - month\$() enthält alle Monatsnamen eines Jahres (Januar, Februar...).
 - days.in.month() speichert die Zahl der Tage für jeden Monat.
 - total.days() enthält für jeden Monat die Zahl der Tage seit Jahresanfang.
 - weekday\$() beinhaltet die Strings der Wochentage, also Montag, Dienstag ...
- Die übergeordneten Teilprogramme verwenden diese Routine und steuern den Dialog mit dem Benutzer. Das Programm kann auf folgende Art um weitere Menüpunkte erweitert werden:
 - Nach Zeile 420 Einsetzen des neuen Punktes ab ch\$(4).
 - Alle Strings müssen mit Leerzeichen auf 40 Buchstaben aufgefüllt werden.
 - Ändern Sie in Zeile 350 die Variable num; sie enthält die Zahl der Menüpunkte.
 - Erweitern Sie die ON-GOTO-Befehlszeile in Zeile 580.

So könnten Sie das Programm z. B. zur Berechnung der beweglichen Festtage erweitern.

Martin Kotulla/LM

Listing: Kalender

```

100 ' *****
110 ' *****
120 ' * *
130 ' * RUND UM DEN KALENDER *
140 ' * *
150 ' *****
160 ' *****
170 '
180 ' (C) Martin Kotulla 16.6.1985
190 '
200 '
210 DIM month$(12), days.in.month(12), total.days(12), weekday$(7):one.year=365
220 ' MONATE UND ZUGEOERIGE TAGE EINLESEN *****
230 FOR i=1 TO 12:READ month$(i), days.in.month(i)
240 total.days(i)=total.days(i-1)+days.in.month(i-1)
250 total.days(i-1)=total.days(i-1)+1:NEXT i
260 total.days(12)=total.days(12)+1
270 FOR i=0 TO 6:READ weekday$(i):NEXT i
280 MODE 1:INK 0,1:INK 1,24:INK 2,24,1:INK 3,21
290 PAPER 0:BORDER 2:SPEED INK 25,25
300 SYMBOL AFTER 252
310 SYMBOL 252,&6C,&0,&78,&C,&7C,&CC,&76,&0
320 SYMBOL 253,&0,&0,&0,&0,&0,&0,&0,&FF
330 SYMBOL 254,&66,&0,&3C,&66,&66,&66,&3C,&0
340 SYMBOL 255,&66,&0,&66,&66,&66,&66,&3E,&0
350 num=3:row=10:col=10:choice=1
360 PEN 3:LOCATE 1,1:PRINT STRING$(40,210);
370 LOCATE 1,2:PRINT CHR$(24);SPACE$(10);"RUND UM DEN KALENDER";
380 PRINT SPACE$(10);CHR$(24)
390 LOCATE 10,6:PRINT CHR$(164);" Martin Kotulla 1985"
400 ch$(1)=" WOCHENTAG BERECHNEN"+SPACE$(19)
410 ch$(2)=" KALENDER AUSDRUCKEN"+SPACE$(19)
420 ch$(3)=" PROGRAMM BEENDEN"+SPACE$(22)
430 PEN 1:LOCATE 1,row:FOR i=1 TO num:PRINT ch$(i):NEXT i
440 PEN 3:LOCATE 3,20:PRINT "Bitte w"+CHR$(252)+"hlen Sie mit ";
450 PRINT CHR$(1);CHR$(10);CHR$(1);CHR$(11);" und COPY!"
460 PEN 1:LOCATE 1,choice*2+row-3
470 PRINT STRING$(40,253);CHR$(24);ch$(choice);CHR$(24)
480 a$=INKEY$
490 IF a$=CHR$(224) THEN 580 ' Copy gedrueckt
500 IF a$="" OR (a$<>CHR$(&F0) AND a$<>CHR$(&F1)) THEN 480
510 LOCATE 1,choice*2+row-3:PRINT SPACE$(40);ch$(choice)
520 IF a$=CHR$(&F0) THEN choice=choice-1 ELSE choice=choice+1
530 IF choice<1 THEN choice=num
540 IF choice>num THEN choice=1
550 SOUND 1,1000,10
560 LOCATE 1,choice*2+row-3:PRINT STRING$(40,253);
570 PRINT CHR$(24);ch$(choice);CHR$(24):GOTO 480
580 ON choice GOTO 600,870,1190 ' In Unterprogramm springen
590 ' 1- Wochentag berechnen *****
600 PEN 1:MODE 1:PRINT STRING$(40,253);:PRINT CHR$(24);ch$(1);CHR$(24):PRINT
610 LOCATE 1,4

```



```

620 PEN 3:PRINT " Hier k"+CHR$(254)+"nnen Sie ein Datum eingeben"
630 PRINT:PRINT " und den Computer den zugeh"+CHR$(254)+"rigen"
640 PRINT:PRINT " Wochentag berechnen lassen.":PRINT
650 PEN 1:PRINT STRING$(40,253)
660 PRINT:LINE INPUT " Tag im Monat? ",day$:day=VAL(day$)
670 PRINT:LINE INPUT " Welcher Monat? ",month$:month=VAL(month$)
680 PRINT:LINE INPUT " Welches Jahr? ",year$:year=VAL(year$)
690 PRINT:PRINT
700 IF month<>0 THEN 760 ' Pruefen ob Monatseingabe z.B. "1" oder "Januar"
710 FOR i=1 TO 12
720 IF LEFT$(UPPER$(month$),3)=LEFT$(UPPER$(month$(i)),3) THEN 750
730 NEXT i ' Nicht gefunden:
740 PRINT CHR$(7);:GOTO 610
750 month$=MID$(STR$(i),2):month=i
760 GOSUB 1340 ' Tag berechnen
770 year$=MID$(STR$(year),2)
780 month$=MID$(STR$(month),2)
790 day$=MID$(STR$(day),2)
800 IF errflag=1 THEN PRINT CHR$(7);" * Fehler in der Eingabe!":GOTO 610
810 PRINT " Der "+day$+"."+month$+"."+year$+" war ein "+dayname$+"."
820 WHILE INKEY$<>"":WEND ' Tastaturpuffer loeschen
830 LOCATE 1,23:PRINT STRING$(40,253);CHR$(24);
840 PRINT " Bitte dr"+CHR$(255)+"cken Sie eine Taste!"+SPACE$(10)+CHR$(24);
850 WHILE INKEY$="":WEND:MODE 1:GOTO 360
860 ' 2- Kalender ausdrucken *****
870 PEN 1:MODE 1:PRINT STRING$(40,253);:PRINT CHR$(24);ch$(2);CHR$(24):PRINT
880 LOCATE 1,4
890 PEN 3:PRINT " Hiermit k"+CHR$(254)+"nnen Sie wahlweise auf"
900 PRINT:PRINT " dem Bildschirm oder Drucker den"
910 PRINT:PRINT " Kalender eines Jahres ausgeben."
920 PEN 1:PRINT:PRINT STRING$(40,253):PRINT
930 PRINT " F"+CHR$(255)+"r welches Jahr? ";
940 LINE INPUT year$:year=VAL(year$):IF year<100 THEN year=year+1900
950 PRINT:PRINT " Druckerausgabe? ";:PEN 2:PRINT CHR$(143):PEN 1
960 in$=UPPER$(INKEY$):IF in$="" THEN 960
970 IF in$="J" THEN dv=8:GOTO 1010
980 IF in$="N" THEN dv=0:GOTO 1000
990 PRINT CHR$(7);:GOTO 960
1000 MODE 1:WINDOW 10,34,2,24:INK 1,0:INK 0,10:BORDER 1
1010 FOR month=1 TO 12:PRINT #dv:PRINT #dv," ";month$(month);":":PRINT #dv
1020 PRINT #dv," So-Mo-Di-Mi-Do-Fr-Sa":PRINT #dv
1030 day=1:GOSUB 1340
1040 IF weekday=0 THEN PRINT #dv," 01 "; ELSE PRINT #dv,TAB(weekday*3+2);"01 ";
1050 IF month=2 AND schaltjahr=1 THEN lastday=29:GOTO 1070
1060 lastday=days.in.month(month)
1070 FOR i=2 TO lastday
1080 dy$=MID$(STR$(i),2)+" ":IF LEN(dy$)=2 THEN dy$="0"+dy$
1090 IF POS(#dv)>20 THEN PRINT #dv,TAB(2);
1100 PRINT #dv,dy$;
1110 NEXT i:PRINT #dv:NEXT month
1120 IF dv=0 THEN PRINT:PRINT:WINDOW 1,40,23,24 ELSE LOCATE 1,23
1130 WHILE INKEY$<>"":WEND ' Tastaturpuffer loeschen
1140 PRINT STRING$(40,253);CHR$(24);
1150 PRINT " Bitte dr"+CHR$(255)+"cken Sie eine Taste!"+SPACE$(10)+CHR$(24);
1160 WHILE INKEY$="":WEND ' Auf Tastendruck warten

```


Anwendungen

```
1170 MODE 1:BORDER 2:INK 0,1:INK 1,24:INK 2,24,1:INK 3,21:GOTO 360
1180 ' 3- Programm beenden *****
1190 PEN 1:MODE 1:PRINT STRING$(40,253);:PRINT CHR$(24);ch$(1);CHR$(24):PRINT
1200 LOCATE 1,4
1210 PEN 3:PRINT " Wollen Sie die Arbeit mit diesem"
1220 PRINT:PRINT " Programm wirklich beenden? ";
1230 PEN 2:PRINT CHR$(143);
1240 in$=UPPER$(INKEY$):IF in$="" THEN 1240
1250 IF in$="J" THEN MODE 1:PEN 1:BORDER 1:END
1260 IF in$<>"N" THEN PRINT CHR$(7);:GOTO 1240
1270 MODE 1:GOTO 360
1280 ' *
1290 ' * Unterprogramm: Tag ausrechnen
1300 ' *
1310 ' *
1320 ' *
1330 ' Ueberpruefung der Daten: -----
1340 IF year<100 THEN year=year+1900
1350 IF year<1584 THEN errflag=1:RETURN ELSE errflag=0
1360 IF year>4000 THEN errflag=1:RETURN
1370 IF month<1 OR month>12 THEN errflag=1:RETURN
1380 IF day<1 OR day>days.in.month(month) THEN errflag=1:RETURN
1390 ' Schaltjahr beruecksichtigen -----
1400 IF year MOD 4=0 THEN schaltjahr=1 ELSE schaltjahr=0
1410 IF year MOD 100=0 AND year MOD 400<>0 THEN schaltjahr=0
1420 IF schaltjahr=0 AND month=2 AND day>27 THEN errflag=1:RETURN
1430 IF schaltjahr=1 AND month=2 AND day>28 THEN errflag=1:RETURN
1440 ' Tag berechnen -----
1450 passed.days=one.year*year+year\4-year\100+year\400-one.year
1460 passed.days=passed.days+total.days(month)+day-1
1470 IF month<=2 AND schaltjahr=1 THEN passed.days=passed.days-1
1480 weekday=passed.days-7*INT(passed.days/7)
1490 dayname$=weekday$(weekday):RETURN
1500 ' Daten zur Wochentagsberechnung *****
1510 DATA Januar,31,Februar,28,Maerz,31,April,30,Mai,31,Juni,30,Juli,31
1520 DATA August,31,September,30,Oktober,31,November,30,Dezember,31
1530 DATA Sonntag,Montag,Dienstag,Mittwoch,Donnerstag,Freitag,Samstag
```



Schreibschutz abgefragt

Bei den Schneider-Computern mit 3-Zoll-Floppies lassen sich Disketten – wie bei vielen anderen Computern – durch Umlegen eines kleinen Hebels an der Diskette mit einem Schreibschutz versehen. Diesen Schreibschutz kann man mit dem folgenden kurzen Maschinenprogramm auch von Basic aus abfragen.

Bei der Programmeingabe müssen Sie sich entscheiden, ob Sie den Quellcode des Maschinenprogramms oder das Basic-Ladeprogramm mit den DATA-Zeilen abtippen wollen. Der Basic-Loader gibt zusätzlich einige Bedienungshinweise und erzeugt frei verschiebbaren Maschinencode und legt ihn immer genau unter HIMEM ab. Auf diese Weise kann das „Speicherplatzopfer“ auf 80 Bytes beschränkt werden.

Nach der Initialisierung der RSX-Erweiterung steht Ihnen ein neuer RSX-Befehl zur Verfügung:

|DISC.WP

„WP“ steht für „Write-protected“, also schreibgeschützt. Nach der Eingabe dieses RSX-Befehls können Sie die Speicherstelle 356 abfragen, die normalerweise vom Computer nicht benutzt wird und damit für unsere Zwecke zur Verfügung steht:

A=PEEK(356)

Wenn A=0, dann ist der Schreibschutz auf der Diskette nicht eingearastet; ein Wert von 255 sagt hingegen aus, daß der Schreibschutz gesetzt ist.

Dieses Programm läßt sich z. B. gewinnbringend in Programmen einsetzen, die die Diskettenlaufwerke benutzen. Es ist nämlich zumindest beim CPC-464 nicht möglich, Fehlermeldungen der Floppy in Basic abzufangen! Probieren Sie doch einmal ON ERROR GOTO: dieser Befehl ist bei Diskettenfehlern wirkungslos! Mit DISC.WP können Sie

wenigstens diese – besonders ärgerliche – Fehlerquelle ausschließen. Besonders für fortgeschrittene Maschinenprogrammierer dürfte die Funktionsweise des Programms interessant sein:

Wie fast alle Peripheriegeräte, die an Z80-Computer angeschlossen sind, werden auch die Floppies über die Z80-Ports verwaltet. Dies sind Ein- und Ausgabekanäle zur Kommunikation der CPU mit anderen ICs und Geräten. Genau auf diese Weise wird auch der FDC (Floppy-Disc-Controller) 765 angesteuert: Die Portadresse &FB7E verwaltet das Hauptstatus-Register des Controllers, die Adresse &FB7F das Datenregister. Werden über den Datenport die Befehlsbytes &04 und &00 ausgegeben, so liest der FDC sein Statusregister 3 aus, in dessen Bit 6

das WP-Flag steht. Der Inhalt des Registers wird auf den Datenport geschickt und kann dort mit einem IN-Befehl gelesen werden.

Ganz so einfach ist die Sache aber doch nicht: Der Floppy-Controller ist für Maschinenprogramme zu langsam! So muß zuerst festgestellt werden, ob der FDC ein „Request for Master“ schickt, eine Art Bestätigung der Empfangsbereitschaft. Außerdem muß man nach jedem Byte, das auf den Datenport gegeben wird, dem Floppy-Controller Zeit geben, die benötigten Informationen zu liefern. Am einfachsten ist es, sich in einem ROM-Listing des Floppy-Betriebssystems ab der Adresse &C95C die Hilfsroutine anzuschauen und im eigenen Programm zu verwenden!

Martin Kotulla/LM



... Sie sind also Experte für Datenfernverarbeitung?

Listing: DISC-SCHREIBSCHUTZ AUSLESEN

```

100 ' ; *****
110 ' ; *
120 ' ; *   RSX-Erweiterung zur Abfrage des Schreibschutzes von Disketten *
130 ' ; *
140 ' ; *****
150 '
160 '     ORG    &A000           ; Der Basic-Loader ist relocatibel!
170 '
180 ' PORT    EQU    &FB7E       ; Status-Port des FDC 765
190 ' MARK    EQU    &0164       ; Dort wird der WP-Status gespeichert
200 ' LOGEXT   EQU    &BCD1       ; KL LOG EXTERNAL zum Einbinden von RSXs
210 '
220 ' RSXTAB   DEFW   NAMTAB       ; Zeiger auf Namenstabelle
230 '         JP     WPTEST       ; Sprung in die Maschinenroutine
240 '
250 ' NAMTAB   DEFM   "DISC.W"     ; Befehlsname DISC.WP
260 '         DEFB   &D0,&00       ; Ende des Wortes und der Tabelle
270 '
280 ' SPACE    DEFS   &04         ; Vier Bytes Hilfsspeicher fuer Kernel-ROM
290 '
300 ' INIT     LD     BC,RSXTAB    ; Zeiger auf Sprungtabelle
310 '         LD     HL,SPACE      ; Zeiger auf Hilfsspeicher fuers Betriebssystem
320 '         JP     LOGEXT       ; Erweiterung einbinden, Ruecksprung nach Basic
330 '
340 ' WPTEST   LD     BC,PORT      ; Zeiger auf Status-Port des FDC
350 '         LD     A,&04         ; Befehlsword des FDC fuer "Sense Drive Status"
360 '         CALL   FDC          ; An den FDC ausgeben
370 '         SUB    A            ; Zweites Befehlsword fuer "Sense Drive Status"
380 '         CALL   FDC          ; Ebenso an den FDC ausgeben
390 '
400 ' WAIT     LD     A,8         ; Kurze Warteschleife, bis der FDC
410 '
420 ' WAITLP   DEC    A           ; den Wert des Statusregisters ST3
430 '         NOP                ; geliefert hat
440 '         JR     NZ,WAITLP
450 '
460 '         INC    BC           ; Zeiger auf Datenport des FDC
470 '         IN     A,(C)        ; Wert des Statusregisters ST3 auslesen
480 '         BIT    6,A          ; Das 6.Bit enthaelt das WP-Flag
490 '         LD     A,0          ; Flag besetzen mit Null, wegen Flags kein SUB A
500 '         JR     Z,RETURN     ; Wenn nicht WP, sofort Ruecksprung
510 '         XOR    255          ; Sonst Akku mit &FF laden
520 '
530 ' RETURN   LD     (MARK),A     ; In MARK das Flag zur Basic-Abfrage speichern
540 '         RET                ; Ruecksprung nach Basic
550 '
560 ' FDC       PUSH   AF          ; Auszugebenden Akkuwert sichern
570 '
580 ' HOLD      IN     A,(C)       ; Statusregister des FDC einlesen
590 '         ADD    A,A          ; Bit 7 ins Carry-Flag schieben
600 '         JR     NC,HOLD      ; Warten, solange nicht REQUEST FOR MASTER
610 '
620 ' OUTPUT   POP     AF          ; Akkuwert wiederherstellen
630 '         INC    BC           ; Zeiger auf Datenport des FDC 765
640 '         OUT    (C),A        ; Akkuinhalt ausgeben

```



```

650 '      DEC C      ; Wieder Zeiger auf den Statusport des FDC
660 '      LD A,5     ; Kurze Warteschleife
670 '
680 ' LOOP DEC A
690 '      NOP
700 '      JR NZ,LOOP
710 '      RET        ; Ruecksprung zum aufrufenden Programm
720 '      END        ; *****

```

Listing: DISC-SCHREIBSCHUTZ AUSLESEN

```

100 ' *****
110 ' * *
120 ' * DISC-SCHREIBSCHUTZ AUSLESEN *
130 ' * *
140 ' *****
150 '
160 SYMBOL AFTER 255:SYMBOL 255,&66,&0,&66,&66,&66,&66,&3E,&0
170 MEMORY HIMEM-&50:start=HIMEM+1
180 DEF FNmsb(a)=INT(a/256) AND 255
190 DEF FNlsb(a)=UNT(a) AND 255
200 FOR i=start TO start+&4C:READ a:sum=sum+a:POKE i,a:NEXT i
210 IF sum=7984 THEN 230 ' Pruefsumme korrekt!
220 PRINT CHR$(7);" * Fehler in den DATA-Zeilen!":PRINT:LIST 450-
230 FOR i=1 TO 6:READ a:a=a-&A000+start
240 value=PEEK(a)+PEEK(a+1)*256-40960+start
250 POKE a,FNlsb(value):POKE a+1,FNmsb(value)
260 NEXT i
270 CALL start+&11 ' RSX-Erweiterung einbinden
280 MODE 1:INK 0,1:INK 1,24:INK 2,6,16:INK 3,21
290 PAPER 0:PEN 3:BORDER 2:SPEED INK 30,20
300 LOCATE 1,1:PRINT STRING$(40,210);
310 LOCATE 1,2
320 PRINT CHR$(24)+ " DISKETTEN-SCHREIBSCHUTZ AUSLESEN "+CHR$(24)
330 LOCATE 10,5:PRINT CHR$(164)+ " Martin Kotulla 1985"
340 PRINT STRING$(40,210)
350 PEN 1:PRINT " Dieses Programm bietet einen neuen"
360 PRINT:PRINT " Befehl, mit dem bestimmt werden kann,"
370 PRINT:PRINT " ob die Diskette im Laufwerk A schreib-"
380 PRINT:PRINT " gesch"+CHR$(255)+"tzt ist oder nicht."
390 PRINT STRING$(40,210);:PRINT
400 PEN 2:PRINT:PRINT TAB(9);"!DISC.WP:PRINT PEEK(356)"
410 PEN 3:PRINT:PRINT " Z.B. ist die eingelegte Diskette":PRINT
420 !DISC.WP:IF PEEK(356)=0 THEN PRINT " ohne Schreibschutz!":PRINT:END
430 PRINT " mit einem Schreibschutz versehen!":PRINT
440 ' Maschinencode *****
450 DATA &05,&A0,&C3,&1A,&A0,&44,&49,&53,&43,&2E,&57,&D0,&00,&20,&27,&0D
460 DATA &0A,&01,&00,&A0,&21,&0D,&A0,&C3,&D1,&BC,&01,&7E,&FB,&3E,&04,&CD
470 DATA &3B,&A0,&97,&CD,&3B,&A0,&3E,&08,&3D,&00,&20,&FC,&03,&ED,&78,&CB
480 DATA &77,&3E,&00,&28,&02,&EE,&FF,&32,&64,&01,&C9,&F5,&ED,&78,&87,&30
490 DATA &FB,&F1,&03,&ED,&79,&0D,&3E,&05,&3D,&00,&20,&FC,&C9
500 ' Daten zum Verschieben des Maschinencodes *****
510 DATA &a000,&a003,&a012,&a015,&a020,&a024
520 END ' -----

```


Disc-Backup auf Kasette

Mit diesem Programm, das auf allen Schneider-CPC's lauffähig ist, können Sie komplette Disketten auf Kasette speichern.

Dieses Programm lohnt sich, solange die 3-Zoll-Disketten noch so teuer sind. Eine C60-Kassette, auf der sich eine komplette Diskette speichern läßt, kostet etwa 3 Mark – verglichen mit dem Diskettenpreis ein unschlagbarer Kostenvorteil.

Die Eingabe dürfte keine Schwierigkeiten bereiten, deshalb gleich zur Bedienung von CBACKUP:

Nach dem Programmstart werden einige allgemeine Informationen zum Programm ausgegeben und Sie werden aufgefordert, die zu lesende oder zu beschreibende Diskette ins Laufwerk A einzulegen. Das Programm „verkraftet“ CP/M-, Vendor- und Data-Only-Disketten. Lediglich das „exotische“ IBM-Format wird nicht verarbeitet. Eine zu „ladende“ Diskette sollte aber im gleichen Format formatiert sein, wie die, die auf Kasette gespeichert wurde, da alle Formate die System- und Katalogspuren unterschiedlich verwalten. Nach einem Tastendruck geht es weiter im Programm: Sie haben die Wahl zwischen

- (1) Diskette abspeichern und
- (2) Kasette einlesen.

Diskette abspeichern

Hier werden Sie gefragt, ob die Kassettenmeldungen angezeigt werden sollen oder nicht. Geben Sie „J“, „j“, „y“ oder „Y“ für Ja oder Yes ein, so müssen Sie nach jeweils 45 gespeicherten Tracks die Meldung „PRESS REC AND PLAY THEN ANY KEY“ abwarten und eine Taste drücken. Dies ist nur dann sinnvoll, wenn Sie kürzere als C60-Kassetten verwenden und diese nach einiger Zeit umdrehen müssen.

Bei einer Eingabe von „N“ oder „n“ für Nein läuft das Programm automatisch ab und gibt nur von Zeit zu Zeit „Reading“ mit den Tracknummern und „Writing“ aus. Sie können das Programm somit unbeaufsichtigt laufen lassen.

Die nächste Wahlmöglichkeit sieht so aus:

- (0) SPEED WRITE 0
- (1) SPEED WRITE 1

Hiermit wird die Geschwindigkeit der Datenaufzeichnung auf dem Kassettenrecorder festgelegt. Entsprechend den Basic-Befehlen steht SPEED WRITE 0 für 1000 Baud und SPEED WRITE 1 für 2000 Baud. Bei der höheren Geschwindigkeit dauert das Sichern einer Diskettenseite etwa eine halbe Stunde.

Kasette einlesen

Mit diesem Unterprogramm lassen sich auf Kasette gespeicherte Diskettenseiten wieder auf eine Diskette zurückschreiben. Bei der Frage „Cassettenmeldungen anzeigen?“ gilt das gleiche wie oben.

Sobald Sie diese Frage beantwortet haben, beginnt der Computer mit dem Laden von Kasette. Selbstverständlich muß dann die PLAY-Taste gedrückt werden.

Wenn die Daten übertragen sind, erscheint eine entsprechende Meldung und Sie werden gefragt, ob noch eine Diskette geladen oder gespeichert werden soll. Der Computer springt bei „N“ für Nein nach Basic zurück oder startet bei „J“ für Ja das Backup-Programm erneut.

Noch einige Hinweise: Verwenden Sie bitte nur Qualitätskassetten. Bei „billigen“ Kassetten ist die Gefahr von Dropouts (Aussetzer) zu hoch und würde ggf. Ihre Sicherheitskopie unbrauchbar machen. Nehmen Sie aber nur Ferro-Bänder; Sie tun Ihrem Computer mit Chrom-Kassetten keinen Gefallen! Auch lassen sich von der Backup-Kassette keine Programme direkt laden, die Daten müssen zuerst wieder auf eine „richtige“ Diskette überspielt werden, um die Programme lauffähig zu machen. M. Kotulla/LM

ARNOR Z80 ASSEMBLER version 1.09

Page 001

```

00002      ; *****
00003      ; *
00004      ; *      Quellcode zum Maschinenspracheteil *
00005      ; *                      von CBACKUP          *
00006      ; *
00007      ; *****
00008
00009      A000      (A000)          ORG      &A000
00010
00011      A000      (BCD4)          FINDCM EQU      &BCD4          ; KL FIND COMMAND

```



```

00012 A000 84          CMD84  DEFB  &84          ; RSX-Read Sector = &84
00013 A001 85          CMD85  DEFB  &85          ; RSX-Write Sector = &85
00014
00015 A002 (0002)      RDJMP   DEFS  2          ; Speicher fuer Sprung-
00016 A004 (0001)      RDR0M   DEFS  1          ; Adr. und ROM-States,
00017 A005 (0002)      WTJMP   DEFS  2          ; die von KL FIND COMMAND
00018 A007 (0001)      WTROM   DEFS  1          ; geliefert werden
00019
00020                ; ***** Initialisierung der RSX-Spruenge *****
00021
00022 A008 21 00 A0      INIT    LD     HL,CMD84      ; Zeiger auf RSX-Namen
00023 A00B CD D4 BC      CALL    FINDCM             ; KL FIND COMMAND
00024 A00E 22 02 A0      LD      (RDJMP),HL
00025                ; Adresse fuer RST-Sprung speichern
00026 A011 79           LD      A,C                ; ROM-Status in den Akku
00027 A012 32 04 A0      LD      (RDR0M),A        ; mit der Adr. speichern
00028
00029 A015 21 01 A0      LD      HL,CMD85          ; Zeiger auf RSX-Namen
00030 A018 CD D4 BC      CALL    FINDCM             ; KL FIND COMMAND
00031 A01B 22 05 A0      LD      (WTJMP),HL
00032                ; Adresse fuer RST-Sprung speichern
00033 A01E 79           LD      A,C                ; ROM-Status in den Akku
00034 A01F 32 07 A0      LD      (WTROM),A
00035                ; Und mit der Adresse speichern
00036 A022 C9           RET                        ; Ruecksprung nach Basic
00037
00038                ; ***** 45 Sektoren lesen *****
00039
00040 A023 1E 00      .READ  LD      E,0              ; Laufwerk A=0
00041 A025 06 2D      LD      B,45              ; 45 Sektoren laden
00042 A027 0E 41      LD      C,&41              ; Sektornummer
00043 A029 16 00      LD      D,0                ; Tracknummer
00044 A02B 21 68 42      LD      HL,17000
00045                ; Startadr. des Sektorpuffers
00046
00047 A02E DF      RDLOOP  RST     &18              ; Sektor einlesen
00048 A02F 02 A0      DEFW    RDJMP              ; Zeiger auf Sprungvektor
00049
00050 A031 79      RDNEXT  LD      A,C              ; Sektornummer in den Akku
00051 A032 0C      INC     C                      ; Zeiger naechster Sektor
00052 A033 FE 49      CP      &49              ; War der letzte Sektor #9
00053 A035 38 03      JR      C,RDCONT
00054                ; Nein - keinen neuen Track anfangen
00055
00056 A037 0E 41      RDXCHG LD      C,&41          ; Zeiger auf Sektor 1 ...
00057 A039 14      INC     D                      ; ... des naechsten Tracks
00058
00059 A03A 24      RDCONT  INC     H              ; Bufferzeiger um
00060 A03B 24      INC     H              ; (2*256 Bytes) erhoehen
00061 A03C 10 F0      DJNZ   RDLOOP
00062                ; Weitermachen, bis alle Sektoren gelesen

```



```

00063  A03E  C9                      RET          ; Ruecksprung nach Basic
00064
00065                      ; ***** 45 Sektoren schreiben *****
00066
00067  A03F  1E 00      .WRITE  LD      E,0      ; Laufwerk A=0
00068  A041  06 2D      LD      B,45      ; 45 Sektoren schreiben
00069  A043  0E 41      LD      C,&41      ; Sektornummer
00070  A045  16 00      LD      D,0      ; Tracknummer
00071  A047  21 68 42  LD      HL,17000    ; Startadr. Sektorpuffer
00072
00073  A04A  DF          WTLOOP RST      &18
00074                      ; RSX-Routine WRITE SECTOR aufrufen
00075  A04B  05 A0      DEFW  WTJMP      ; Zeiger auf Sprungvektor
00076
00077  A04D  79          WTNEXT LD      A,C      ; Sektornummer in den Akku
00078  A04E  0C          INC      C      ; Zeiger auf n. Sektor
00079  A04F  FE 49      CP      &49      ; War letzter Sektor #9?
00080  A051  38 03      JR      C,WRCONT
00081                      ; Nein,keinen neuen Track anfangen
00082
00083  A053  0E 41      WTXCHG LD      C,&41      ; Zeiger auf Sektor 1 ...
00084  A055  14          INC      D      ; ... des naechsten Tracks
00085
00086  A056  24          WRCONT INC      H
00087                      ; Bufferzeiger um 512 Bytes erhoehen
00088  A057  24          INC      H      ; (2*256 Bytes)
00089  A058  10 F0      DJNZ  WTLOOP
00090                      ; Weitermachen, bis alle Sektoren geschrieben
00091  A05A  C9          RET          ; Ruecksprung nach Basic
00092  A05B  (A05B)    END          ; *****

```

Errors: 00000 Warnings: 00000

SYMBOL TABLE:

A000 CMD84	A001 CMD85	BCD4 FINDCM	A008 INIT
A002 RDJMP	A004 RDRDM	A023 READ	A02E RDLOOP
A031 RDNEXT	A037 RDXCHG	A03A RDCONT	A005 WTJMP
A007 WTROM	A03F WRITE	A04A WTLOOP	A04D WTNEXT
A053 WTXCHG	A056 WRCONT		

Listing: DISC to TAPE (Ein BACKUP-PROGRAMM)

```

100 ' *****
110 ' * Disc-BACKUP auf Cassetten *
120 ' *****
130 '

```



```

140 SYMBOL AFTER 250:MEMORY 16999
150 FOR i=40960 TO 41050:READ a:POKE i,a:NEXT i
160 SYMBOL 254,&66,&0,&66,&66,&66,&3E,&0
170 SYMBOL 255,&66,&0,&3C,&66,&66,&66,&3C,&0
180 ' Maschinencode-Daten: 45 Disc-Sektoren laden und speichern *****
190 DATA &84,&85,&00,&00,&00,&00,&00,&00,&21,&00,&A0,&CD,&D4,&BC,&22,&02
200 DATA &A0,&79,&32,&04,&A0,&21,&01,&A0,&CD,&D4,&BC,&22,&05,&A0,&79,&32
210 DATA &07,&A0,&C9,&1E,&00,&06,&2D,&0E,&41,&16,&00,&21,&68,&42,&DF,&02
220 DATA &A0,&79,&0C,&FE,&49,&38,&03,&0E,&41,&14,&24,&24,&10,&F0,&C9,&1E
230 DATA &00,&06,&2D,&0E,&41,&16,&00,&21,&68,&42,&DF,&05,&A0,&79,&0C,&FE
240 DATA &49,&38,&03,&0E,&41,&14,&24,&24,&10,&F0,&C9
250 ' Titelbild und Anleitung *****
260 MODE 1:INK 0,13:INK 1,0:INK 2,0,3:INK 3,1
270 PAPER 0:PEN 3:BORDER 10:SPEED INK 30,20
280 LOCATE 1,1:PRINT STRING$(40,210);
290 LOCATE 1,2
300 PRINT CHR$(24)+SPACE$(8)+"DISC-BACKUP AUF CASSETTE"+SPACE$(8)+CHR$(24)
310 LOCATE 10,5:PRINT CHR$(164);" Martin Kotulla 1985"
320 PRINT:PRINT STRING$(40,210):PRINT
330 PEN 1:PRINT " Mit diesem Programm k"+CHR$(255)+"nnen Sie"
340 PRINT:PRINT " komplette Disketten auf C60-Cassetten"
350 PRINT:PRINT " abspeichern."
360 WHILE INKEY$<>"":WEND ' Tastaturpuffer loeschen
370 PEN 2:PRINT:PRINT:PRINT " Bitte legen Sie die Diskette ins"
380 PRINT:PRINT " Laufwerk und dr"+CHR$(254)+"cken Sie eine Taste!"
390 WHILE INKEY$="":WEND ' Auf Tastendruck warten
400 ' Interface zu den Maschinenspracheteilen *****
410 init=&A008 ' Initialisierungsroutine
420 readb=&A023 ' 45 Sektoren von der Diskette lesen
430 writeb=&A03F ' 45 Sektoren auf Diskette schreiben
440 ' Diskettentyp initialisieren *****
450 CALL init
460 :DISC
470 OPENOUT "DUMMY.$$$":CLOSEOUT ' FDC-Parameter aktualisieren
480 fdcblock=PEEK(&BE42)+PEEK(&BE43)*256 ' Zeiger auf FDC-Datenblock
490 format=PEEK(fdcblock+&F)-1 ' Formatierungstyp feststellen
500 POKE &A028,format+1:POKE &A044,format+1
510 POKE &A034,format+9:POKE &A050,format+9
520 POKE &A038,format+1:POKE &A054,format+1
530 ' Menueauswahl *****
540 MODE 1:PEN 3:PRINT:PRINT " Geben Sie bitte folgende Daten an:"
550 PEN 1:PRINT:PRINT STRING$(40,210):PRINT:PEN 3
560 PEN 2:PRINT " [1] ";:PEN 1:PRINT " Diskette abspeichern"
570 PRINT
580 PEN 2:PRINT " [2] ";:PEN 1:PRINT " Cassette einlesen"
590 PRINT
600 a$=INKEY$:IF a$="" THEN 600
610 IF a$="1" THEN 650
620 IF a$="2" THEN 940
630 PRINT CHR$(7);:GOTO 600
640 ' Diskette auf Cassette transferieren *****
650 PRINT " Cassettenmeldungen anzeigen? ";

```



```

660 a$=UPPER$(INKEY$):IF a$="" THEN 660
670 IF a$="J" OR a$="Y" THEN cas$="":GOTO 700
680 IF a$="N" THEN cas$="!":GOTO 700
690 PRINT CHR$(7);:GOTO 660
700 PRINT UPPER$(a$)
710 PEN 1:PRINT:PRINT STRING$(40,210):PRINT
720 PEN 2:PRINT " [0] ";:PEN 1:PRINT " SPEED WRITE 0"
730 PRINT
740 PEN 2:PRINT " [1] ";:PEN 1:PRINT " SPEED WRITE 1"
750 PRINT
760 a$=INKEY$:IF a$="" THEN 760
770 IF a$="0" THEN SPEED WRITE 0:GOTO 800
780 IF a$="1" THEN SPEED WRITE 1:GOTO 800
790 PRINT CHR$(7);:GOTO 760
800 PRINT STRING$(40,210)
810 PEN 3:WINDOW 2,39,22,24
820 track=0
830 FOR i=1 TO 8
840 POKE &A02A,track:POKE &A046,track:track=track+5
850 IF cas$="" THEN 870
860 PRINT USING " Reading .... Track ## - ##";track-5;track
870 CALL readb
880 FOR j=1 TO 5500:NEXT j ' Warten, bis Floppymotor aus ist!
890 IF cas$="!" THEN PRINT " Writing ...."
900 :TAPE:SAVE cas$+"CBACKUP DISC"+STR$(i),b,17000,23040
910 NEXT i
920 GOTO 1120 ' Programmende
930 ' Cassette auf Diskette transferieren *****
940 PRINT " Cassettenmeldungen anzeigen? ";
950 a$=UPPER$(INKEY$):IF a$="" THEN 950
960 IF a$="J" OR a$="Y" THEN cas$="":GOTO 990
970 IF a$="N" THEN cas$="!":GOTO 990
980 PRINT CHR$(7);:GOTO 950
990 PRINT UPPER$(a$):PRINT:PRINT STRING$(40,210)
1000 PEN 3:WINDOW 2,39,15,24
1010 track=0
1020 FOR i=1 TO 8
1030 POKE &A02A,track:POKE &A046,track:track=track+5
1040 IF cas$="!" THEN PRINT " Reading ...."
1050 :TAPE:LOAD cas$+"CBACKUP DISC"+STR$(i)
1060 IF cas$="" THEN 1080
1070 PRINT USING " Writing .... Track ## - ##";track-5;track
1080 CALL writeb
1090 FOR j=1 TO 5500:NEXT j ' Warten, bis der Floppymotor aus ist!
1100 NEXT i
1110 ' Programmende *****
1120 CLS:PRINT "Fertig - Die Daten sind "+CHR$(254)+"bertragen!"
1130 PRINT:PRINT "Noch eine Diskette laden/speichern?"
1140 a$=UPPER$(INKEY$):IF a$="" THEN 1140
1150 IF a$="N" THEN MODE 1:PEN 1:END
1160 IF a$="J" THEN CLEAR:MODE 1:GOTO 260
1170 PRINT CHR$(7);:GOTO 1140

```


„Memory Check“ — Prüfen Sie Ihren Computer!

Eigentlich sollte es ja nicht vorkommen, aber kein Computer ist vor Defekten sicher. Besonders ärgerlich ist das, wenn ein Speicherchip nicht mehr richtig funktioniert und aus scheinbar unerklärlichen Gründen Programme, die vorher noch einwandfrei liefen, dauernd „aussteigen“. Mit dem Maschinenprogramm „MEMORY CHECK“ können Sie sich Gewißheit über den Zustand Ihres Computers verschaffen.

Das Programm läuft auf allen drei Schneider-Computern.

MEMORY CHECK ist vollständig in Maschinensprache geschrieben. Das bedeutet, daß Sie entweder den Quellcode abtippen und assemblieren oder den Basic-Loader eingeben können. Entscheiden Sie sich für den Quellcode, dann müssen Sie zum Starten des Programms CALL &A2A1 eingeben; der Basic-Loader macht dies automatisch und sucht außerdem geeignete Bildschirmfarben heraus.

Das Programm meldet sich mit MEMORY CHECK V1.0 und zeigt in der Bildschirmmitte laufend die geprüften Speicheradressen in Hexadezimal-Schreibweise an. Um die Verarbeitungsgeschwindigkeit zu erhöhen, erfolgt die Anzeige nur an Page-Grenzen, also nach jeweils 256 Bytes. Ist Ihr Computer fehlerfrei, erscheint nach kurzer Zeit Speicher von &0040–&A000 Ok. In einem zweiten Durchgang wird dann der Bildschirm-RAM von &C000 bis &FFFF untersucht. Auch hier erfolgt bei einwandfreien Testergeb-

nissen eine entsprechende Bildschirmausgabe: Speicher von &C000–&FFFF Ok – Bitte druecken Sie eine Taste! Nach dem Tastendruck wird ein Reset ausgelöst, da eine Rückkehr nach Basic wegen der völlig veränderten Speicherinhalte kaum noch möglich ist.

Sollten Sie zu den Pechvögeln mit defektem Computer gehören, erscheint bei nicht funktionierenden Speicherstellen die Fehlermeldung Adreßfehler und die defekte Adresse wird ausgegeben. Nach einem Tastendruck wird auch hier ein Reset ausgelöst. In einem solchen Fall bleibt Ihnen nichts anderes übrig, als den Computer reparieren zu lassen.

Es ist aber dringend zu empfehlen, das Programm schon vor dem Auftreten solcher Fehler abzutippen und zu speichern – wenn die Fehler auftauchen, kann es zur Programmeingabe schon zu spät sein!

Für Interessierte hier noch einige Angaben zur Funktionsweise des Programms: MEMORY CHECK setzt in einer Programmschleife zuerst jede Speicherstelle auf Null und liest den Inhalt der Speicherstelle wieder. Erkennt das Programm einen anderen Wert als Null, so erfolgt eine Fehlermeldung. In einem zweiten Test werden daraufhin alle Bits der Speicherstelle auf Eins gesetzt, MEMORY CHECK lädt also den Wert 255 in die Speicherstelle. Wird beim erneuten Lesen ein anderer Wert festgestellt, so erfolgt die Fehlermeldung.

Ziemlich schwierig war die Wahl der Speicherbereiche: Der Bereich von &0000 bis &0040 ist für Verän-

derungen tabu, da dort die Restart-Vektoren stehen, die von allen ROM-Routinen verwendet werden. So würde z. B. die Bildschirmausgabe zum Systemabsturz führen. Ebenso verbietet sich die Veränderung des System-RAMs, da hier wichtige Daten für den Computer stehen, insbesondere die Sprungtabelle, ohne die keine ROM-Routinen mehr verwendet werden können. Auch befindet sich im Adreßbereich unter 49152 der Systemstack des Z80-Prozessors. Diesen z. B. in einem Unterprogramm zu verändern, führt beim RET-Rücksprung in den allermeisten Fällen zum System-Crash. Deshalb beschränkt sich MEMORY CHECK darauf, den Bereich &0040 bis &A200 zu untersuchen. Hier können alle Speicherinhalte verändert werden, wenn man mit dem Z80-Maschinenbefehl DI alle Interrupts des Computers verbietet und auf eine Rückkehr nach Basic verzichtet. In einer zweiten Schleife untersucht das Programm nach obig beschriebenem Verfahren den RAM-Speicher von &C000 bis &FFFF. Da dort aber normalerweise der Bildschirmspeicher liegt, wäre es nicht gerade ästhetisch, die Tests vor den Augen des Programmbenutzers durchzuführen. Schwerer wirkt aber das Problem, daß keine Bildschirmausgaben möglich wären. So kopiert MEMORY CHECK ganz einfach den gesamten Bildschirmspeicher mit LDIR in den Bereich von &4000 bis &7FFF und schaltet dann auf diesen Alternativbildschirm um. Der Benutzer merkt davon natürlich überhaupt nichts. Martin Kotulla/LM

Listing: MEMORY-CHECK

```

1000 ' ; *****
1010 ' ; *
1020 ' ; *      MEMORY CHECK - prueft den RAM-Speicher
1030 ' ; *
1040 ' ; *****
1050 '
1060 '      ORG  &A200      ; Startadresse des Programms
1070 '
1080 ' WAITKY EQU  &BB18      ; KM WAIT KEY wartet auf Tastendruck
1090 ' OUTPUT EQU  &BB5A      ; TXT OUTPUT druckt Zeichen auf dem Screen
1100 ' SETBAS EQU  &BC08      ; SCR SET BASE verschiebt den Bildschirm
1110 '
1120 ' TEXT1  DEFB  4,1,10,32,24,32,32,32,32,32,32,77,69
1130 '      DEFB  77,79,82,89,32,67,72,69,67,75,32
1140 '      DEFB  86,49,46,48,32,32,32,32,32,32,24
1150 '      DEFB  13,10,0
1160 '
1170 ' TEXT2  DEFB  31,2,20,83,112,101,105,99,104,101,114,32
1180 '      DEFB  118,111,110,32,38,48,48,52,48,45
1190 '      DEFB  38,65,48,48,48,32,79,107,10,13,0
1200 '
1210 ' TEXT3  DEFB  31,2,22,83,112,101,105,99,104,101,114,32
1220 '      DEFB  118,111,110,32,38,67,48,48,48,45
1230 '      DEFB  38,70,70,70,70,32,79,107,13,10,10
1240 '
1250 ' TEXT4  DEFB  32,66,105,116,116,101,32,101,105,110
1260 '      DEFB  101,32,84,97,115,116,101,32,100
1270 '      DEFB  114,117,101,99,107,101,110,33,0
1280 '
1290 ' ERRMSG  DEFB  10,13,10,32,65,100,114,101,115,115,102
1300 '      DEFB  101,104,108,101,114,32,105,110,32,0
1310 '
1320 ' CRLF    DEFB  13,10,10,0      ; Wagenruecklauf und zweimal Line-Feed
1330 '
1340 ' CURSET  DEFB  31,18,10,0      ; Steuerzeichen fuer LOCATE 18,10
1350 '
1360 ' START   LD    HL,TEXT1      ; Zeiger auf Text 1
1370 '      CALL PRINT      ; "MEMORY CHECK V1.0" ausgeben
1380 '      DI              ; Interrupts unterdruecken
1390 '
1400 ' CHECK1  LD    HL,&0040      ; Ende des RST-Bereichs
1410 '      LD    DE,TEXT1      ; Zeiger auf Anfang des Programms
1420 '      CALL CHKMEN      ; Speicher zwischen (HL) und (DE) pruefen
1430 '      LD    HL,TEXT2      ; Zeiger auf Text 2
1440 '      CALL PRINT      ; "Speicher von &0040-&A000 Ok" ausgeben
1450 '
1460 ' SCMOVE  LD    HL,&C000      ; Zeiger auf Original-Bildschirmfang
1470 '      LD    DE,&4000      ; Zeiger auf zweiten Bildschirmspeicher
1480 '      LD    BC,&4000      ; Der Bildschirm besteht aus 16KByte
1490 '      LDIR              ; Bildschirm umkopieren
1500 '      LD    A,&40          ; Highbyte der neuen Startadresse
1510 '      CALL SETBAS      ; Neue Bildschirmbasis setzen

```



```

1520 '
1530 ' CHECK2 LD HL,&C000 ; Zeiger auf Anfang des Pruefbereichs
1540 ' LD DE,&FFFA ; Zeiger auf Ende des Pruefbereichs
1550 ' CALL CHKMEM ; Speicher zwischen (HL) und (DE) pruefen
1560 ' LD HL,TEXT3 ; Wenn Ok, dann Zeiger auf Text 3
1570 ' CALL PRINT ; "Speicher von &C000-&FFFF Ok" ausgeben
1580 ' CALL WAITKY ; Auf Tastendruck warten
1590 ' RST &00 ; Reset ausfuehren
1600 '
1610 ' ; CHKMEM prueft den Speicher von (HL) bis (DE) *****
1620 '
1630 ' CHKMEM SUB A ; Akku loeschen
1640 ' LD (HL),A ; In (HL) alle Bits zuruecksetzen
1650 ' LD A,(HL) ; Adresse wieder in den Akku einlesen
1660 ' CP 0 ; Mit Null vergleichen
1670 ' JR NZ,ERROR ; Wenn nicht Null, dann Fehlermeldung
1680 '
1690 ' LD A,&FF ; Akku auf logisch Eins
1700 ' LD (HL),A ; In (HL) alle Bits setzen
1710 ' LD A,(HL) ; Adresse wieder in den Akku einlesen
1720 ' CP &FF ; Mit &FF vergleichen
1730 ' JR NZ,ERROR ; Wenn nicht &FF, dann Fehlermeldung
1740 '
1750 ' SHOW? LD A,L ; CP-Befehl vorbereiten
1760 ' CP 0 ; Ist eine Page-Grenze erreicht?
1770 ' JR NZ,NEXT ; Nein - weiterhin pruefen
1780 '
1790 ' HEXADR PUSH HL ; HL-Register sichern
1800 ' LD HL,CURSET ; Zeiger auf LOCATE-String
1810 ' CALL PRINT ; Cursor positionieren
1820 ' POP HL ; HL-Register wiederherstellen
1830 ' PUSH HL ; Und wiederum sichern
1840 ' CALL HEXOUT ; Adresse als Hexziffer ausgeben
1850 ' POP HL ; HL-Register wiederherstellen
1860 '
1870 ' NEXT INC HL ; Adresszeiger erhoehen
1880 ' CALL CPHLDE ; CP HL,DE: Speicherende erreicht?
1890 ' JR NZ,CHKMEM ; Nein - Speicher weiterhin pruefen
1900 ' RET ; Ruecksprung zum aufrufenden Programm
1910 '
1920 ' ERROR LD A,7 ; Fehler: BEEP-Code laden
1930 ' CALL OUTPUT ; Und ausgeben
1940 ' DI ; Interrupts wieder unterdruecken
1950 ' PUSH HL ; HL-Register sichern
1960 ' LD HL,ERRMSG ; Zeiger auf "Adressfehler in "
1970 ' CALL PRINT ; Diesen String ausgeben
1980 ' POP HL ; HL-Register wiederherstellen
1990 ' CALL HEXOUT ; Adresse als Hexziffern ausgeben
2000 ' LD HL,CRLF ; Zeiger auf CR/LF/LF-String
2010 ' CALL PRINT ; Ausgeben
2020 ' LD HL,TEXT4 ; Zeiger auf "Bitte Taste druecken!"
2030 ' CALL PRINT ; Ausgeben
2040 ' CALL WAITKY ; Auf Tastendruck warten
2050 ' RST &00 ; Reset ausfuehren
2060 ' POP HL ; HL-Register wiederherstellen

```



```

2070 '          RET          ; Ruecksprung
2080 '
2090 ' ; CPHLDE simuliert CP HL,DE *****
2100 '
2110 ' CPHLDE LD    A,H
2120 '          XOR    D
2130 '          LD     A,H
2140 '          JP     P,CPPLUS
2150 '          ADD    A,A
2160 '
2170 ' CPSUB  SBC    A,A
2180 '          RET    C
2190 '          INC    A
2200 '          RET
2210 '
2220 ' CPPLUS CP     D
2230 '          JR     NZ,CPSUB
2240 '          LD     A,L
2250 '          SUB    E
2260 '          JR     NZ,CPSUB
2270 '          RET
2280 '
2290 ' ; PRINT druckt von (HL) bis-(HL)=0 *****
2300 '
2310 ' PRINT  PUSH AF          ; Akku und Flags retten
2320 '
2330 ' PTLOOP LD    A,(HL)      ; Akku mit ASCII-Zeichen laden
2340 '          CP     0        ; Mit Null vergleichen
2350 '          JR     Z,PTRETN  ; Bei Null: Stringende
2360 '          CALL  OUTPUT     ; Sonst auf dem Bildschirm ausgeben
2370 '          DI          ; Interrupts wieder unterdruecken
2380 '          INC    HL        ; Adresszeiger erhoehen
2390 '          JR     PTLOOP    ; Und Ruecksprung in die Schleife
2400 '
2410 ' PTRETN POP    AF          ; Akku und Flags wiederherstellen
2420 '          RET          ; Ruecksprung zum aufrufenden Programm
2430 '
2440 ' HEXOUT LD    A,91        ; Ascii-Code fuer "["
2450 '          CALL  OUTPUT     ; Ausgeben
2460 '          DI          ; Interrupts verbieten
2470 '          LD     A,H        ; Akku mit dem H-Register laden
2480 '          CALL  SHIFT      ; Shiften und als Hexziffer ausgeben
2490 '          LD     A,H        ; Akku mit dem H-Register laden
2500 '          CALL  PRT         ; Als Hexziffer ausgeben
2510 '          LD     A,L        ; Akku mit dem L-Register laden
2520 '          CALL  SHIFT      ; Shiften und als Hexziffer ausgeben
2530 '          LD     A,L        ; Akku mit dem L-Register laden
2540 '          CALL  PRT         ; Als Hexziffer ausgeben
2550 '          LD     A,93        ; Ascii-Code fuer "]"
2560 '          CALL  OUTPUT     ; Ausgeben
2570 '          DI          ; Interrupts verbieten
2580 '          RET          ; Ruecksprung zum aufrufenden Programm
2590 '
2600 ' SHIFT  RRA          ; Viermal je ein Bit aus dem rechten
2610 '          RRA          ; Nibble des Akkus herausshiften, so

```



```

2620 '      RRA      ; dass nur noch das linke Nibble uebrig-
2630 '      RRA      ; bleibt
2640 '
2650 ' PRT      AND    &0F      ; Linkes Nibble des Akkus ausblenden
2660 '      ADD    A,48      ; ASCII-Offset addieren
2670 '      CP     58      ; Groesser als "9"?
2680 '      JR     C,SHOW      ; Wenn nicht, direkt ausgeben
2690 '      ADD    A,7      ; Sonst Differenz zwischen "9" und "A"
2700 '
2710 ' SHOW     CALL  OUTPUT      ; Ausgeben
2720 '      RET      ; Ruecksprung zum aufrufenden Programm
2730 '
2740 '      END      ; *****

```

Listing: MEMORY-CHECK

```

100 ' *****
110 ' * *
120 ' *      MEMORY CHECK *
130 ' * *
140 ' *****
150 '
160 MEMORY 41471
170 FOR i=41472 TO 41845
180 READ a:POKE i,a:NEXT i
190 INK 1,0:INK 0,13:BORDER 10
200 PEN 1:PAPER 0:SPEED INK 30,20
210 CALL &A2A1 ' MEMORY CHECK aufrufen
220 DATA &04,&01,&0A,&20,&18,&20,&20,&20,&20,&20,&4D,&45,&4D,&4F,&52
230 DATA &59,&20,&43,&48,&45,&43,&4B,&20,&56,&31,&2E,&30,&20,&20,&20,&20
240 DATA &20,&20,&18,&0D,&0A,&00,&1F,&02,&14,&53,&70,&65,&69,&63,&68,&65
250 DATA &72,&20,&76,&6F,&6E,&20,&26,&30,&30,&34,&30,&2D,&26,&41,&30,&30
260 DATA &30,&20,&4F,&6B,&0A,&0D,&00,&1F,&02,&16,&53,&70,&65,&69,&63,&68
270 DATA &65,&72,&20,&76,&6F,&6E,&20,&26,&43,&30,&30,&30,&2D,&26,&46,&46
280 DATA &46,&46,&20,&4F,&6B,&0D,&0A,&0A,&20,&42,&69,&74,&74,&65,&20,&65
290 DATA &69,&6E,&65,&20,&54,&61,&73,&74,&65,&20,&64,&72,&75,&65,&63,&6B
300 DATA &65,&6E,&21,&00,&0A,&0D,&0A,&20,&41,&64,&72,&65,&73,&73,&66,&65
310 DATA &68,&6C,&65,&72,&20,&69,&6E,&20,&00,&0D,&0A,&0A,&00,&1F,&12,&0A
320 DATA &00,&21,&00,&A2,&CD,&38,&A3,&F3,&21,&40,&00,&11,&00,&A2,&CD,&DA
330 DATA &A2,&21,&26,&A2,&CD,&38,&A3,&21,&00,&C0,&11,&00,&40,&01,&00,&40
340 DATA &ED,&B0,&3E,&40,&CD,&08,&BC,&21,&00,&C0,&11,&FA,&FF,&CD,&DA,&A2
350 DATA &21,&47,&A2,&CD,&38,&A3,&CD,&18,&BB,&C7,&97,&77,&7E,&FE,&00,&20
360 DATA &21,&3E,&FF,&77,&7E,&FE,&FF,&20,&19,&7D,&FE,&00,&20,&0D,&E5,&21
370 DATA &9D,&A2,&CD,&38,&A3,&E1,&E5,&CD,&47,&A3,&E1,&23,&CD,&25,&A3,&20
380 DATA &D9,&C9,&3E,&07,&CD,&5A,&BB,&F3,&E5,&21,&84,&A2,&CD,&38,&A3,&E1
390 DATA &CD,&47,&A3,&21,&99,&A2,&CD,&38,&A3,&21,&68,&A2,&CD,&38,&A3,&CD
400 DATA &18,&BB,&C7,&E1,&C9,&7C,&AA,&7C,&F2,&30,&A3,&87,&9F,&D8,&3C,&C9
410 DATA &BA,&20,&F9,&7D,&93,&20,&F5,&C9,&F5,&7E,&FE,&00,&28,&07,&CD,&5A
420 DATA &BB,&F3,&23,&18,&F4,&F1,&C9,&3E,&5B,&CD,&5A,&BB,&F3,&7C,&CD,&64
430 DATA &A3,&7C,&CD,&68,&A3,&7D,&CD,&64,&A3,&7D,&CD,&68,&A3,&3E,&5D,&CD
440 DATA &5A,&BB,&F3,&C9,&1F,&1F,&1F,&1F,&E6,&0F,&C6,&30,&FE,&3A,&38,&02
450 DATA &C6,&07,&CD,&5A,&BB,&C9
460 END ' *****

```


Anzeige der Funktionstasten

Auf den Schneider-Computern lassen sich mit dem KEY-Befehl die Funktionstasten sehr einfach belegen. Leider ist es dann nicht mehr möglich, die Belegung der Tasten anzeigen zu lassen. Das Programm „KEYLIST“ schafft da Abhilfe.

Das Programm initialisiert einen RSX-Befehl, mit dem der Sprachumfang des Basic-Interpreters erweitert wird:

|KEYLIST

Nach Eingabe dieses Befehls werden die ersten 25 Funktionstasten angezeigt, und zwar im gleichen Format wie bei der Eingabe des KEY-Befehls:

```
KEY &80,"0"
KEY &81,"1"
KEY &82,"Dies ist Funktions-
taste 3"
usw.
```

Dieses Format hat den Vorteil, daß Sie durch Benutzung der COPY-Taste alte Tastenbelegungen übernehmen und dabei modifizieren können.

Wenn der Bildschirm vollgeschrieben ist, hält das Programm an und wartet auf einen Tastendruck von Ihnen. Nach diesem werden die übrigen Tastenbelegungen angezeigt. Einige Worte zur Funktionsweise des Programms: KEYLIST liest beim CPC-464 aus der Speicherstelle &B4E1 (beim CPC-664 und CPC-6128 ist dies die Adresse &B62B) einen Zeiger auf den Beginn der Belegungstabelle der Funktionstasten. Beim CPC-464 ergibt dies normalerweise die Startadresse 46150. In der darauffolgenden Schleife werden für jede Taste zuerst der Text „KEX &“, die Tastennummer, ein Komma und ein Anführungszeichen ausgegeben. In einer weiteren Schleife liest KEYLIST dann die Stringketten aus dem Speicher und zeigt sie am Bildschirm an. Sollte die Taste nicht belegt sein und somit der Funktionsstring die Länge Null haben, gibt KEYLIST einen Leerstring aus. Sobald der ASCII-Code 159 erreicht wird – das ist die höchste belegbare Funktionstaste –, kehrt das Programm wieder nach Basic zurück.

Der Basic-Loader ist übrigens so programmiert, daß er mit einer kleinen Maschinenroutine automatisch

den verwendeten Computer erkennt. Dies geht ganz einfach:

```
CALL KL U ROM ENABLE
(Basic-ROM einschalten)
LD A,(&C002)
(Modifikationbyte im Basic-
ROM lesen)
LD (&016D),A
(Im PEEKbaren Speicher
unterbringen)
RET
(Rücksprung nach Basic)
```

Durch `Version=PEEK(&016D)` kann der Computertyp festgestellt werden:

```
Version=0: CPC-464
Version=1: CPC-664
Version=2: CPC-6128
Andere Werte: bisher nicht
verwendet
```

Bei `Version=1` und `Version=2` muß im Maschinenprogramm eine Adresse geändert werden, nämlich &B4E1 in &B62B. Im Bedarfsfall wird dies vom Programm automatisch erledigt. Erkennt das Programm einen Wert größer als 2, so gibt es eine entsprechende Fehlermeldung aus. Martin Kotulla/LM

Listing: KEYLIST

```
1000 ' ; *****
1010 ' ; *
1020 ' ; * KEYLIST - RSX-Erweiterung zur Anzeige der Funktionstasten-Belegung *
1030 ' ; *
1040 ' ; *****
1050 '
1060 '      ORG  &A000      ; Der Basic-Loader ist relocalisierbar!
1070 '
1080 ' BUFFST EQU  &B4E1      ; Start Expbuffer, CPC-664/6128: &B62B
```



```

1090 ' WAITKY EQU &BB18 ; KM WAIT KEY - wartet auf Tastendruck
1100 ' OUTPUT EQU &BB5A ; TXT OUTPUT - Zeichen auf Bildschirm ausgeben
1110 ' WRCHAR EQU &BB5D ; TXT WR CHAR - druckt auch Control-Codes
1120 ' LOGEXT EQU &BCD1 ; KL LOG EXTERNAL - RSX-Routinen einbinden
1130 '
1140 ' RSXTAB DEFW NAMTAB ; Zeiger auf Namenstabelle
1150 ' JP KYLIST ; Sprungvektor auf die Routine
1160 '
1170 ' NAMTAB DEFM "KEYLIS" ; RSX-Name KEYLIST
1180 ' DEFB &D4,&00 ; Ende des Namens & Ende der Tabelle
1190 '
1200 ' SPACE DEFS &04 ; Hilfsspeicher fuer Betriebssystem
1210 '
1220 ' INIT LD BC,RSXTAB ; Zeiger auf RSX-Tabelle
1230 ' LD HL,SPACE ; Zeiger auf Hilfsspeicher
1240 ' CALL LOGEXT ; RSX-Routine einbinden
1250 ' RET ; Ruecksprung nach Basic
1260 '
1270 ' TEXT DEFB 13
1280 ' DEFM "KEY"
1290 ' DEFB 32,38
1300 '
1310 ' KYLIST LD HL,(BUFFST) ; Zeiger auf Beginn der Funktionszeichen
1320 ' LD C,128 ; Code der ersten Funktionstaste
1330 ' LD A,12 ; Code fuer CLEAR SCREEN
1340 ' CALL OUTPUT ; Bildschirm loeschen
1350 '
1360 ' PRINT PUSH BC ; BC-Register sichern
1370 ' PUSH HL ; HL-Register sichern
1380 ' LD B,6 ; 6 Zeichen ausgeben
1390 ' LD HL,TEXT ; Zeiger auf String "KEY &"
1400 '
1410 ' PLOOP LD A,(HL) ; Ein Zeichen des Strings lesen
1420 ' CALL OUTPUT ; Auf dem Bildschirm ausgeben
1430 ' INC HL ; Stringzeiger erhoehen
1440 ' DJNZ PLOOP ; Fertig? - wenn nicht, dann weitermachen
1450 ' POP HL ; HL-Register wiederherstellen
1460 ' POP BC ; BC-Register wiederherstellen
1470 '
1480 ' SHOWHX CALL HEX ; ASCII-Code der Funktionstaste ausgeben
1490 ' LD A,44 ; ASCII-Code des Kommas
1500 ' CALL OUTPUT ; Ebenfalls ausgeben
1510 ' LD A,34 ; ASCII-Code des Anfuehrungszeichens
1520 ' CALL OUTPUT ; Wiederum ausgeben
1530 '
1540 ' LOOP1 LD A,(HL) ; Laengenbyte der Zeichenkette lesen
1550 ' LD B,A ; Als Schleifenzaehler ins B-Register
1560 ' INC HL ; Zeiger auf Zeichenadresse erhoehen
1570 ' CP 0 ; War die Funktionstaste ein Leerstring?
1580 ' JR Z,CLOSE ; Ja - Schlusszeichen und naechste Taste
1590 '
1600 ' LOOP2 LD A,(HL) ; Ein Zeichen des Funktionsstrings lesen
1610 ' PUSH AF ; AF-Register sichern
1620 ' PUSH BC ; BC-Register sichern
1630 ' PUSH DE ; DE-Register sichern

```



```

1640 '      PUSH HL          ; HL-Register sichern
1650 '      CALL WRCHAR      ; Zeichen ausgeben
1660 '      POP HL           ; HL-Register wiederherstellen
1670 '      POP DE           ; DE-Register wiederherstellen
1680 '      POP BC           ; BC-Register wiederherstellen
1690 '      POP AF           ; AF-Register wiederherstellen
1700 '      INC HL           ; Zeiger auf das naechste Zeichen setzen
1710 '      DJNZ LOOP2       ; Solange weiter, bis der String gelesen ist
1720 '
1730 ' CLOSE LD A,34         ; Anfuehrungszeichen als Schlusszeichen laden
1740 '      CALL OUTPUT      ; Und ausgeben
1750 '      LD A,10          ; Line-Feed laden
1760 '      CALL OUTPUT      ; Und ausgeben
1770 '
1780 ' NEXT INC C            ; Zaehler fuer die Funktionstasten erhoehen
1790 '      LD A,C           ; Vorbereitung des CP-Befehls
1800 '      CP 153           ; Nach 25 Tasten ist der Bildschirm voll
1810 '      CALL Z,WAITKY    ; Wenn ja, dann auf Tastendruck warten
1820 '      CP 160           ; Hoechster Funktionstastencode ist 159
1830 '      JR NZ,PRINT      ; Solange nicht erreicht, weitermachen
1840 '      RET              ; Sonst Ruecksprung nach Basic
1850 '
1860 ' HEX LD A,C            ; Hexziffer in den Akku laden
1870 '      CALL SHIFT       ; Shiften und linkes Nibble ausgeben
1880 '      LD A,C           ; Akku wiederherstellen
1890 '      CALL HXPRNT      ; Linkes Nibble ausblenden, rechtes ausgeben
1900 '      RET              ; Ruecksprung zum aufrufenden Programm
1910 '
1920 ' SHIFT RRA             ; Mit diesen vier RRA-Befehlen wird
1930 '      RRA              ; das rechte Nibble aus dem Akku
1940 '      RRA              ; herausgeschifftet und das linke Nibble
1950 '      RRA              ; zum rechten Nibble gemacht
1960 '
1970 ' HXPRNT AND 15         ; Mit 15 UNDieren loescht linkes Nibble
1980 '      ADD A,48         ; Aus Binaer wird ASCII
1990 '      CP 58            ; Groesser als 9?
2000 '      JR C,HEXOUT      ; Nein - Zeichen ausdrucken
2010 '      ADD A,7          ; Ja - ASCII-Zahlen-Buchstaben-Offset!
2020 '
2030 ' HEXOUT CALL OUTPUT    ; Hexziffer ausgeben
2040 '      RET              ; Und Ruecksprung zum aufrufenden Programm
2050 '
2060 '      END

```

Listing: KEYLIST

```

100 ' *****
110 ' *
120 ' *      KEYLIST
130 ' *
140 ' *****

```



```

150 '
160 DEF FNmsb(a)=255 AND INT(a/256)
170 DEF FNlsb(a)=255 AND UNT(a)
180 MODE 1:INK 0,13:INK 1,0:INK 2,0,3:INK 3,1
190 PAPER 0:PEN 3:BORDER 10:SPEED INK 30,20
200 LOCATE 1,1:PRINT STRING$(40,210);
210 LOCATE 1,2:PRINT CHR$(24)+SPACE$(16)+"KEYLIST"+SPACE$(17)+CHR$(24)
220 LOCATE 10,5:PRINT CHR$(164);" Martin Kotulla 1985"
230 PRINT:PRINT STRING$(40,210):PRINT
240 PEN 1:PRINT " Mit diesem Programm wird der Basic-"
250 PRINT:PRINT " Interpreter Ihres Computers um einen"
260 PRINT:PRINT " RSX-Befehl erweitert, mit dem die Be-"
270 PRINT:PRINT " legung der Funktionstasten angezeigt"
280 PRINT:PRINT " werden kann: ";
290 PEN 2:PRINT ":KEYLIST"
300 PEN 3:PRINT:PRINT STRING$(40,210);
310 PEN 2:PRINT:PRINT " Bitte druecken Sie eine Taste!":CALL &BB18:PEN 1
320 MEMORY HIMEM-&90:start=HIMEM+1
330 FOR i=start TO start+&8E:READ a:POKE i,a:NEXT i
340 RESTORE 630:FOR i=1 TO 8:READ a:a=a+start
350   value=PEEK(a)+PEEK(a+1)*256-40960+start
360   POKE a,FNlsb(value):POKE a+1,FNmsb(value)
370 NEXT i
380 CALL start+17 ' RSX-Routine initialisieren
390 ' Maschinenroutine zum Erkennen des Computers *****
400 POKE &160,&CD:POKE &161,&0:POKE &162,&B9:POKE &163,&3A
410 POKE &164,&2:POKE &165,&C0:POKE &166,&32:POKE &167,&6D
420 POKE &168,&1:POKE &169,&C9:CALL &160
430 version=PEEK(&16D)
440 IF version=0 THEN :KEYLIST:END
450 IF version<>1 AND version<>2 THEN 480
460 POKE start+34,&2B:POKE start+35,&B6
470 :KEYLIST:END
480 CLS:PEN 2
490 PRINT:PRINT " Ich kann mit dieser Version des"
500 PRINT:PRINT " Computers nichts anfangen!":PRINT:PEN 1:END
510 ' Daten des Maschinencodes *****
520 DATA &05,&A0,&C3,&21,&A0,&4B,&45,&59,&4C,&49,&53,&D4,&00,&00,&00,&00
530 DATA &00,&01,&00,&A0,&21,&0D,&A0,&CD,&D1,&BC,&C9,&0D,&4B,&45,&59,&20
540 DATA &26,&2A,&E1,&B4,&0E,&80,&3E,&0C,&CD,&5A,&BB,&C5,&E5,&06,&06,&21
550 DATA &1B,&A0,&7E,&CD,&5A,&BB,&23,&10,&F9,&E1,&C1,&CD,&74,&A0,&3E,&2C
560 DATA &CD,&5A,&BB,&3E,&22,&CD,&5A,&BB,&7E,&47,&23,&FE,&00,&28,&0F,&7E
570 DATA &F5,&C5,&D5,&E5,&CD,&5D,&BB,&E1,&D1,&C1,&F1,&23,&10,&F1,&3E,&22
580 DATA &CD,&5A,&BB,&3E,&0A,&CD,&5A,&BB,&0C,&79,&FE,&99,&CC,&18,&BB,&FE
590 DATA &A0,&20,&B8,&C9,&79,&CD,&7D,&A0,&79,&CD,&81,&A0,&C9,&1F,&1F,&1F
600 DATA &1F,&E6,&0F,&C6,&30,&FE,&3A,&38,&02,&C6,&07,&CD,&5A,&BB,&C9
610 DATA aa
620 ' Daten zum Relokalisieren des Maschinencodes *****
630 DATA 0000,0003,0018,0021,0048,0060,0118,0122
640 END ' -----

```


Der Kompaktor

Mit diesem Programm erfüllen wir zahlreiche Leserwünsche. Immer wieder wurde die Redaktion gebeten, zu allen Maschinenprogrammen auch die entsprechenden Assemblerlistings abzudrucken und zu erläutern. Wir können dies leider nicht bei allen Programmen tun. Dies würde den Rahmen eines Heftes total sprengen und wir müssten komplette Bücher für das eine oder andere größere Programm schreiben. Den Kompaktor, der auch im Textverarbeitungsprogramm unseres ersten Schneider-Kompaktwissen-Heftes enthalten war, wollen wir aber etwas näher erläutern, da er als Utilitie (Hilfsprogramm) auch bei Ihren eigenen Programmen Einsatz finden kann.

Eigentlich sagt der Name schon, welchen Sinn das Programm hat. Irgendetwas wird kompakter gemacht also „verdichtet“. Im vorliegenden Falle geht es um im Speicher stehende Daten.

Häufig befinden sich in einem Speicher jede Menge nicht brauchbare, also redundante Bytes. Bei einer Datei oder einem Text eines Textverarbeitungsprogrammes sind dies zum Beispiel Leerstellen (Spaces). Bei einem Bildspeicher können es aber auch „Nullen“ sein. Sollen diese Daten nun auf einem externen Speichermedium (Kassette, Diskette, Microdrive usw.) „verewigt“ werden, so braucht man für diesen Vorgang mehr Platz und mehr Zeit wenn diese „Redundanzbytes“ mit abgespeichert werden. Genau hier aber wird der Sinn und Zweck des Kompaktoreinsatzes deutlich. Der Kompaktor soll Abhilfe schaffen.

Der Kompaktor besteht aus drei Teilen. (Die Benennungen haben wir willkürlich gewählt.)

Der Teil A (clear) dient dazu, den Speicherbereich zu löschen.

Teil B (kompr) ist der „Packer“ der den Speicherbedarf komprimiert, also der eigentliche Kompaktor.

Teil C (expan) ist der „Entpacker“

der die Daten wieder auf die ursprüngliche Speicherplatzausdehnung expandiert.

Teil A ist also eigentlich kein Bestandteil dieser Utilitie, ist aber deshalb integriert, damit man zu Beginn einen „sauberen“ Speicher besitzt. Durch diesen Teil wird (im vorliegenden Programm) der Speicher mit Leerstellen aufgefüllt.

Teil B sollte vor jedem Save-Vorgang aufgerufen werden. Von Basic aus erhält das Programm die Adressen für Speicheranfang und -ende. Das Speicherende ist im vorliegenden Falle mit „&F9“ gekennzeichnet. (Betrachten Sie die Arbeitsweise vor allem auch in Verbindung mit dem im ersten Heft abgedruckten Textverarbeitungsprogramm!)

Nach dem Programmaufruf beginnt der Teil B mit der Suche nach Leerstellen (chr\$(32)). Sobald dieser Code gefunden ist, schaut das Programm nach, ob mindestens noch weitere drei Leerstellen folgen. Ist dies nicht der Fall, dann geht die Suche weiter. Sind aber mehr als drei Speicherstellen hintereinander mit diesem Code belegt, dann wird komprimiert. Dazu ist als erstes die Anzahl der Leerstellen von Nöten. Diese wird zusammen mit einem „Kompressionszeichen“ (&F8) in die ersten drei Leerstellen geladen. Anschließend wird der Speicher so verschoben, daß alle Leerstellen in diesem Teil verschwinden. Sobald dies erledigt ist, springt das Programm wieder an den Anfang, und die Suche geht von neuem los. Irgendwann ist nichts mehr zum Komprimieren da, der Programmteil hat seine Schuldigkeit getan und übergibt in die Speicherstellen &160/&161 die Länge des komprimierten Speichers.

Nach einem „SAVE“ sollte sich nun unmittelbar der Lauf des dritten Programmteiles anschließen, damit der Datenspeicher wieder in den „Originalzustand“ zurückgesetzt wird. Denn sonst könnte es vorkommen, daß durch ein Print-Kommando etwas aus dem komprimierten

Speicher ausgegeben wird, was zu größter Verwirrung führt. Zum Beispiel Print chr\$(4).

Um dem also vorzubeugen, regeneriert der Expansionsteil wieder den Speicher.

Von Basic aus kommen die originalen Adressen für Speicheranfang und -ende. Gesucht wird nun nach &F8. Wird das Programm fündig, so kann der Speicher wieder um die Anzahl der Leerstellen verschoben werden. Die Leerstellen werden eingefügt und &F8 verschwindet. Nachdem dieser Programmteil abgeschlossen ist, befindet sich der Speicher wieder im „Originalzustand“. Wer bisher alles mitverfolgt hat, weiß es bereits: Auch direkt nach einem „LOAD“ sollte der Aufruf zur Expansion erfolgen. Andererseits aber macht es dem Expansionsteil nichts, eine nichtkomprimierte Datei vorgesetzt zu bekommen.

Der Aufruf der Routinen ist eine Schneider-Besonderheit, die nicht mit allen Z-80-Rechnern möglich ist. Mit CALL adresse,a,b können einem Maschinenprogramm auch die Werte zweier Variablen bekannt gemacht werden. Das Register „DE“ enthält bereits die letzte Variable, in diesem Falle also den Wert von „b“. Die zweite Variable erhält man über den Zeiger „IX“, der bei „IX+2“ auf das Lowbyte von „a“ zeigt. Natürlich können auch mehr als zwei Variable an das Programm übergeben werden. Das „A“-Register enthält die Anzahl der Variablen nach dem CALL-Befehl.

Generelles zum Programm

Die Adresse &9D00 ist als Startadresse nur ein Beispiel. Der gesamte Code ist an jeder Stelle des Speichers „abrufbar“. Deshalb wurden als „Merkadressen“ auch die Speicherstellen ab &160 benutzt. Soll der Kompaktor zu Komprimierung von anderen Codefolgen (also nicht für Leerstellen) benutzt werden, so

sind die entsprechenden Codes im Programm zu ändern. Hierfür sollte dann das Assemblerlisting zu Rate gezogen werden. Das Programm komprimiert in der vorliegenden Version, wenn mindestens vier Leerstellen vorhanden sind. Das kann teilweise einige Zeit in Anspruch nehmen. Wird dieser Wert z. B. auf zehn erhöht, dann kann evtl. eine wesentlich schnellere Abarbeitung erfolgen. Dies hängt aber auch vom Speicherinhalt ab. Auf

welchen Wert man die Grenze setzt, muß der Anwendungsfall entscheiden.

Damit Sie den Kompaktor auch direkt in Aktion sehen können, haben wir ein kleines Demonstrationsprogramm mit abgedruckt. Dabei wird dann der Speicher vor der Kompression, nach Kompression und dann wieder nach der Expansion dargestellt. Damit das ganze nicht zu langweilig ist, wird der Abstand zwischen den Worten des „geistrei-

chen“ Textes mit einer Anzahl zufälliger Leerstellen gefüllt. Dadurch wird auch sichtbar, daß der Kompaktor erst bei der vorgegebenen Anzahl der Leerstellen aktiv wird. Ganz zum Schluß soll noch angemerkt werden, daß dieses Programm bei vielen Anwendungen eingesetzt werden kann. Als Beispiel sei hier nur auf den Einsatz bei einer Dateiverwaltung hingewiesen.

B. Fernhomberg/LM

ARNOR Z80 ASSEMBLER version 1.10

Page 001

```

00002      ;***** (c) 10/85 FB
00003      ;*** CPC KOMPAKTOR ***
00004      ;*****
00005
00006      ;Alle Routinen werden von BASIC mit
00007      ; CALL x,Speicherende,Speicheranfang
00008      ;aufgerufen.
00009
00010      ;*** VARIABLENLISTE ***
00011
00012      9F00 (0160)      SPHL EQU &160      ;Laenge kompr. Speicher
00013      9F00 (0162)      SPBC EQU &162
00014      9F00 (0164)      SPDE EQU &164
00015      9F00 (0166)      ZWSP EQU &166
00016
00017      9D00 (9F00)      ORG &9D00,$
00018
00019      ;*** KOMPRESSION ***
00020
00021      9D00 ED 53 62 01      KOMPR LD (SPBC),DE      ;Speicheranfang und
00022      9D04 DD 6E 02          LD L,(IX+2)      ; Speicherende kommen
00023      9D07 DD 66 03          LD H,(IX+3)      ; aus BASIC.
00024      9D0A 36 F9          LD (HL),&F9      ;Zeichen Speicherende
00025      9D0C 23          INC HL
00026      9D0D 22 60 01      LD (SPHL),HL
00027      9D10 2A 62 01      LD HL,(SPBC)      ;Speicheranfang
00028
00029      9D13 EB          BEGINN EX DE,HL      ;DE ist akt. Sp.adr.
00030      9D14 2A 60 01      LD HL,(SPHL)      ;akt. Speicherende
00031      9D17 A7          AND A      ;Errechnung Zaehler fuer
00032      9D18 ED 52          SBC HL,DE      ; Space Suche.
00033      9D1A E5          PUSH HL
00034      9D1B C1          POP BC      ;BC ist Zaehler
00035      9D1C EB          EX DE,HL
00036
00037      9D1D 3E 20          LD A,&20      ;Suche nach Space
00038      9D1F ED B1          CPIR
00039      9D21 28 0E          JR Z,KOMP      ;Ist BC=0, dann Ende
00040

```



```

00041 9D23 ED 4B 62 01 ENDE LD BC,(SPBC) ;akt. Speicherende minus
00042 9D27 2A 60 01 LD HL,(SPHL) ; Speicheranfang =
00043 9D2A A7 AND A ; Gesamtlänge des komp.
00044 9D2B ED 42 SBC HL,BC ; Speichers.
00045 9D2D 22 60 01 LD (SPHL),HL ;Länge ist in SPHL
00046 9D30 C9 RET ; gespeichert.
00047
00048 9D31 2B KOMP1 DEC HL ;1. Adr. von Space
00049 9D32 E5 PUSH HL ; wird gerettet.
00050 9D33 01 01 00 LD BC,1
00051
00052 9D36 03 KOMP1 INC BC ;Anzahl der Spaces
00053 9D37 23 INC HL ; wird gesucht.
00054 9D38 BE CP (HL)
00055 9D39 28 FB JR Z,KOMP1
00056
00057 9D3B D1 POP DE
00058 9D3C 97 SUB A ;Ist BC<4, dann wird
00059 9D3D B8 CP B ; das nächste Space
00060 9D3E 20 05 JR NZ,KOMP2 ; gesucht.
00061 9D40 3E 03 LD A,3
00062 9D42 B9 CP C
00063 9D43 30 CE JR NC,BEGINN
00064
00065 9D45 EB KOMP2 EX DE,HL
00066 9D46 36 F8 LD (HL),&F8 ;Kompressionszeichen
00067 9D48 23 INC HL ;Die Anzahl der Spaces
00068 9D49 71 LD (HL),C ; wird in den Speicher
00069 9D4A 23 INC HL ; geladen.
00070 9D4B 70 LD (HL),B
00071 9D4C 23 INC HL
00072 9D4D EB EX DE,HL ;DE VerschiebungsZIEL
00073
00074 9D4E D5 PUSH DE ;Adr. wird gerettet
00075
00076 9D4F E5 PUSH HL ;HL ist QUELLE
00077 9D50 E5 PUSH HL
00078 9D51 C1 POP BC
00079
00080 9D52 2A 60 01 LD HL,(SPHL) ;akt. Sp.ende minus
00081 9D55 A7 AND A ; Quelle ergibt
00082 9D56 ED 42 SBC HL,BC ; Zaehler fuer Sp.-
00083 9D58 E5 PUSH HL ; verschiebung.
00084 9D59 C1 POP BC ;BC ZAEHLER
00085 9D5A E1 POP HL
00086
00087 9D5B ED B0 LDIR ;Speicherverschiebung
00088
00089 9D5D E1 POP HL ;Adr. fuer Fortsetzung
00090 9D5E ED 53 60 01 LD (SPHL),DE ;aktuelles Speicherende
00091 9D62 18 AF JR BEGINN
00092
00093 ;*** EXPANSION ***
00094
00095 9D64 DD 6E 02 EXPAN LD L,(IX+2) ;Speicheranfang und

```


00096	9D67	DD 66 03		LD	H, (IX+3)	; Speicherende kommen
00097	9D6A	ED 53 62 01		LD	(SPBC), DE	; aus BASIC.
00098	9D6E	A7		AND	A	
00099	9D6F	ED 52		SBC	HL, DE	
00100	9D71	22 64 01		LD	(SPDE), HL	; Speicherlaenge
00101						
00102	9D74	2A 62 01		LD	HL, (SPBC)	; Suche nach akt.
00103	9D77	3E F9		LD	A, &F9	; Speicherende.
00104	9D79	ED 4B 64 01		LD	BC, (SPDE)	; Gesamtspeicherlaenge
00105	9D7D	ED B1		CP1R		
00106	9D7F	C0		RET	NZ	
00107	9D80	2B		DEC	HL	
00108						
00109	9D81	22 60 01		LD	(SPHL), HL	; akt. Speicherende
00110	9D84	2A 62 01	BEGEX	LD	HL, (SPBC)	
00111	9D87	3E F8		LD	A, &F8	; Suche nach
00112	9D89	ED 4B 64 01		LD	BC, (SPDE)	; Kompressions-
00113	9D8D	ED B1		CP1R		; zeichen.
00114	9D8F	C0		RET	NZ	; Ist BC=0, dann ENDE.
00115						
00116	9D90	2B		DEC	HL	
00117	9D91	E5		PUSH	HL	; 1.Spaceadresse
00118	9D92	23		INC	HL	
00119	9D93	4E		LD	C, (HL)	; BC Anzahl der einzu-
00120	9D94	23		INC	HL	; fuegenden Spaces
00121	9D95	46		LD	B, (HL)	; minus 4.
00122	9D96	0B		DEC	BC	
00123	9D97	0B		DEC	BC	
00124	9D98	0B		DEC	BC	
00125	9D99	0B		DEC	BC	
00126	9D9A	ED 43 66 01		LD	(ZWSP), BC	; Anzahl wird gerettet.
00127						
00128	9D9E	2A 60 01		LD	HL, (SPHL)	; das alte Sp.ende ist
00129	9DA1	E5		PUSH	HL	; die QUELLE.
00130	9DA2	09		ADD	HL, BC	
00131	9DA3	22 60 01		LD	(SPHL), HL	; neues akt. Ende
00132						
00133	9DA6	EB		EX	DE, HL	; DE ZIEL
00134	9DA7	E1		POP	HL	
00135	9DA8	C1		POP	BC	
00136	9DA9	E5		PUSH	HL	; 1.Spaceadr. minus
00137	9DAA	A7		AND	A	; Quelle ergibt den
00138	9DAB	ED 42		SBC	HL, BC	; ZAEHLER.
00139	9DAD	E5		PUSH	HL	
00140	9DAE	C1		POP	BC	
00141	9DAF	E1		POP	HL	; HL QUELLE
00142						
00143	9DB0	ED B8		LDDR		; Speicherverschiebung
00144						
00145	9DB2	ED 4B 66 01		LD	BC, (ZWSP)	; Einfuegen der Spaces
00146	9DB6	03		INC	BC	
00147	9DB7	03		INC	BC	
00148	9DB8	36 20		LD	(HL), &20	; HL QUELLE
00149	9DBA	E5		PUSH	HL	
00150	9DBB	D1		POP	DE	


```

00151 9DBC 13          INC  DE          ;DE ZIEL
00152 9DBD ED B0      LDIR
00153
00154 9DBF 18 C3      JR    BEGEX
00155 9DC1 (9FC1)     END

```

Errors: 00000 Warnings: 00000

SYMBOL TABLE:

9D13 BEGINN	9D84 BEGEX	9D23 ENDE	9D64 EXPAN
9D00 KOMPR	9D31 KOMP	9D36 KOMP1	9D45 KOMP2
0160 SPHL	0162 SPBC	0164 SPDE	0166 ZWSP

Listing: KOMPDEMO (Kompaktor-Demonstration)

```

100 '***** c 10.85/FB
110 '*** KOMPAKTOR DEMO ***
120 '*****
130 MODE 1:MEMORY 19999
140 LOCATE 9,2:PRINT"K O M P A K T O R   Demo"
150 LOCATE 8,3:PRINT STRING$(19,"=")
160 LOCATE 1,9:PRINT"ACHTUNG! Dieses Programm arbeitet nur,"
170 LOCATE 10,11:PRINT"wenn der KOMPAKTOR geladen ist."
180 LOCATE 9,18:PRINT"Bitte legen Sie die Cassette mit"
190 LOCATE 9,20:PRINT"dem KOMPAKTOR M-Code ein."
200 LOCATE 9,22:PRINT"Starten Sie dann den Recorder."
210 LOAD"!KOMPCODE.BIN",&9D00
220 '*****
230 '*** DEMO ***
240 '*****
250 MODE 2
260 LOCATE 2,2:PRINT"Speicheranfang: 20000",:spa=20000
270 PRINT CHR$(&F8);": Zeichen fuer Kompression"
280 LOCATE 2,4:PRINT"Speicherende   : 20080",:spe=20080
290 PRINT CHR$(&F9);": Zeichen fuer akt. Speicherende"
300 '
310 LOCATE 2,8:PRINT "Dieses steht als Test im Speicher:"
320 a$=" DER"+STRING$(RND*10," ")+ "KOMP."+STRING$(RND*10," ")+ "HAT"
330 a$=a$+STRING$(RND*10," ")+ "GROSSE"+STRING$(RND*10," ")+ "VORTEILE."
340 a$=a$+STRING$(80-LEN(a$),CHR$(32))
350 FOR n=1 TO LEN(a$):POKE 19999+n,ASC(MID$(a$,n,1)):NEXT
360 LOCATE 1,10:GOSUB 520
370 '
380 CALL &9D00,spe,spa:'KOMPRESSION
390 LOCATE 2,14:PRINT"Nach KOMPRESSION (Laenge:";
400 PRINT PEEK(&160)+256*PEEK(&161);"): "
410 LOCATE 1,16:GOSUB 520
420 '
430 CALL &9D64,spe,spa:'EXPANSION
440 LOCATE 2,20:PRINT"und dann nach EXPANSION:"
450 LOCATE 1,22:GOSUB 520
460 '

```



```

470 LOCATE 2,25:PRINT"Bitte Taste druecken":CALL &BB06
480 GOTO 250
490 '*****
500 '*** SUBPROGRAMM ***
510 '*****
520 PRINT CHR$(&18);:FOR n=20000 TO 20080:c=PEEK(n)
530 IF c<32 THEN PRINT CHR$(c+48);:ELSE PRINT CHR$(c);
540 NEXT:PRINT CHR$(&18);:RETURN

```

Listing: KOMPAKTOR

```

100 '***** c 10.85/FB
110 '*** HEX-Lader fuer KOMPAKTOR ***
120 '*****
130 '
140 DIM a$(13):SPEED WRITE 1
150 A$(1)= "ED 53 62 01 DD 6E 02 DD 66 03 36 F9 23 22 60 01"
160 A$(2)= "2A 62 01 EB 2A 60 01 A7 ED 52 E5 C1 EB 3E 20 ED"
170 A$(3)= "B1 28 0E ED 4B 62 01 2A 60 01 A7 ED 42 22 60 01"
180 A$(4)= "C9 2B E5 01 01 00 03 23 BE 28 FB D1 97 B8 20 05"
190 A$(5)= "3E 03 B9 30 CE EB 36 F8 23 71 23 70 23 EB D5 E5"
200 A$(6)= "E5 C1 2A 60 01 A7 ED 42 E5 C1 E1 ED B0 E1 ED 53"
210 A$(7)= "60 01 18 AF*DD 6E 02 DD 66 03 ED 53 62 01 A7 ED"
220 A$(8)= "52 22 64 01 2A 62 01 3E F9 ED 4B 64 01 ED B1 C0"
230 A$(9)= "2B 22 60 01 2A 62 01 3E F8 ED 4B 64 01 ED B1 C0"
240 A$(10)= "2B E5 23 4E 23 46 0B 0B 0B 0B ED 43 66 01 2A 60"
250 A$(11)= "01 E5 09 22 60 01 EB E1 C1 E5 A7 ED 42 E5 C1 E1"
260 A$(12)= "ED B8 ED 4B 66 01 03 03 36 20 E5 D1 13 ED B0 18"
270 A$(13)= "C3 00 00 00 00 00 00 00 00 00 00 00 00 00"
280 '
290 'Pruefcode
300 '
310 DATA 1547,1989,1382,1575,2048,2636,1778
320 DATA 1688,1644,1079,2369,1822,195
330 '
340 '
350 MODE 2:RESTORE:INPUT"Startadresse fuer KOMPAKTOR: ",adr:sadr=adr
360 IF adr>43710 THEN 350 ELSE MEMORY (adr-1)
370 FOR m=1 TO 13:sum=0:a$(m)="00"+a$(m)
380 FOR n=1 TO 16:a=VAL("&"+MID$(a$(m),n*3))
390 POKE adr,a:sum=sum+a:adr=adr+1
400 NEXT:READ a
410 IF a<>sum THEN LOCATE 20,12:PRINT"FEHLER IN A$(";m;")":END
420 NEXT m
430 '
440 LOCATE 20,9:PRINT"Allen in Ordnung. Hex-Code ist geladen."
450 LOCATE 20,14:PRINT"Adr. Kompression: &";HEX$(sadr)
460 LOCATE 20,16:PRINT"Adr. Expansion : &";HEX$(sadr+100)
470 LOCATE 1,20
480 PRINT"Aufruf der Routinen: CALL (Adr.),Speicherende,Speicheranfang"
490 LOCATE 1,22:PRINT"SAVE KOMPAKTOR CODE:"
500 SAVE"Kompcode.bin",B,sadr,193
510 END

```


DIN-Tastatur für Schneider CPC 464/664/ 6128 unter CP/M 2.2

Leider stehen für den CPC unter CP/M keine deutschen Zeichen zur Verfügung. Mit diesem Programm haben Sie aber auch unter CP/M 2.2 eine DIN-Tastatur.

Zur DIN-Tastatur gehören die Umlaute und die entsprechende Tastaturbelegung wie sie auf einer Schreibmaschine vorkommt.

Beim Schneider-Computer kann man bekanntlich den Zeichensatz und die Tastaturbelegung ändern. Unter Basic gibt es entsprechende Befehle, die dazu benutzt werden können. Das Betriebssystem stellt für die Maschinensprache Sprungadressen zur Verfügung. Diese können auch unter CP/M 2.2 benutzt werden. Mit diesen Sprungadressen werden die Wiederholungen und die Änderung der Tastaturbelegung vorgenommen. So wird zum Beispiel auch die TAB-Taste und die ENTER-Taste in die Wiederholung mit einbezogen, da sie auch auf der

Schreibmaschine so vorhanden ist. Das Ändern des Zeichensatzes stellt dabei eine Besonderheit dar. Die Umlaute werden nicht in die Matrix eingebaut, sondern werden immer dann angerufen, wenn ein Zeichen zur Ausgabe ansteht. Dazu wird der Vector TXT WRITE CHAR der Betriebssystemroutine verbogen und auf eine Routine im Speicher gesetzt. Dort wird abgefragt ob ein entsprechendes ASCII-Zeichen (z. B. 7B) ansteht. Ist dies der Fall, wird der Zeiger auf die Deutschen Zeichen in einem Zwischenspeicher gestellt, und dem Betriebssystem mitgeteilt. Erst dann wird das Zeichen ausgegeben.

Wie erstelle ich mir nun ein Command-File auf Diskette?

Sie können den ED.COM unter CP/M dazu benutzen. Dieser Editor ist allerdings schwierig zu handhaben. Gehen Sie in CP/M Betriebssystem

und tippen Sie ED DIN.ASM ein. Der Editor meldet sich mit dem Bereitschaftszeichen :*.

Geben Sie nun I für Einfügen ein und tippen das Listing ein. Wenn Sie damit fertig sind, geben sie CONTROL-Z ein und dann E um das Programm auf Diskette zu sichern.

Falls Ihnen das zu Umständlich ist, können Sie auch jeden anderen Editor benutzen (z. B. TASWORD, TEXPACK) um das Listing einzugeben.

Ist das ASCII-File DIN.ASM auf Diskette, geben Sie unter CP/M den Befehl ASM DIN ein. Dieses generiert ein File DIN.HEX. Um das endgültige Programm zu erstellen, müssen Sie jetzt noch den Befehl LOAD DIN eingeben. Dann steht Ihnen das Programm DIN.COM zur Verfügung. Durch eingeben von DIN wird das Programm gestartet und Sie haben eine DIN-Tastatur.

Wolfgang Gentzsch/LM

```

;*****
;*      DIN-Tastatur unter CP/M 2.2      *
;*                                          *
;* Deutsche Umlaute und Tastenbelegung  *
;* unter CP/M 2.2 auf dem CPC 464/664    *
;*                                          *
;* Copyright (C) Wolfgang Gentzsch 1985  *
;*****
;
;Betriebssystem Call's
;
BB39 =      repeat      equ      0bb39h      ;KM SET REPEAT
BB27 =      trans       equ      0bb27h      ;KM SET TRANSLATE
BB2D =      shift       equ      0bb2dh      ;KM SET SHIFT

```



```

BB33 =      control      equ      0bb33h      ;KM SET CONTROL
BDD3 =      vector      equ      0bdd3h      ;TXT WRITE CHAR
;
;System Adressen
;
B700 =      tpa equ      0b700h      ;Treiber dort ablegen
134E =      matrix      equ      0134eh      ;TXT GET MATRIX
;
;
0100      org      0100h      ;TPA Start
;
;Tastaturwiederholungen einstellen
;FF = erlaubt  00 = nicht erlaubt
;AKKU = Tastennummer

0100 06FF      mvi      b,0ffh
0102 3E44      mvi      a,44h      ;Taste 'TAB'
0104 CD39BB      call      repeat
0107 06FF      mvi      b,0ffh
0109 3E12      mvi      a,12h      ;Taste 'ENTER'
010B CD39BB      call      repeat

;Tasten neu belegen
;a = Tastennummer b = neue Uebersetzung

010E 21FC01      lxi      h,tasten
0111 0610      mvi      b,10h      ;16 Tasten neu belegen
0113 C5      loop      push      b      ;Anzahl auf Stack
0114 7E      mov      a,m      ;Akku mit Tastennummer laden
0115 23      inc      h
0116 46      mov      b,m      ;belegung ohne Shift
0117 F5      push      psw      ;Tastennummer auf Stack
0118 E5      push      h      ;Adresse der Tabelle retten
0119 CD27BB      call      trans      ;KM SET TRANSLATE
011C E1      pop      h      ;Tabellen-Adresse zurueck
011D F1      pop      psw      ;Tastennummer zurueck
011E 23      inc      h
011F 46      mov      b,m      ;mit Shift
0120 F5      push      psw
0121 E5      push      h
0122 CD2DBB      call      shift      ;KM SET SHIFT
0125 E1      pop      h
0126 F1      pop      psw
0127 23      inc      h
0128 46      mov      b,m      ;mit Control
0129 E5      push      h
012A CD33BB      call      control      ;KM SET CONTROL
012D E1      pop      h
012E 23      inc      h
012F C1      pop      b      ;Anzahl der Tasten
0130 05      dcr      b      ;eine weniger
0131 78      mov      a,b
0132 FE00      cpi      00h      ;fertig?
0134 C21301      jnz      loop      ;wenn nicht, weiter
;
;
;

```



```

;Zeichentreiber im Speicher ablegen
;Der Bereich B700H - B800H wird vom CP/M nicht benutzt!
;
0137 215201      lxi      h,treiber ;Tabelle des Neuen Treibers
013A 1100B7      lxi      d,tpa      ;dort wird er abgelegt
013D D5          push     d          ;wird dem Betriebssystem
                                ;uebergeben
013E 06AA        mvi      b,tend-treiber ;laenge des Treibers

0140 7E          ldir      mov      a,m      ;Byte aus Treiber lade
n
0141 12          stax     d          ;im TPA neu ablegen
0142 23          inx      h
0143 13          inx      d
0144 05          dcr      b
0145 78          mov      a,b
0146 FE00        cpi      00h        ;alles verschoben ?
0148 C24001      jnz      ldir      ;sonst wiederholen
014B E1          pop      h          ;HL wird DE
014C 22D4BD      shld     vector+1 ;TXT WRITE CHAR patch
014F C30000      jmp      0000h      ;Warmstart CCP

;
0152 E5          treiber      push     h          ;wird TXT GET MATRIX u
eobergeben
0153 2162B7      lxi      h,tpa+para-treiber
0156 FE40        cpi      40h        ;$?
0158 37          stc          ;Carry-Flag an
0159 CA4E13      jz      matrix      ;ins Betriebssystem
015C 216AB7      lxi      h,tpa+sz-treiber
015F FE7E        cpi      7eh
0161 37          stc
0162 CA4E13      jz      matrix
0165 2172B7      lxi      h,tpa+dach-treiber
0168 FE5E        cpi      5eh
016A 37          stc
016B CA4E13      jz      matrix
016E 217AB7      lxi      h,tpa+kae-treiber
0171 FE7B        cpi      7bh
0173 37          stc
0174 CA4E13      jz      matrix
0177 2182B7      lxi      h,tpa+gae-treiber
017A FE5B        cpi      5bh
017C 37          stc
017D CA4E13      jz      matrix
0180 218AB7      lxi      h,tpa+koe-treiber
0183 FE7C        cpi      7ch
0185 37          stc
0186 CA4E13      jz      matrix
0189 2192B7      lxi      h,tpa+goe-treiber
018C FE5C        cpi      5ch
018E 37          stc
018F CA4E13      jz      matrix
0192 219AB7      lxi      h,tpa+kue-treiber
0195 FE7D        cpi      7dh
0197 37          stc
0198 CA4E13      jz      matrix
019B 21A2B7      lxi      h,tpa+gue-treiber

```



```

019E FE5D      cpi      5dh
01A0 37        stc
01A1 CA4E13    jz       matrix
;
;Berechnen der Zeichenmatrix im ROM
;Reg. HL zeigt auf entsprechende Matrix im ROM
;
01A4 D5        push     d                ;wird benoetigt
01A5 110038    lxi      d,3800h          ;Tabelle im ROM
01A8 6F        mov     l,a              ;Zeichen in Akku
01A9 2600      mvi     h,00h
01AB 29        dad     h
01AC 29        dad     h
01AD 29        dad     h
01AE 19        dad     d
01AF D1        pop     d
01B0 B7        ora     a
01B1 C34E13    jmp     matrix
;
;Tabelle der neuen Zeichen
;
01B4 3E60386C38para db      3eh,60h,38h,6ch,38h,0ch,0f8h,00h
01BC 386C6C7C66sz db      38h,6ch,6ch,7ch,66h,66h,6ch,60h
01C4 386CC60000dach db      38h,6ch,0c6h,00h,00h,00h,00h,00h
01CC 0066003E66kae db      00h,66h,00h,3eh,66h,66h,3bh,00h
01D4 663C66667Egae db      66h,3ch,66h,66h,7eh,66h,66h,00h
01DC 0066003C66koe db      00h,66h,00h,3ch,66h,66h,3ch,00h
01E4 C67CC6C6C6goe db      0c6h,7ch,0c6h,0c6h,0c6h,0c6h,7ch,00h
01EC 0066006666kue db      00h,66h,00h,66h,66h,66h,3ch,00h
01F4 6600666666gue db      66h,00h,66h,66h,66h,66h,3ch,00h
;
01FC =         tend     equ      $
;
;Hier beginnt die Tabelle der neuen Tastenbelegung
;
;1.Byte = Tastennummer
;2.Byte = Zeichen ohne Shift
;3.Byte = Zeichen mit Shift
;4.Byte = Zeichen mit Control
;
01FC 1C7B5B1B tasten db      1ch,7bh,5bh,1bh
0200 1A7D5D1D db      1ah,7dh,5dh,1dh
0204 1D7C5C1C db      1dh,7ch,5ch,1ch
0208 197E3F00 db      19h,7eh,3fh,00h
020C 2B7A5A1A db      2bh,7ah,5ah,1ah
0210 47795919 db      47h,79h,59h,19h
0214 39334000 db      39h,33h,40h,00h
0218 185E6000 db      18h,5eh,60h,00h
021C 112B2A00 db      11h,2bh,2ah,00h
0220 1323271E db      13h,23h,27h,1eh
0224 163C3E00 db      16h,3ch,3eh,00h
0228 1E2D5F1F db      1eh,2dh,5fh,1fh
022C 20303D00 db      20h,30h,3dh,00h
0230 272C3B00 db      27h,2ch,3bh,00h
0234 1F2E3A00 db      1fh,2eh,3ah,00h
0238 29372F00 db      29h,37h,2fh,00h
;
023C          end

```


Die zweite Möglichkeit für DIN.COM

Wolfgang Gentzsch, der Programmator des vorhergehenden Artikels hat mit seinem Programm bestimmt vielen Lesern einen Gefallen getan. Schwierigkeiten der Erzeugung von DIN.COM können aber bei CPC-Besitzern auftreten, die bisher noch wenig mit CP/M gearbeitet haben aber trotzdem dieses File abspeichern wollen. Deshalb hat ComputerSchau mal wieder in die Trickkiste gegriffen. Erzeugen wir dieses File doch einfach von Basic aus.

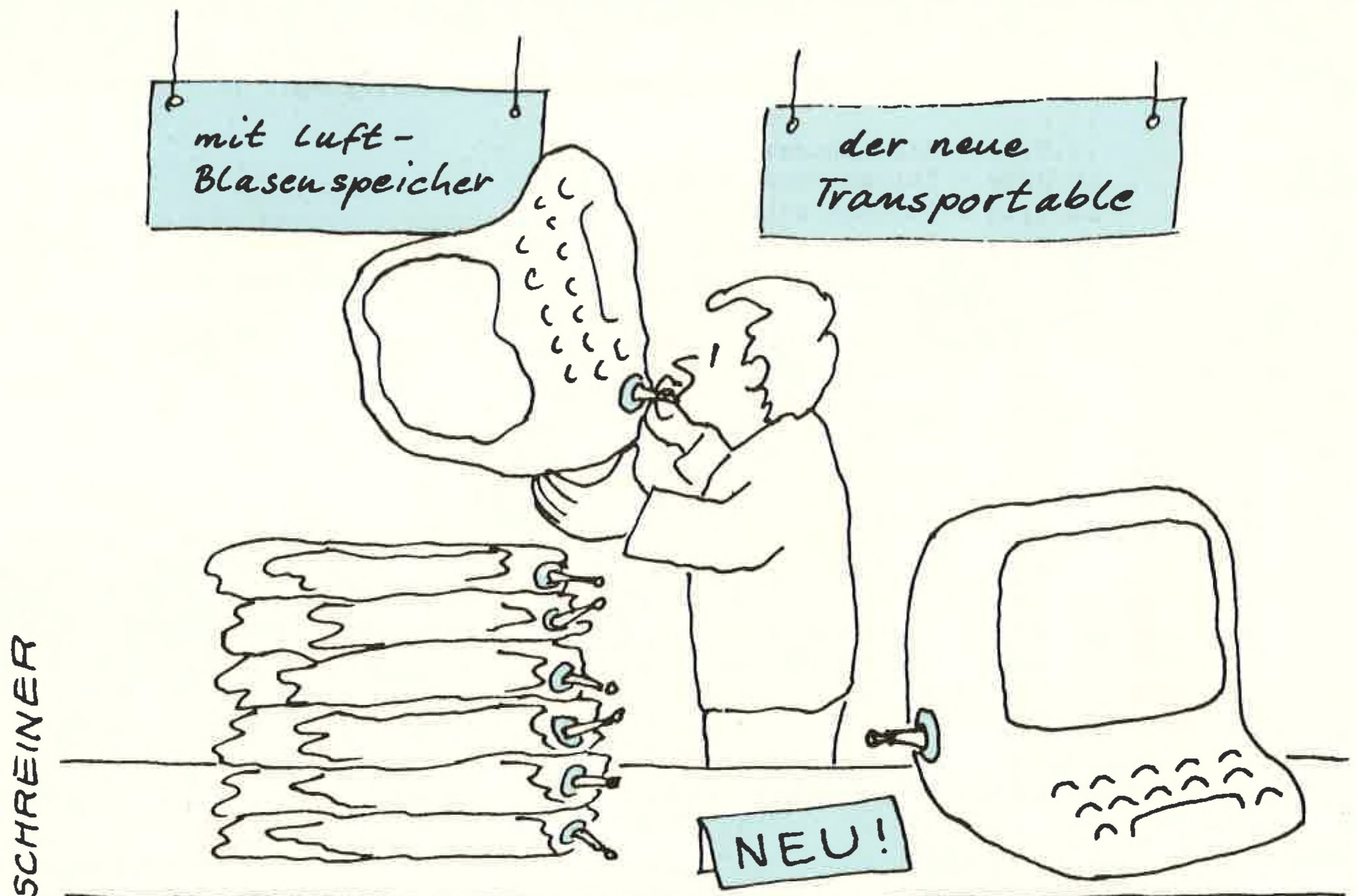
Etwas „untertrieben“ habe ich in obigen Vorspann. Denn mit Basic alleine schafft man es auch noch nicht so ohne weiteres. Aber mit dem, von den Grundlagenartikeln

schon bekannten Maschinenprogramm SECTOR klappt es, wenn man etwas trickst, ganz hervorragend. Dazu braucht man nun kein Textverarbeitungsprogramm und auch nicht den unkonfortablen CP/M-Editor sondern das erzeugende Programm ist ein normales Basiclisting. Aber mit ein paar kleinen Tricks, die es in sich haben.

Wie Ihnen sicherlich bekannt ist, kann man Maschinenprogramme als sogenannte Datastatements in Basicprogrammen ablegen und durch den Programmlauf in die entsprechenden Speicherstellen poken. Ist dieser Vorgang beendet, dann können die Maschinenprogramme auch als Binärfile abgespeichert und von den Basicprogrammen eingelesen werden.

Der Header ist das Problem

Mit einem Programm vom Typ .COM ist dies aber nicht so ohne weiteres möglich. Versuchen Sie ruhig einmal ein COM-File als Binärfile zu laden, es geht nicht. Erinnern Sie sich noch an das Problem des Lesers, der Eproms brennen wollte? Der Grund dieser Tatsache ist, daß bei Binärfiles nicht nur der reine Maschinencode auf der Disk abgelegt wird, sondern diesem ein Header, also ein Vorspann (mit Informationen für den CPC selbst) voransteht. COM-Files dagegen sind „headerlos“. Es ist nun aber nicht möglich, mit den normalen Befehlen für den Abspeichervorgang headerlose Files mit dem Extend COM zu erzeugen. Deshalb müssen wir



einen direkten Diskettenzugriff durchführen. Dieser Zugriff ist wie Ihnen schon bekannt mit SECTOR möglich. Was ist also alles zu tun, um mittels dieser Maschinenroutine das gewünschte File zu erzeugen? Nun als erstes brauchen wir zunächst einmal die Daten von DIN.COM. Dies habe ich für Sie schon erledigt, indem ich mittels einer modifizierten Version des Programmes READER, die Daten in den Bereich ab &9000 eingelesen hatte. Die Daten von DIN.COM standen nun im Speicher.

Vorher hatte ich mir aber den Fileeintrag dieses Programmes näher angesehen und festgestellt, daß drei Records für ihn benötigt werden. Dies bedeutete, daß ein Sector für das Programm ausreicht. Nachdem ich die Daten im Speicher hatte, nahm ich meinen DATA-Generator und ließ mir ein Datalisting zeigen. Eine Generierung von DIN.COM ist damit aber noch nicht möglich, denn beim Abspeichern als Binärfile hat man wieder den Header, an dem sich CP/M „verschluckt“. Also muß ein anderer Weg gegangen werden.

Bitte Listing ansehen

Um dies nun detailliert aufzuzeigen, sollten Sie sich beim Lesen des

nun folgenden auch immer wieder das Listing COMDIN betrachten. In den Zeilen bis 150 steht nichts außergewöhnliches. Diese Zeilen dürfen für Sie „Klartext“ sein.

Die Zeilen 160 und 170 dienen dazu, Sie darauf aufmerksam zu machen, daß Sie eine Diskette im Systemformat einlegen sollen, denn unter CP/M können Sie mit Data-only-Disketten wenig anfangen.

Und nun kommen bereits die ersten Tricks in den Zeilen 180 bis 210. Durch OPENOUT„din.com“ und den nachfolgenden Printbefehlen wird dafür gesorgt, daß ein Eintrag dieses Namens ins Inhaltsverzeichnis der Diskette erfolgt und auch irgendwelche Zeichen in das eigentliche File geschrieben werden. Der Schreibvorgang ist erforderlich, damit im Fileeintrag auch die gewünschte Recordlänge steht. Ist dieser nämlich niedriger, dann wird später zu wenig gelesen und der CPC stürzt ab. Eine direkte Manipulation im Inhaltsverzeichnis wäre umständlicher gewesen, deshalb dieser kleine „Dreh“. Anschließend wird das File korrekt geschlossen. Nun wird schon wieder getrickst. Um nämlich nachher zu wissen, auf welchen Sektor und auf welche Spur wir die Daten direkt schreiben müssen, ist es nötig diese Informationen zu kennen. Durch „öffnen“ des Files aber erfahren wir dies. Ab

der Speicherstelle &A719 und den folgenden stehen nämlich die Blocknummern des eröffneten Files. Da wir nur einen Sektor haben, können wir nun die Blocknummer aus dieser Speicherstelle abholen. Durch Zeile 210 wird diese Blocknummer in Track und Sector umgerechnet. In der Formel zur Umrechnung weist die Zahl 18 daraufhin, daß bei CP/M zwei Spuren à 9 Sektoren reserviert sind. Der Befehl MOD wird vielen 464-Besitzern unbekannt sein. Er ist in deren Handbuch nicht erwähnt, aber trotzdem vorhanden. Er bewirkt, daß ein gerundeter Rest (Modulo) bei einer Teilung von Dividend durch Divisor gebildet wird. (Werfen Sie doch mal einen Blick in ein 664- oder 6128-Handbuch!) Damit wissen wir nun, wohin wir mit den Daten für unsere deutsche Tastatur unter CP/M müssen. Nun werden die Daten aus den Datazeilen gelesen und in den ab &9000 reservierten Buffer abgelegt. Der Rest wird dann durch die bekannte Maschinenroutine erledigt, nachdem vorher die entsprechenden Werte für den Befehl usw. übergeben wurden. Das heißt diesen Teil kennen Sie ja schon. Am Ende des Programmlaufes steht das fertige Programm auf der Diskette. Wie Sie sehen, es ist vieles machbar, wenn man den CPC etwas genauer kennt.

Lothar Miedel

Listing: COMDIN - erzeugt DIN-Tastatur-Programm fuer CP/M

```
100 'Comdin
110 'Programm um von Basic aus das File DIN.COM zu erzeugen
120 '
130 MEMORY &8FFF:MODE 2:PRINT"Bitte die Diskette mit SECTOR.BIN einlegen"
140 PRINT"und dann eine Taste druecken ! ":CALL &BB06
150 LOAD"sector.bin",&A600
160 PRINT:PRINT"Nun bitte Ihre CP/M Arbeitsdiskette einlegen"
170 PRINT"und eine Taste druecken ! ":CALL &BB06
180 OPENOUT"din.com"
190 PRINT#9,STRING$(128,"*"):PRINT#9,STRING$(128,"*")
200 CLOSEOUT:OPENIN"din.com":bn=PEEK(&A719)
210 CLOSEIN:bn =bn*2+18:track=INT(bn/9):sector=bn MOD 9+1
```



```

220 '
230 PRINT:PRINT"Nun lese ich die Daten in den Buffer
240 a=&9000:e=&913F:zb=1000:e=e+1
250 FOR i =a TO e:READ d$:IF LEFT$(d$,1)="/" THEN flag =1
260 IF (flag AND ps<>VAL(d$)) THEN PRINT"Fehler in Zeile "zb+1:END
270 IF (flag AND i=e) THEN GOTO 330
280 IF flag THEN i=i-1:zb=zb+1:ps=0:d$="":flag = 0:GOTO 300
290 d$="/" +d$:POKE i,VAL(d$):ps=ps+VAL(d$):
300 IF i < e THEN NEXT i
310 '
320 'Adressen fuer Maschinenprogramm
330 aktdrivemc=&A702:befehlmc=&A622:buffermc=&A629:drivenmc=&A626
340 formsemc=&A628:trackmc=&A627:mcstart=&A600
350 '
360 buffer$="9000":'          Beginn des Buffers aus dem geschrieben wird
370 form =&40:'              Da DIN.COM fuer CP/M: nur Systemformat erlaubt !!
380 befehl =&85:POKE befehlmc,befehl
390 drive=PEEK(aktdrivemc)
400 '
410 'Werte fuer Bufferadresse berechnen
420 bufflow=VAL("&" +RIGHT$(buffer$,2)):buffhigh=VAL("&" +LEFT$(buffer$,2))
430 '
440 PRINT"Nun wird DIN.COM geschrieben
450 '*****
460 '*exakt einen Sektor schreiben *
470 '*****
480 secform =form+sector:POKE (buffermc),bufflow:POKE (buffermc+1),buffhigh
490 POKE drivenmc,drive:POKE trackmc,track:POKE formsemc,secform:CALL mcstart
500 PRINT"fertig !
510 '
1001 DATA 06,FF,3E,44,CD,39,BB,06,FF,3E,12,CD,39,BB,21,FC,&077B
1002 DATA 01,06,10,C5,7E,23,46,F5,E5,CD,27,BB,E1,F1,23,46,&0787
1003 DATA F5,E5,CD,2D,BB,E1,F1,23,46,E5,CD,33,BB,E1,23,C1,&0A2F
1004 DATA 05,78,FE,00,C2,13,01,21,52,01,11,00,B7,D5,06,AA,&0512
1005 DATA 7E,12,23,13,05,78,FE,00,C2,40,01,E1,22,D4,BD,C3,&069B
1006 DATA 00,00,E5,21,62,B7,FE,40,37,CA,4E,13,21,6A,B7,FE,&06FF
1007 DATA 7E,37,CA,4E,13,21,72,B7,FE,5E,37,CA,4E,13,21,7A,&0683
1008 DATA B7,FE,7B,37,CA,4E,13,21,82,B7,FE,5B,37,CA,4E,13,&07A7
1009 DATA 21,8A,B7,FE,7C,37,CA,4E,13,21,92,B7,FE,5C,37,CA,&0803
1010 DATA 4E,13,21,9A,B7,FE,7D,37,CA,4E,13,21,A2,B7,FE,5D,&0785
1011 DATA 37,CA,4E,13,D5,11,00,38,6F,26,00,29,29,29,19,D1,&047A
1012 DATA B7,C3,4E,13,3E,60,38,6C,38,0C,F8,00,38,6C,6C,7C,&05E5
1013 DATA 66,66,6C,60,38,6C,C6,00,00,00,00,00,00,66,00,3E,&03A6
1014 DATA 66,66,3B,00,66,3C,66,66,7E,66,66,00,00,66,00,3C,&0461
1015 DATA 66,66,3C,00,C6,7C,C6,C6,C6,C6,7C,00,00,66,00,66,&06AA
1016 DATA 66,66,3C,00,66,00,66,66,66,66,3C,00,1C,7B,5B,1B,&044F
1017 DATA 1A,7D,5D,1D,1D,7C,5C,1C,19,7E,3F,00,2B,7A,5A,1A,&0411
1018 DATA 47,79,59,19,39,33,40,00,18,5E,60,00,11,2B,2A,00,&031A
1019 DATA 13,23,27,1E,16,3C,3E,00,1E,2D,5F,1F,20,30,3D,00,&0261
1020 DATA 27,2C,3B,00,1F,2E,3A,00,29,37,2F,00,00,00,00,00,&01A4

```


MID\$, die zweite Möglichkeit mit CPC 464

Auch bei diesem Befehl schweigt sich das Handbuch des CPC 464 teilweise aus. Zwar wird die normale Anwendung des Befehles erläutert, aber dieser Befehl ist noch viel stärker und mächtiger als dort beschrieben. Es können nämlich mittels dieses Befehles Teile von bereits existierenden Strings ausgetauscht und ersetzt werden. Ein entsprechendes Beispielprogramm zeigt allen CPC-Besitzern auf, wie dieser Befehl eingesetzt werden kann und auch was er bewirkt. Da das Programm aber einige identische und auch einige fast übereinstimmende Basiczeilen beinhaltet, wird auch gleich die überaus komfortable Kopiertechnik mittels des COPY-Cursors beschrieben.

Diese nun aufgezeigte „Betriebs-technik“ des Befehles MID\$ hat gegenüber anderen Methoden den Vorteil, daß keine „Müll“-Strings anfallen und dadurch die von vielen so gefürchteten Wartezeiten der sogenannten „Garbage collection“, also der „Müllstringbeseitigung“ während des Programmlaufes reduziert werden können. Da das Programm mittels der Windowtechnik arbeitet, können auch hier viele Leser nachsehen, wie diese programmiert und gehandhabt werden. Aber auch noch weitere kleine Kniffe sind in diesem Programm enthalten, die für andere Programme sinnvoll sein können.

Die Programmbeschreibung

In Zeile 120 wird der Mode 2, also auf 80-Zeichendarstellung geschaltet. Außerdem werden gleich Variablen definiert, die im Programmlauf des öfteren auftreten. Wie zu sehen

ist, sind zwei dieser Variablen Adressen von Firmware-Routinen. Diese Definition hat den Vorteil, daß man nicht bei jedem Aufruf einer derartigen Routine im Firmware-Handbuch nachschlagen muß, welche Adresse nun zu programmieren ist. Außerdem sind während der Programmerstellung die Variablen invers oder warte leichter zu merken als vierstellige hexadezimale Zahlen.

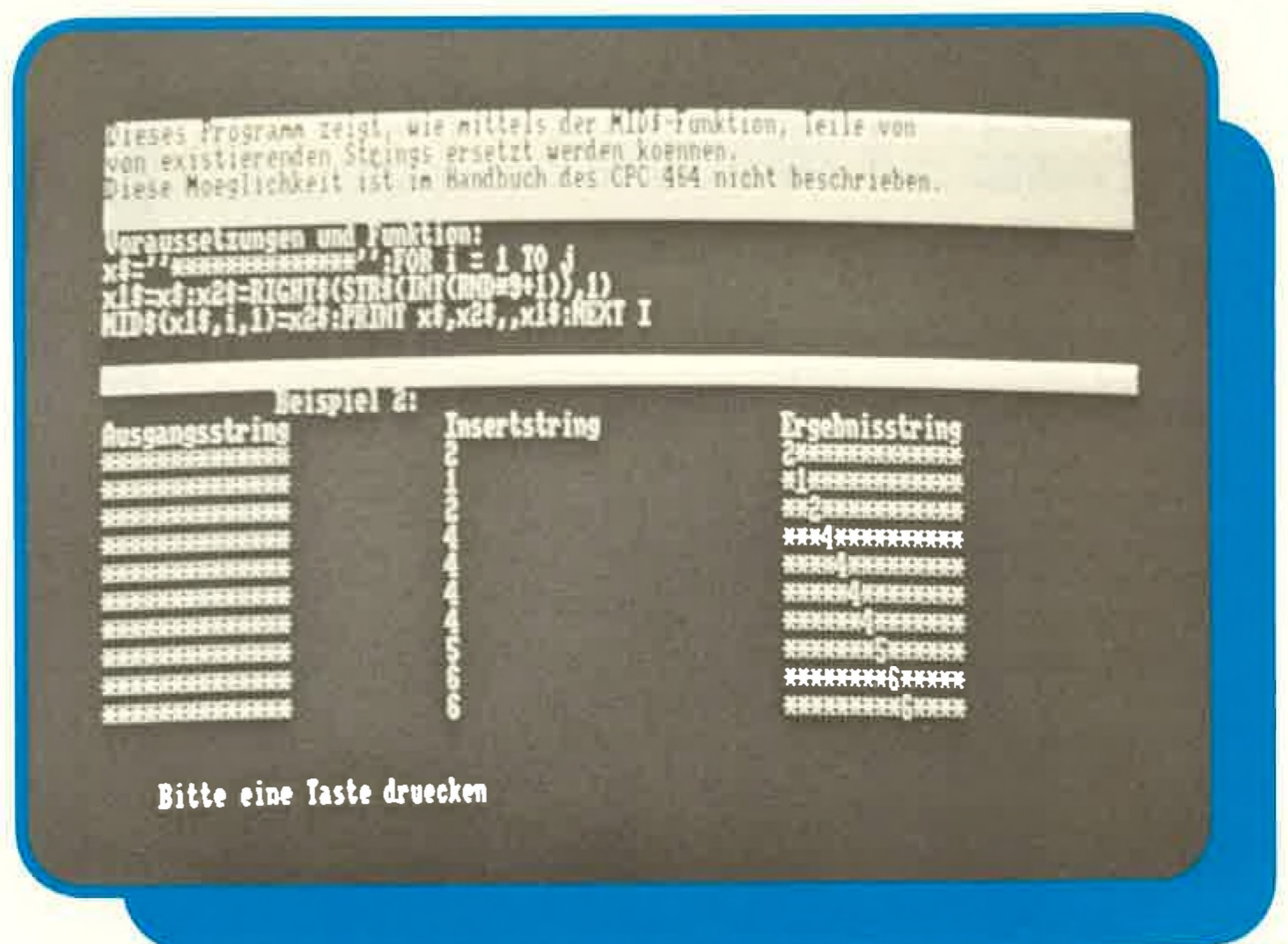
Einmal nachgesehen im Firmwarehandbuch, den Routinen leicht merk- und identifizierbare Namen zugewiesen, die ersten Basiczeilen damit versehen und einiges geht leichter. Auch das Lesen des Programmes ist dadurch sehr viel einfacher. Ich habe mir für meine Programmierarbeiten ein Grundprogramm erstellt, welches einige dieser Routinen enthält. Dieses lade ich mir bei Programmierbeginn ein. Diese Methode hat für mich den

Vorteil, daß die gebräuchlichen Namen der Routinen immer gleich sind und sich mir außerdem im Laufe der Zeit einprägen und ich wesentlich schneller programmieren kann. Ist das Programm fertiggestellt, wird der überflüssige Teil einfach wieder entfernt.

Die Zeilen 130 und 140 legen die Fenster für die Ausgaben fest. Die erforderlichen Erklärungen können Sie Ihrem Handbuch entnehmen. Beachten Sie aber, daß in diesem Programm kein Window mit der Nummer Null definiert wurde.

In Zeile 160 wird nun bereits der gesamte Bildschirm invertiert und auch gelöscht, denn erst dann ist wirklich die Invertierung (Austausch der Farben für PEN und PAPER) durchgeführt.

Die Zeilen 170 bis 190 bewirken eine Textausgabe in Window Null. Da bei diesem Programm eine Wiederholung gleicher bzw. stark ähnli-



cher Abläufe vorkommt, war es am einfachsten, diese Teile mittels der „Kopiertechnik“ zu vervielfältigen. Dies betrifft beispielsweise die Zeilen 200, 300, 460 oder auch 220, 340, 480. Da ich bei vielen Gesprächen mit Lesern immer wieder feststellen konnte, daß viele CPC-Besitzer diese Kopiertechnik nicht in vollem Umfange beherrschen, will ich nun, da das Programm sonst keine großen Besonderheiten mehr beinhaltet, näher auf diese Technik eingehen.

Die Kopiertechnik

Bei normalen Arbeiten mit den CPC's sieht man immer nur einen Cursor, der an der Stelle zu sehen ist, an welcher die nächsten Eingaben erfolgen. Dieser Cursor kann normalerweise mittels der Cursorsteuertasten auf jede (mögliche) beliebige Bildschirmposition gebracht werden. Wird dann die COPY-Taste betätigt, dann werden die entsprechend unter dem Cursor stehenden Zeichen übernommen. Ändert man nun beispielsweise eine Nummer einer auf dem Bildschirm stehenden Basiczeile um sie zu duplizieren, dann wird bei Kopierversuchen akustisch mitgeteilt, daß dies vom CPC nicht akzeptiert wird. Nun besteht die Möglichkeit, da sich der

Cursor ja nicht mehr nach rechts weiterbewegen läßt, durch Betätigung der RETURN- oder ENTER-Taste diese Situation zu meistern.

Der CPC hat aber diese Zeile nicht angenommen und teilt dies durch „Line does not exist“ mit. Dem CPC wurde nämlich eine Zeilennummer mitgeteilt, die in diesem Falle ja noch gar nicht existiert. Führt man nun mit dem Cursor wieder auf den Zeilenanfang dieser Zeile und betätigt dann (bis zum Ende des zu kopierenden Teiles) die COPY-Taste, dann kann diese Zeile übernommen werden. Aber diese eben beschriebene Methode ist nicht die wirkliche Kopiertechnik. Viel besser und auch wesentlich komfortabler ist die nachfolgend beschriebene Methode.

Am besten ist es, wenn Sie sich ein kleines Programmlisting auf dem Bildschirm ausgeben lassen, um gleich praktisch alles mitvollziehen zu können.

Nach der Listingausgabe steht der Cursor unter dem „R“ von Ready. Damit Sie nun nicht Ihr Listing zerstören, geben Sie als erstes eine Ziffer ein, die über der ersten Ziffer der höchsten Zeilennummer (also der zuletzt ausgegebenen) liegt. Wohlgemerkt nicht eine komplette Zeilennummer, sondern nur die erste Ziffer.

Nun halten Sie die SHIFT-Taste ge-

drückt und betätigen die Cursor-Steuertaste Pfeil nach oben. Sie werden daraufhin feststellen können, daß nun plötzlich ein zweiter Cursor auf dem Bildschirm zu sehen ist. Dies ist der COPY-Cursor. Solange Sie nun die SHIFT-Taste gedrückt halten, können Sie auch diesen mittels Cursorsteuerung frei über den Bildschirm bewegen.

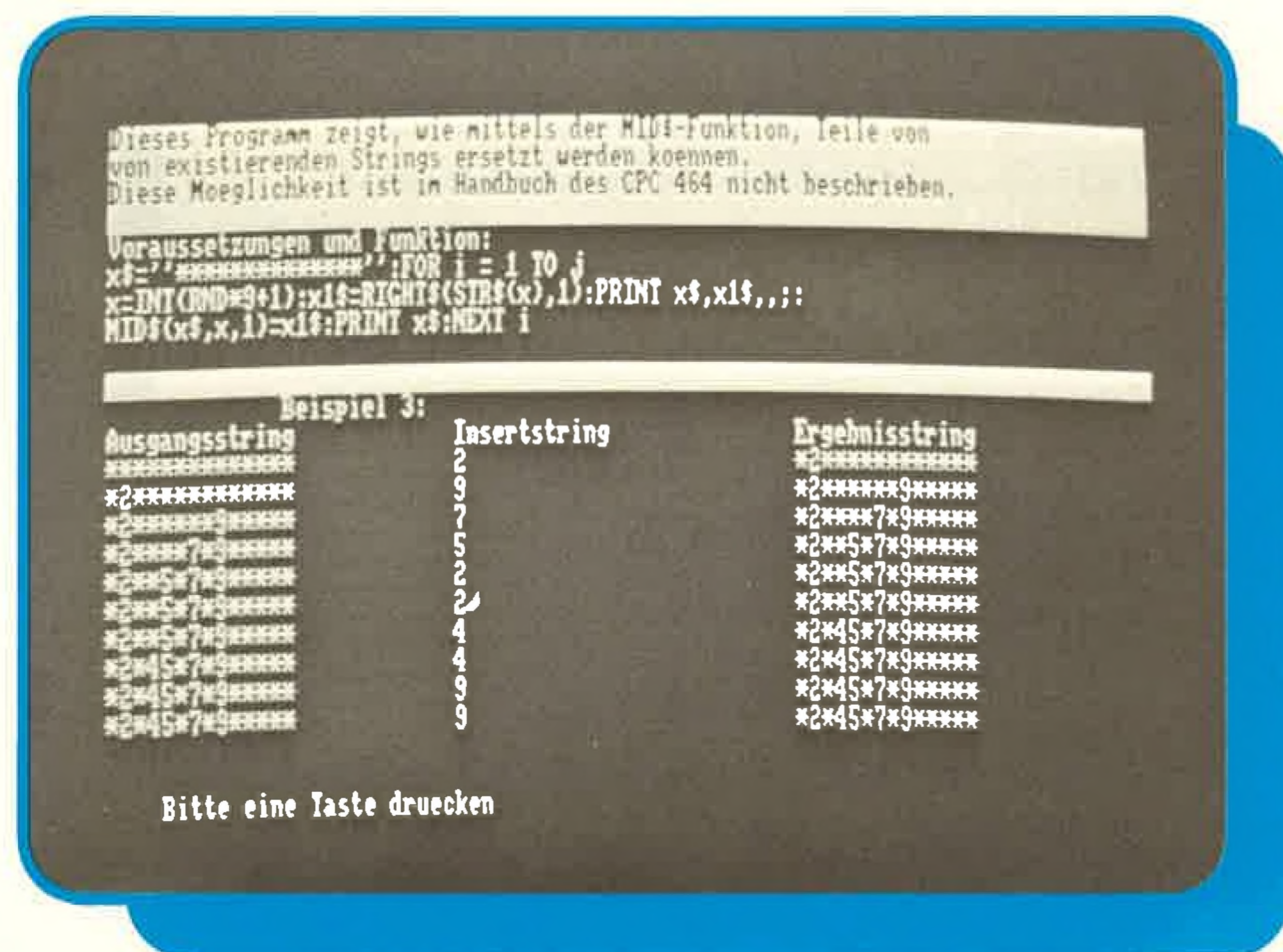
Setzen Sie diesen Copycursor nun z. B. auf die zweite Ziffer der letzten Basiczeilen-Nummer. Nun lassen Sie die Shift-Taste wieder los und betätigen die COPY-Taste. Wie Sie sehen, wird nun jeweils das Zeichen unter dem Copycursor an die Position des Normalcursors kopiert. Wenn Sie dies nun bis zum Ende der Basiczeile durchführen, haben Sie mit Ausnahme der Zeilennummer eine völlig identische Basiczeile auf dem Bildschirm stehen.

Durch Betätigung der ENTER-Taste übernehmen Sie dann diese Basiczeile in Ihr Programm.

Dies ist aber nur die einfachste Art, dieses Kopiertechnikeinsatzes. Sie können nämlich auch verschiedene Basiczeilen oder auch Teilinhalte beliebig bis zur maximal erlaubten Zeilenlänge zusammenfügen. Sie brauchen lediglich wieder die SHIFT-Taste gedrückt zu halten und können nun wieder mittels der Cursorsteuertasten „umeinanderfahren“ um dann wieder an der von Ihnen gewünschten Stelle mit dem Kopiervorgang fortzufahren.

Wenn Sie diese Technik etwas geübt haben, dann werden Sie von dieser Möglichkeit so begeistert sein, daß Sie bei fast allen Arbeiten am Computer darauf zurückgreifen. So z. B. auch beim Laden von Programmen von Diskette, nachdem Sie sich das Inhaltsverzeichnis haben ausgeben lassen. Die CPC-Mitteilung daß ein Programm nicht gefunden werden kann, weil man sich bei der Filenameneingabe vertippt hat, gehört damit der Vergangenheit an. Auch die „Verschönerung“ von Programmlistings nach Beendigung der eigentlichen Programmierarbeit wird damit zu einem Kinderspiel. Übersichtliche und leicht lesbare Listings werden Ihnen diese kleine Lernmühe, die Sie anfangs erbringen müssen „vergelt“.

Lothar Miedel



Listing: MID\$

```

100 '****      Die String-Einfuegung mittels MID$      ****
110 '
120 MODE 2:j=10:invers=&BB9C:warte=&BB06:t$="      Bitte eine Taste druecken"
130 WINDOW #1,1,80,1,8:WINDOW #2,1,80,25,25
140 WINDOW #3,1,80,11,24:WINDOW #4,1,80,5,9
150 '
160 CALL invers:CLS#0
170 PRINT"Dieses Programm zeigt, wie mittels der MID$-Funktion, Teile von
180 PRINT"von existierenden Strings ersetzt werden koennen.
190 PRINT"Diese Moeglichkeit ist im Handbuch des CPC 464 nicht beschrieben."
200 CLS#2:PRINT#2,t$:CALL warte:CLS #3
210 '-----
220 CLS#2:CLS #4:PRINT#3,,,:PRINT#3,"Beispiel 1:"
230 PRINT#3,"Ausgangsstring","Insertstring",,"Ergebnisstring"
240 '
250 CLS#4:PRINT #4,"Voraussetzungen und Funktion:
260 PRINT #4,"x$=' '*****':b$=' 'abc'
270 PRINT #4,"PRINT x$,b$,,:MID$(x$,3,3)=b$:PRINT x$
280 '
290 x$="*****":b$="abc"
300 PRINT#3,x$,b$,,:MID$(x$,3,3)=b$:PRINT#3,x$
310 '
320 CLS#2:PRINT#2,t$:CALL warte:CLS #3
330 '-----
340 CLS#2:CLS #4:PRINT#3,,,:PRINT#3,"Beispiel 2:"
350 PRINT#3,"Ausgangsstring","Insertstring",,"Ergebnisstring"
360 '
370 CLS#4:PRINT #4,"Voraussetzungen und Funktion:
380 PRINT #4,"x$=' '*****':FOR i = 1 TO j
390 PRINT #4,"x1$=x$:x2$=RIGHT$(STR$(INT(RND*9+1)),1)
400 PRINT #4,"MID$(x1$,i,1)=x2$:PRINT x$,x2$,,x1$:NEXT i
410 '
420 x$="*****":FOR i = 1 TO j
430 x1$=x$:x2$=RIGHT$(STR$(INT(RND*9+1)),1)
440 MID$(x1$,i,1)=x2$:PRINT#3,x$,x2$,,x1$:NEXT i
450 '
460 CLS#2:PRINT#2,t$:CALL warte:CLS #3
470 '-----
480 CLS#2:PRINT#3,,,:PRINT#3,"Beispiel 3:"
490 PRINT#3,"Ausgangsstring","Insertstring",,"Ergebnisstring"
500 '
510 CLS#4:PRINT #4,"Voraussetzungen und Funktion:
520 PRINT #4,"x$=' '*****':FOR i = 1 TO j
530 PRINT #4,"x=INT(RND*9+1):x1$=RIGHT$(STR$(x),1):PRINT x$,x1$,,:
540 PRINT #4,"MID$(x$,x,1)=x1$:PRINT x$:NEXT i
550 '
560 x$="*****":FOR i = 1 TO j
570 x=INT(RND*9+1):x1$=RIGHT$(STR$(x),1):PRINT#3,x$,x1$,,i
580 MID$(x$,x,1)=x1$:PRINT#3,x$:NEXT i
590 CLS#2:PRINT#2,t$:CALL warte:CALL invers:CLS #0

```


DEC\$ auf dem CPC 464

Obwohl dem Schneider-CPC-464 allseits Lobeshymnen bezüglich seines starken Basic-Befehlssatzes gesungen werden, darf nicht vergessen werden, daß kleine Mißklänge bei der „Melodie“ mitschwingen können. So gut die Programmierer der Firmware auch gearbeitet haben, kleine und durchaus vermeidbare Flüchtigkeitsfehler sind leider im Basicinterpreter vorhanden. Verschiedene, mir bereits bekannte Fehler wurden dann bei den Nachfolgemodellen wieder „ausgebügelt“. Dies kann dann bereits zu Kompatibilitätsproblemen führen.

Einige kleine Fehler sind nie so richtig bekannt geworden. Beispielsweise kann es bei verschachtelten Schleifen und der vorherigen Festlegung der Laufvariablen durch DEFINT vorkommen, daß das Schleifenziel überlaufen wird. Das gleiche Programm läuft aber auf dem CPC 664 und CPC 6128 völlig fehlerfrei. Es ist also durchaus möglich, daß trotz vermeintlicher syntaktisch richtig eingesetzter Befehle das Programm auf einem CPC 464 nicht korrekt abläuft, auf den beiden „Brüdern“ aber völlig problemlos „spielt“ und zu richtigen Ergebnissen führt. Im obengenannten „Schleifenproblem“ half bereits das

Weglassen der DEFINT-Anweisung. Wer auf derartige Probleme stößt, sollte um anderen CPC-Besitzern mühevoll Suchen zu ersparen, solche kleinen „bugs“ und am besten natürlich auch die Abhilfe, der einen oder anderen Fachzeitschrift mitteilen, damit durch deren Veröffentlichung die Informationen einem möglichst großen Kreis von CPC-Besitzern bekannt werden.

Aber auch andere „Problemchen“ sind noch nicht ausreichend bekannt, weshalb ich hier gleich auf den ersten Punkt dieser Art eingehe. Wenn man sich so wie ich, näher mit den CPC's beschäftigt, läßt man sich irgendwann auch einmal den gesamten Befehlssatz des CPC's ausgeben und stößt dann auch auf Befehle, die entweder im Handbuch überhaupt nicht aufgeführt sind, oder aber nur unzureichend erklärt werden.

DEC\$ ist einer dieser Befehle. Im CPC-464-Handbuch ist er zwar nicht aufgeführt, bei genauerer Betrachtung des Interpreterteiles aber ist er zu finden.

Für Profis:

In der TOKEN-Tabelle steht er als Befehlswort ab &E5E6. Der entsprechende Maschinenprogrammteil dieser Funktion ist ab &F8EA zu

finden. Lange Zeit wurde auch herumgerätselt, was dieser Befehl soll und kann. Fast alle Versuche führten aber zur Meldung: SYNTAX ERROR. Aber wie schon gesagt, der Befehl ist vorhanden, aber leider muß dieser syntaktisch falsch eingesetzt werden. Nun aber erst einmal zur Klärung, was der Befehl eigentlich bewirkt. Er ermöglicht die formatierte Ausgabe von Dezimalzahlen, wodurch ohne großen Aufwand zum Beispiel bei Tabellen auch alle Zahlen stellenrichtig untereinander ausgegeben werden können. Dieser Befehl ist auf allen CPC's vorhanden, muß aber beim CPC 464 mit einer öffnenden Klammer mehr versehen werden als eigentlich erforderlich. Wie dieser Befehl zum Einsatz kommen kann, zeigt das nachfolgende kleine Programmlisting anhand zweier Beispiele sehr deutlich. Besitzer der CPC 664 und 6128 brauchen die erste „öffnende“ Klammer nach dem Befehlswort DEC\$ in den Zeilen 250 und 300 nur wegzulassen, dann läuft dieses Demoprogramm auch auf diesen Rechnern. Andererseits brauchen CPC-464-Besitzer bei Programmen für den CPC 664 oder 6128 nur eine zusätzliche Klammer einzufügen und haben dann ein derartiges Programm auch für ihren Computer angepaßt.

Lothar Miedel

Der Befehl DEC\$ ist in allen CPC-Computern vorhanden. Leider ist im Betriebssystem des CPC 464 ein Fehler, wodurch dieser Befehl bei diesem Computer syntaktisch falsch angewendet werden muss. Der CPC 464 verlangt öffnend eine Klammer mehr als schließend! Das nachfolgende kleine Beispielprogramm zeigt, wie es klappt:

Beispiel 1:
Die Zahl x:

271.940658
528.612386
21.330127
175.138616
657.773343

x\$ durch DEC\$ gewonnen:

271.9407
528.6124
21.3301
175.1386
657.7733

Beispiel 2:

Die Zahl 123.22

123.2200

Der Befehl erlaubt also eine formatierte Ausgabe!
Bitte auch nachsehen unter: PRINT USING

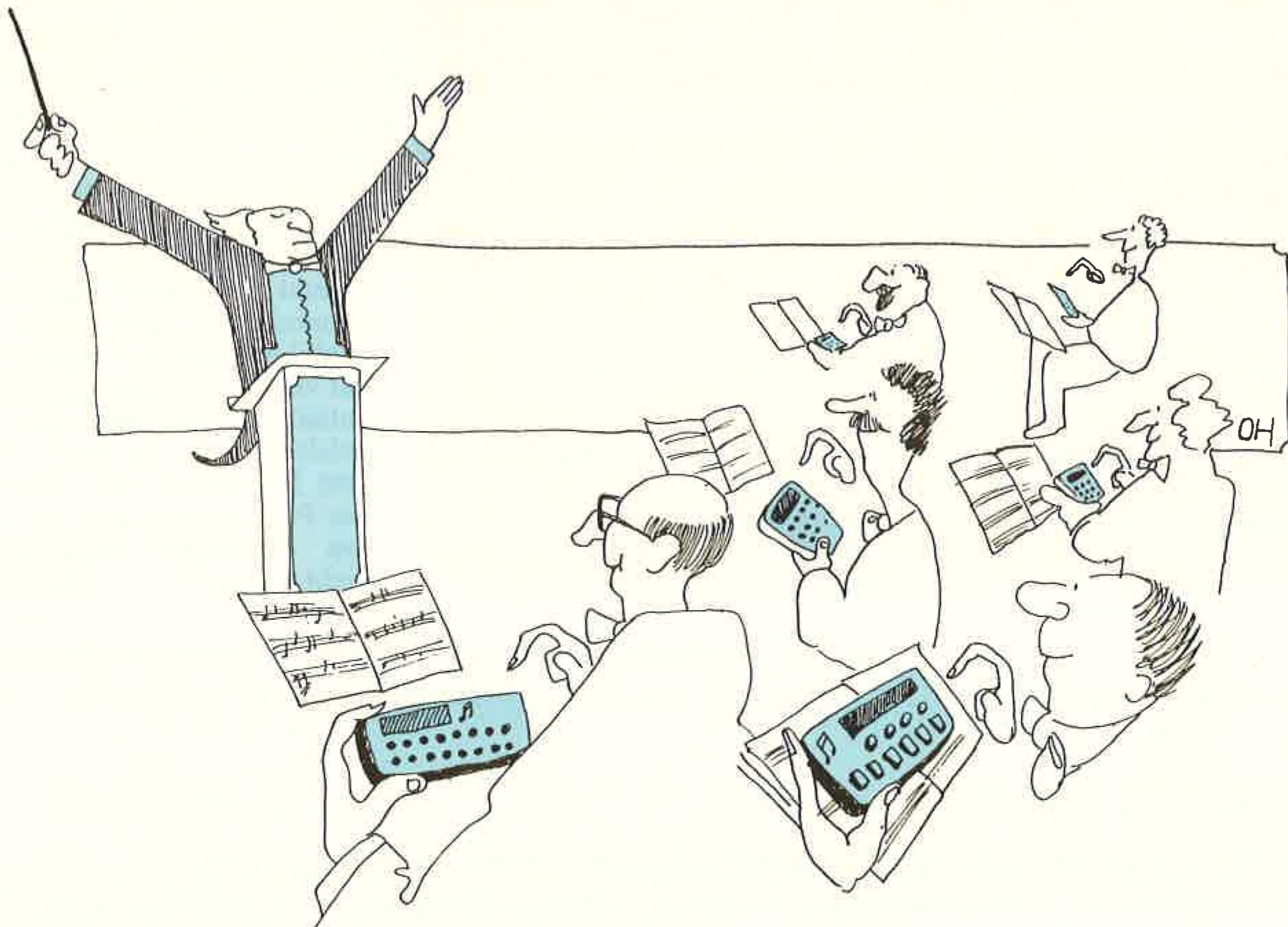
Hardcopy des Programmlaufes. Durch DEC\$ können Zahlen formatiert ausgegeben werden.

Listing: Der Befehl DEC\$ auf dem CPC 464

```

100 REM *****
110 REM * Der Befehl DEC$ auf dem CPC 464 *
120 REM *****
130 '
140 MODE 2:PRINT"Der Befehl DEC$":PRINT
150 PRINT"Der Befehl DEC$ ist in allen CPC-Computern vorhanden. Leider ist
160 PRINT"im Betriebssystem des CPC 464 ein Fehler, wodurch dieser Befehl
170 PRINT"bei diesem Computer syntaktisch falsch angewendet werden muss.
180 PRINT"Der CPC 464 verlangt oeffnend eine Klammer mehr als schliessend!
190 '
200 PRINT"Das nachfolgende kleine Beispielprogramm zeigt, wie es klappt:
210 PRINT:PRINT"Beispiel 1:
220 PRINT" Die Zahl x:",,"x$ durch DEC$ gewonnen:":PRINT
230 '
240 FOR i = 1 TO 5
250 x=RND(1)*1000:PRINT x,,;:x$=DEC$((x,"###.###"):PRINT x$
260 NEXT i
270 '
280 PRINT:PRINT"Beispiel 2:
290 PRINT:PRINT"Die Zahl 123.22";:
300 x$=DEC$((123.22,"###.###"):PRINT,x$
310 '
320 '
330 PRINT:PRINT"Der Befehl erlaubt also eine formatierte Ausgabe !
340 PRINT"Bitte auch nachsehen unter : PRINT USING

```



Quint – Ein Spiel mit dem Joystick

Eigentlich ist unser Ansinnen ja nicht, Ihnen Listings über Spiele abzudrucken. Daß wir es doch gemacht haben, hat verschiedene Gründe. Denken Sie nur einmal an Ihre Bekannten und Verwandten, denken Sie aber auch daran sich einmal etwas zu entspannen. Derartige Programme zeigen, wenn sie nicht gerade in Maschinensprache geschrieben sind, viele Programmier-Tricks die man gut in eigenen Programmen einsetzen kann.

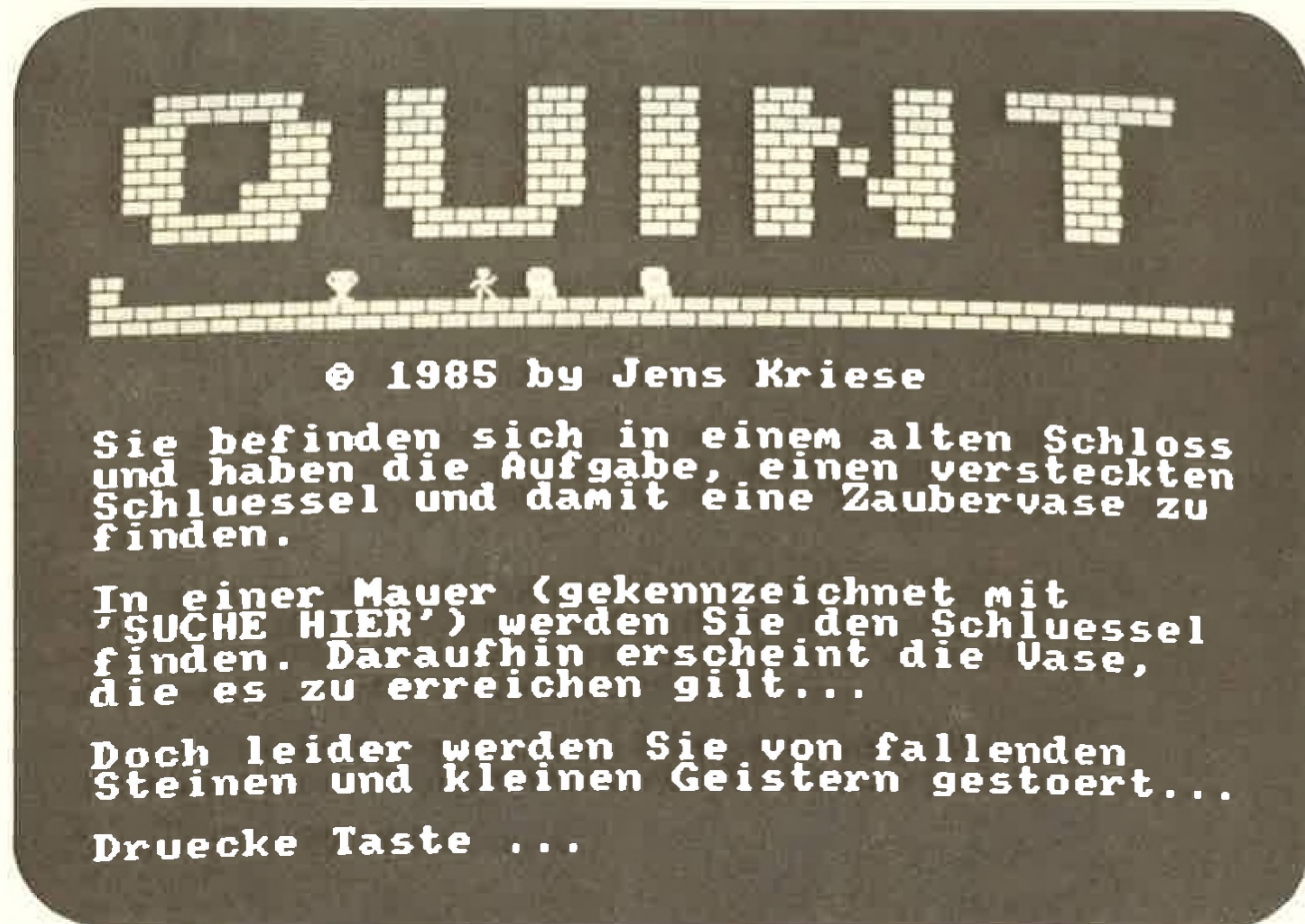
Mit ca. vier Seiten Listing und noch etwas Text, hoffen wir, daß wir in

Ihren Augen den Spieleteil nicht zu umfangreich gemacht haben. Aber ab und zu sollten Sie bei aller Ernsthaftigkeit der Programmierarbeiten auch einmal etwas entspannen. Hierzu eignet sich ein Spiel bestimmt. Spielen Sie aber nicht gleich in der höchstmöglichen Spielstufe, denn sonst bewirken Sie genau das Gegenteil, dann wird es „echt streßig“, auch bei diesem Spiel. Also „klein“ anfangen, wenn Sie entspannen wollen und ... vor allem auch wieder rechtzeitig aufhören! Sonst sind Sie bald in Richtung Las Vegas unterwegs, um Ihrem

Spieltrieb zu frönen. Mit diesem Spiel haben Sie aber vielleicht auch die Möglichkeit, Ihre Familienangehörigen an den Computer zu locken, die bisher nicht verstanden, was Sie an Ihrem Computer so begeistert. Da der CPC keine „eingebauten“ Spritemöglichkeiten hat, können diese erst durch Zusatzprogramme ermöglicht werden. Programmveröffentlichungen hierüber sind in verschiedenen Zeitschriften bereits erfolgt.

Wer sich also darüber informieren will, braucht nur etwas zeitlich zurückliegende Fachzeitschriften durchzublättern. Aber auch in zukünftigen Artikeln der verschiedensten Publikationen wird dieser Thematik bestimmt noch genügend Platz eingeräumt.

Das in diesem Heft abgedruckte Programm kommt ohne Sprites aus und ist trotzdem optisch sogar ansprechend. Wo die Tricks dieses Spieles liegen, bzw. wie die „Spielfiguren“ erzeugt werden, sollten Sie sich nach einem Abbruch des Programmes einmal im Listing ansehen. Bestimmt gibt Ihnen diese Art der Programmierung Anregungen, die Sie auch bei anderen Programmen sehr gut verwenden können. Das Listing enthält die komplette Bedienungsanleitung. Wer keinen Joystick besitzt kommt um ein Umschreiben des Programmes oder den Kauf eines derartigen Eingabemediums nicht herum. Jens Kriese/LM



Listing: QUINT - Ein Spiel mit Joysticksteuerung

```
100 ON ERROR GOTO 2170
110 ON BREAK GOSUB 2110
120 dat=0:st=0:Q=0:MODE 1:DIM hs$(16):DIM hsdat$(16):DIM hs(16)
```



```

130 FOR i=1 TO 16:hs$(i)="JENS KRIESE":hsdat$(i)="120485":hs(i)=0:NEXT
140 m$=CHR$(248):DIM x(41,25)
150 pu=0
160 INK 2,10,20:INK 1,17:INK 3,26:GOSUB 580:GOSUB 1070
170 DI:mox=10:moy=24:uox=30:uoy=24:FOR i=1 TO 24:LOCATE 1,i
180 PRINT"                               ";:NEXT
190 LOCATE 11,12:PEN 2:PRINT"EINEN MOMENT BITTE!";:PEN 1
200 t=24:schl=0:st=st+1
210 FOR i=1 TO 40:FOR ii=1 TO 24:x(i,ii)=0:NEXT:NEXT
220 FOR i= 2 TO 24
230 LOCATE 12,25:PRINT"  STUFE";st;"  ":LOCATE 26,25:PRINT"PUNKTE:";pu;
240 FOR ii= 1 TO st/2*RND(1)*1.4+1:x(RND(1)*39+1,i)=1:NEXT:NEXT
250 LOCATE 11,12:PRINT"                               ";
260 FOR i= 2 TO 24:FOR ii= 1 TO 40
270 IF x(ii,i)=1 THEN LOCATE ii,i:PRINT"]";
280 NEXT:NEXT
290 LOCATE 1,1:PEN 3:PRINT"[[[[[[[[[[[[[[[[[[[[SUCHE[HIER[[[[[[[[[[[[[[[[[[";:PEN 1
300 t=1011-st*10
310 :
320 mx=INT(RND(1)*30+9):my=INT(RND(1)*9+15)
330 vx=INT(RND(1)*30+9):vy=INT(RND(1)*14+9):x(vx,vy)=0
340 s1=INT(RND(1)*20+10)
350 IF schl=1 AND mx=vx AND my=vy THEN 560
360 LOCATE mx,my:PEN 3:PRINT m$;:PEN 1
370 EI:EVERY 50 GOSUB 930
380 :
390 GOSUB 450
400 IF mx=s1 AND my=1 THEN GOSUB 540
410 IF schl=1 AND mx=vx AND my=vy THEN GOTO 560
420 IF RND(1)>0.96 AND st>2 THEN GOSUB 660
430 IF st>5 THEN GOSUB 700
440 GOTO 380
450 :
460 DI:mx1=mx:my1=my
470 IF JOY(0)=1 AND x(mx,my-1)=0 AND my>=2 THEN my=my-1:GOTO 520
480 IF JOY(0)=2 AND x(mx,my+1)=0 AND my<=23 THEN my=my+1:GOTO 520
490 IF JOY(0)=4 AND x(mx-1,my)=0 AND mx>=2 THEN mx=mx-1:GOTO 520
500 IF JOY(0)=8 AND x(mx+1,my)=0 AND mx<=39 THEN mx=mx+1:GOTO 520
510 EI:RETURN
520 :
530 DI:LOCATE mx1,my1:PRINT" ";:LOCATE mx,my:PEN 3:PRINT m$;:PEN 1:EI:RETURN
540 :
550 LOCATE vx,vy:PEN 2:PRINT"\";CHR$(7);:PEN 1:schl=1:RETURN
560 :
570 pu=pu+t:GOTO 170
580 :
590 SYMBOL AFTER 90
600 SYMBOL 93,62,127,127,255,255,255,255,126
610 SYMBOL 91,239,239,239,0,254,254,254,0
620 SYMBOL 92,126,189,189,126,60,24,24,126
630 SYMBOL 123,124,254,214,254,254,127,127,85
640 SYMBOL 125,30,127,107,127,127,254,254,170
650 RETURN

```


[illegible]


```

1190 PRINT"                ";CHR$(164);" 1985 by Jens Kriese"
1200 PRINT
1210 PRINT"Sie befinden sich in einem alten Schloss";
1220 PRINT"und haben die Aufgabe, einen versteckten";
1230 PRINT"Schluessel und damit eine Zaubervase zu ";
1240 PRINT"finden. ";
1250 PRINT
1260 PRINT"In einer Mauer (gekennzeichnet mit ";
1270 PRINT"'SUCHE HIER') werden Sie den Schluessel ";
1280 PRINT"finden. Daraufhin erscheint die Vase, ";
1290 PRINT"die es zu erreichen gilt... ";
1300 PRINT
1310 PRINT"Doch leider werden Sie von fallenden ";
1320 PRINT"Steinen und kleinen Geistern gestoert...";
1330 LOCATE 1,25:PRINT"Druecke Taste ..."
1340 IF INKEY$="" THEN 1340
1350 IF dat=0 THEN GOSUB 1540
1360 LOCATE 1,10
1370 PRINT"Die Kontrolle erfolgt ueber Joystick 0. ";
1380 PRINT" ";
1390 PRINT"Die fallenden Steine: ] ";
1400 PRINT" ";
1410 PRINT"Die Zaubervase: ";
1420 PEN 2:PRINT"\";:PEN 1:PRINT" ";
1430 PRINT" ";
1440 PRINT"Die Geister : ";
1450 PEN 3:PRINT") {";:PEN 1:PRINT" ";
1460 FOR i=1 TO 9
1470 PRINT" ";:NEXT
1480 LOCATE 1,20:PRINT"Waehlen Sie bitte Ihre Spielstaerke ";
1490 INPUT"(0 bis 30) >";sps
1500 IF sps<0 OR sps>30 THEN 1360
1510 st=sps-1
1520 LOCATE 1,25:PRINT" ";
1530 RETURN
1540 :
1550 LOCATE 1,10
1560 PRINT"Wichtig fuer das High-Score: ";
1570 PRINT" ";
1580 PRINT"1 = Neue Datei ";
1590 PRINT"2 = Datei besteht bereits ";
1600 FOR i=1 TO 11:PRINT" ";:NEXT
1610 a$=INKEY$
1620 IF a$="1" THEN dat=1:RETURN
1630 IF a$<>"2" THEN 1610
1640 LOCATE 1,15:PRINT"Bitte legen Sie die Cassette/Disk mit"
1650 PRINT"Ihren High-Score-Werten ein und druecken"
1660 PRINT"Sie (PLAY) eine Taste..."
1670 IF INKEY$="" THEN 1670
1680 OPENIN"!quinthi":FOR i= 1 TO 15
1690 INPUT #9,hs$(i):INPUT #9,hsdat$(i):INPUT #9,hs(i)
1700 NEXT:CLOSEIN:RETURN
1710 :

```




Aufstellung: Vergleichstabelle der CPC-Systemadressen

Fragezeichen bei den Geraetetypen bedeuten, dass bisher keine entsprechenden Vergleichsadressen gefunden wurden.

Fragezeichen bei der Funktion bedeuten, dass der Sinn dieser Speicherstellen noch nicht eindeutig erkannt wurde.

Speicherstellen deren Adressen bei den drei CPC's uebereinstimmen, wurden nur in wenigen Ausnahmefaelen in die Tabelle aufgenommen. L. Miedel

464	664	6128	Funktion
AB80	A67C	A67C	HIMEM DEFAULT
ABFF	B0FF	B0FF	DEFAULT UPPER ROM BOUNDARY -1
AC00	AC00	AC00	FLAG IGNORE BLANKS
AC1C	AC01	AC01	FLAG FOR AUTO
AC1D	AC02	AC02	NEW LINE #
AC1F	AC04	AC04	STEP FOR AUTO
AC21	AC06	AC06	OUTPUT CHANNEL #
AC22	AC07	AC07	INPUT CHANNEL #
AC23	AC08	AC08	POS PRINTER
AC24	AC09	AC09	WIDTH FOR PRINTER
AC25	AC0A	AC0A	POS (TAPE) # OF CHARS
AC26	AC0C	AC0C	FLAG FOR FIRST FOR NEXT
AC27	AC0D	AC0D	STORE FOR NEXT VARIABLE 1
AC29	AC11	AC11	STORE FOR NEXT VARIABLE 2
AC2C	AC12	AC12	ADRESS FOR NEXT LB
AC2D	AC13	AC13	ADRESS FOR NEXT HB
AC2E	AC14	AC14	USED BY WHILE/WEND
AC30	AC16	AC16	USED BY ON
AC31	AC17	AC17	USED BY ON
AC32	AC18	AC18	?????
AC33	AC19	AC19	?????
AC34	AC1A	AC1A	LINE # FOR ON BREAK GOSUB (1)
AC35	AC1B	AC1B	LINE # FOR ON BREAK GOSUB (2)
AC36	AC1C	AC1C	BASIC PC ON BREAK
AC38	AC1E	AC1E	SOUND CHANNEL #1 (BIT 0)
AC44	AC2A	AC2A	SOUND CHANNEL #2 (BIT 1)
AC50	AC36	AC36	SOUND CHANNEL #3 (BIT 2)
AC5C	AC42	AC42	TIMER BLOCK #0
AC62	AC48	AC48	IM EVENT BLOCK #0
ACA4	AC8A	AC8A	EDIT BUFFER
ADA6	AD8C	AD8C	ERROR ADRESS
ADA8	AD8E	AD8E	PC ON ERROR BREAK
ADAA	AD90	AD90	LAST BASIC ERROR #
ADAB	AD92	AD92	CONTINUE POINTER
ADAD	AD94	AD94	BASIC PC ON STOP OR END
ADAF	AD96	AD96	ON ERROR ADRESS
ADB1	AD98	AD98	FLAG ON ERROR
ADB2	AD99	AD99	SOUND CHAN-STAT
ADB3	AD9A	AD9A	SOUND VOL-ENV
ADB4	AD9B	AD9B	SOUND TON-ENV
ADB5	AD9C	AD9C	SOUND PERIOD
ADB7	AD9E	AD9E	SOUND NOISE
ADB8	AD9F	AD9F	SOUND VOLUME
ADB9	ADA0	ADA0	SOUND TIME
ADBB	ADA2	ADA2	ENVELOPE TABLE ADRESS
ADBC	ADA3	ADA3	SOUND ENVELOPE ADRESS
ADCB	ADB2	ADB2	FAC USED BY POWER
ADD0	ADB7	ADB7	TABELLE SKALARE VARIABLE
AE04	ADEB	ADEB	TABELLE FOR FN
AE06	ADED	ADED	TABELLE FOR ARRAYS
AE26	AE0D	AE0D	?????
AE27	AE0E	AE0E	POINTER TO BASIC STACK
AE29	AE10	AE10	POINTER TO FN SUBPROG (1)
AE2B	AE12	AE12	POINTER TO FN SUBPROG (2)
AE2D	AE14	AE14	SAVE FOR SEMIKOLON ON PRINT
AE2E	AE15	AE15	LAST DATA LINE #
AE30	AE17	AE17	POINTER TO NEXT DATA
AE32	AE19	AE19	TEMP STORAGE BASIC STACK POINTER
AE34	AE1B	AE1B	PROGRAMM COUNTER ON RUN
AE36	AE1D	AE1D	BASIC PROG COUNTER PC
AE38	AE1F	AE1F	FLAG TRON/TROFF
AE39	AE20	AE20	FLAG USED ASSEMBLING A BASIC LINE
AE3A	AE21	AE21	BASIC PROG LINE FORMAT
AE3B	AE22	AE22	USED BY DELETE LINE #/LOWER ADR
AE3D	AE24	AE24	USED BY DELETE LINE #/UPPER ADR
AE3F	AE26	AE26	LOAD POINTER WHILE LOAD
AE41	AE28	AE28	LOAD MERGE FLAG
AE42	AE29	AE29	LOAD/CHAIN FLAG
AE43	AE2A	AE2A	USED BY LOAD CHAIN
AE45	AE2C	AE2C	FLAG FILE READ PROTECTED
AE46	AE2D	AE2D	NUMBER EDIT BUFFER (1)
AE53	AE3A	AE3A	NUMBER EDIT BUFFER (2)
AE58	AE3F	AE3F	NUMBER EDIT BUFFER (3)
AE67	AE4E	AE4E	NUMBER EDIT BUFFER INDEX (1)
AE68	AE4F	AE4F	NUMBER EDIT BUFFER (4)
AE69	AE50	AE50	NUMBER EDIT BUFFER INDEX (2)
AE6E	AE52	AE52	TEMP STORE FOR CHAR
AE6F	AE53	AE53	IM BUFFER UMWANDLUNG ASCII
AE70	AE54	AE54	NUMBER EDIT BUFFER INDEX (3)
AE72	AE55	AE55	ADRESS OF CALLED ROUTINE
AE74	AE57	AE57	ROM SELECTION ON CALL
AE75	AE58	AE58	SAVE HL ON CALL
AE77	AE5A	AE5A	SAVE SP ON CALL
AE79	AE5C	AE5C	ZONE FOR TAB
AE7A	AE5D	AE5D	FLAG FOR PRINT USING
AE7B	AE5E	AE5E	POINTER HIMEM FOR BASIC
AE7D	AE60	AE60	POINTER HIMEM FOR SYMBOL AFTER
AE7F	AE62	AE62	POINTER LOW MEMORY BOUNDARY
AE81	AE64	AE64	POINTER START OF BASIC-PRG -1
AE83	AE66	AE66	POINTER END OF BASIC-PRG
AE85	AE68	AE68	POINTER START VAR TABLE
AE87	AE6A	AE6A	POINTER START DIM'D VAR
AE89	AE6C	AE6C	POINTER UPPER END DIM
AE8B	AE6F	AE6F	START OF BASIC STACK
B08B	B06F	B06F	POINTER BASIC STACK
B091	B075	B075	TAPE BUFFER FLAG
B092	B076	B076	POINTER TAPE BUFFER/LOWER END
B094	B078	B078	POINTER TAPE BUFFER/UPPER END

B096	????	????	HIMEM FOR SYMBOL AFTER
B098	B07A	B07A	USED BY GARBAGE COLLECT
B09A	B07C	B07C	POINTER TO START OF STRING STACK
B09C	B07E	B07E	STRING STACK
B0BA	B09C	B09C	TEMPORARY STRING DESCRIPTOR
B0BD	????	????	USED ON GARBAGE COLLECT
B0BF	????	????	SAVE ON GARBAGE COLLECT
B0C1	B09F	B09F	VARTYPE
B0C2	B0A0	B0A0	FLOATING POINT ACU/FAC
B0C3	B0A1	B0A1	FAC + 1
B0C4	B0A2	B0A2	FAC + 2
B0C5	B0A3	B0A3	FAC + 3
B100	B82D	B82D	KL INTERRUPT SERVICE QUEUE
B101	B82E	B82E	KL INTERRUPT PENDING QUEUE
B102	B82F	B82F	KL INTERRUPT SERVICE CHAIN
B104	B831	B831	KL INTERRUPT SERVICE CLASS
B105	B832	B832	KL SAVE SP ON INTERR SERVICE
B187	B8B4	B8B4	KL TIME BYTE 0+1
B189	B8B6	B8B6	KL TIME BYTE 2+3
B18B	B8B8	B8B8	KL TIME BYTE 4
B18C	B8B9	B8B9	KL FRAME FLY LIST POINTER
B18E	B8BB	B8BB	KL FAST TICKER LIST POINTER
B190	B8BD	B8BD	KL POINTER TO TICK LIST
B192	B8BF	B8BF	KL SLOW TICKER COUNT
B193	B8C0	B8C0	KL SYNC EVENT QUEUE
B194	B8C1	B8C1	KL SYNC EVENT QUEUE +1
B195	B8C2	B8C2	KL EVENT CLASS
B196	B8C3	B8C3	KL TEMP STORE EXT COMMAND NAME
B1A6	B8D3	B8D3	KL RSX QUEUE
????	????	B8D5	?????
B1A8	B8D5	B8D6	KL ROM SELECT ADRESS
B1A9	B8D6	B8D7	KL ROM START
B1AB	B8D8	B8D9	KL ROM STATE TO CALL
B1AA	B8D9	B8DA	START OF ROM HIGH BYTE (C0)
B1C8	B7C3	B7C3	SCR SCREEN MODE
B1C9	B7C4	B7C4	SCR SCREEN START
B1CA	B7C5	B7C5	SCR OFFSET TO SCREEN START
B1CB	B7C6	B7C6	SCR BASE OF RAM FOR SCREEN
B1CC	B7C7	B7C7	SCR PIXELS WRITE
B1CD	B7C8	B7C8	SCR PIXELS WRITE SET JUMP ADRESS
B1CF	????	????	SCR CURRENT PIXEL BIT MAP
B1D7	B7D2	B7D2	SCR TIME FLASHING PERIOD 1
B1D8	B7D3	B7D3	SCR TIME FLASHING PERIOD 2
B1D9	B7D4	B7D4	SCR COLOR TABLE FLASH 1 (BORDER)
B1EA	B7E5	B7E5	SCR COLOR TABLE FLASH 2 (BORDER)
B1FB	B7F6	B7F6	SCR FLAG WHICH FLASH PERIOD (1 OR 2)
B1FC	B7F7	B7F7	SCR FLAG
B1FD	B7F8	B7F8	TIME COUNT/CURRENT FLASH PERIOD
B1FE	B7F9	B7F9	SCR FRAME FLY LIST
B20C	B6B5	B6B5	TXT CURRENT TEXT STREAM
B20D	B6B6	B6B6	TXT TABLE (STREAM-PARAMETER 8*15)
B285	B726	B726	TXT CURSOR COLUMN/ROW
B287	B728	B728	TXT WINDOW FLAG
B288	B729	B729	TXT ROW-WINDOW LEFT UPPER CORNER
B289	B72A	B72A	TXT COLUMN-WINDOW LEFT UPPER CORNER
B28A	B72B	B72B	TXT ROW WINDOW RIGHT BOTTTUM CORNER

B28B	B72C	B72C	TXT COLUMN-WINDOW RIGHT BOTTOM CORNER
B28C	B72D	B72D	TXT ROLL COUNT
B28D	B72E	B72E	TXT CURSOR ENABLE FLAG (USER)
B28E	????	????	TXT FLAG VDU ENABLE
B28F	B72F	B72F	TXT PEN INK
B290	B730	B730	TXT PAPER INK
B291	B731	B731	TXT ADRESS OF BACK-/FOREGROUND ROUTINE
B293	B733	B733	TXT FLAG GRAPHIC CHAR WRITE
B294	B734	B734	TXT FIRST CHAR OF USER MATRIX TABLE
B295	B735	B735	TXT FLAG USER MATRIX TABLE
B296	B736	B736	TXT START OF USER MATRIX TABLE
B298	B738	B738	TXT BUFFER FOR UNPACKED CHAR MATRIX
B2B8	B758	B758	TXT CONTROL CODE PUFFER INDEX
B2B9	B759	B759	TXT CONTROL CODE BUFFER
B2C3	B763	B763	CONTROL CODE TABLE
B328	B693	B693	GRA USER ORIGIN X
B32A	B695	B695	GRA USER ORIGIN Y
B32C	B697	B697	GRA CURSOR X
B32E	B699	B699	GRA CURSOR Y
B330	B69B	B69B	GRA WINDOW WIDTH X-LEFT
B332	B69D	B69D	GRA WINDOW WIDTH X-RIGHT
B334	B69F	B69F	GRA WINDOW HEIGHT Y-TOP
B336	B6A1	B6A1	GRA WINDOW HEIGHT Y-BOTTOM
B338	B6A3	B6A3	GRA PEN INK
B339	B6A4	B6A4	GRA PAPER INK
B33A	B6A5	B6A5	GRA TEMP STORE 1
B33C	B6A7	B6A7	GRA TEMP STORE 2
B33E	B6A9	B6A9	GRA TEMP STORE 3
????	B6AA	B6AA	?????
B340	B6AB	B6AB	GRA TEMP STORE 4
????	B6AA	B6AA	?????
????	B6AC	B6AC	?????
B342	B6AD	B6AD	GRA TEMP STORE X ON DRAW
????	B6AE	B6AE	?????
B344	B6AF	B6AF	GRA TEMP STORE Y ON DRAW
????	B6AE	B6AE	?????
????	B6B0	B6B0	?????
B346	B6B1	B6B1	GRA TEMP FLAG
????	B6B2	B6B2	?????
B348	B6B3	B6B3	?????
B349	B6B4	B6B4	?????
B34C	B496	B496	KM KEY NORMAL ENTRY
B39C	B4E6	B4E6	KM KEY SHIFT ENTRY
BE3C	B536	B536	KM KEY CONTROL ENTRY
B43C	B586	B586	KM KEY REPEAT MAP
B446	B590	B590	KM KEY EXP BUFFER
B4DE	B628	B628	KM EXPANSION STRING FLAG/COUNT
B4DF	B629	B629	KM EXP BUFFER FLAG
B4E0	B62A	B62A	KM KEYBOARD PUT BACK CHAR
B4E1	B62B	B62B	KM POINTER TO F-KEY EXP BUFFER
B4E3	B62D	B62D	KM POINTER TO END OF EXP B+1
B4E5	B62F	B62F	KM POINTER EXP PUFFER LO-BYTE
B4E6	B630	B630	KM POINTER EXP-BUFFER HI-BYTE
B4E7	B631	B631	KM CAPS LOCK STATE
B4E8	B632	B632	KM SHIFT LOCK STATE
B4E9	B633	B633	KM KEY REPEAT SPEED

B4EA	B634	B634	KM KEY STARTUP DELAY
B4EB	B635	B635	KM KEY STATE MAP
B4ED	B637	B637	CONTAINS CTRL AND SHIFT BITS
B4F1	B63B	B63B	JOYSTICK 2
B4F3	B63D	B63D	BIT 2 = BREAK KEY
B4F4	B63E	B63E	JOYSTICK 1
B4F5	B63F	B63F	KM KEY CHANGE STATE MAP
B4FF	B649	B649	KM KEY LAST CYCLE STATE MAP
B501	B64B	B64B	SHIFT AND CAPS BITS ?
B509	B653	B653	KM TIME COUNT FOR REPEAT SPEED
B50A	B654	B654	?????
B50B	B655	B655	?????
B50C	B656	B656	KM BREAK ENABLE FLAG
B50D	B657	B657	KM EVENT BLOCK BREAK
B51D	B1B9	B1B9	START SOUND QUEUE CHANNEL A
B520	B1BC	B1BC	?????
B522	B1BE	B1BE	NOISE PERIOD CHANNNEL A
B539	B1D5	B1D5	?????
B53C	B686	B686	?????
B53E	B688	B688	?????
B540	B68A	B68A	ADR-1 KEY TRANSLATE TABLE
B541	B68B	B68B	KM TRANSLATE NORMAL ENTRY POINTER
B543	B68D	B68D	KM TRANSLATE SHIFT ENTRY POINTER
B545	B68F	B68F	KM TRANSLATE CONTROL ENTRY POINTER
B547	B691	B691	KM REPEAT KEY POINTER TO TABLE
B550	B1EC	B1EC	SOUND FLAG
B551	B1ED	B1ED	SOUND SAVE FOR AKTIVE SOUND
B552	B1EE	B1EE	SOUND CHANNEL BITS OF AKTIVE SOUND
B554	B1F0	B1F0	SOUND RENDENZVOUS BYTE ?
B55C	B1F8	B1F8	SOUND QUEUE CHANNEL A
B59B	B237	B237	SOUND QUEUE CHANNEL B
B5DA	B276	B276	SOUND QUEUE CHANNEL C
B60A	B2A6	B2A6	SOUND AMPLITUDE ENVELOPE
B619	B2B5	B2B5	?????
B6FA	B396	B396	SOUND TONE ENVELOPE
B800	B118	B118	CAS MESSAGE FLAG
B801	B119	B119	CAS FLAG ?
B802	B11A	B11A	CAS FILE TYPE ON READ
B803	B11B	B11B	CAS START ADRESS INPUT BUFFER LOW
B805	B11D	B11D	CAS POINTER INPUT BUFFER HIGH
B807	B11F	B11F	CAS FILE HEADER INPUT
B80D	B071	B071	POINTER LOW END \$-SPACE
B80F	B073	B073	POINTER UPPER BOUND \$-SPACE
B817	B12F	B12F	CAS IN BLOCK NUMBER
B818	B130	B130	CAS IN LAST BLOCK FLAG
B819	B131	B131	CAS IN FILE TYPE
B81A	B132	B132	CAS IN DATA LENGTH
B81C	B134	B134	CAS IN DATA LOCATION
B81E	B136	B136	CAS IN FIRST BLOCK FLAG
B81F	B137	B137	CAS IN USER FIELDS
B847	B15F	B15F	CAS OUTPUT FILE TYPE WRITE
B848	B160	B160	CAS OUT DIRECT POINTER DATA LO
B84A	B162	B162	CAS OUT DIRECT POINTER DATA HIGH
B84C	B164	B164	CAS OUT DIRECT/FILENAME
B85C	B174	B174	CAS OUT DIRECT/BLOCK NUMBER
B85D	B175	B175	CAS OUT DIRECT/LAST BLOCK FLAG

B85E	B176	B176	CAS OUT DIRECT/FILE TYPE
B85F	B177	B177	CAS OUT LEN OF DATA
B861	B179	B179	CAS OUT DIRECT/DATA LOCATION
B863	B17B	B17B	CAS OUT DIRECT/FIRST BLOCK FLAG
B864	B17C	B17C	CAS OUT DIRECT/TOTAL LENGTH OF DATA
B866	B17E	B17E	CAS OUT DIRECT/ENTRY FOR HEADER
B88C	B1A4	B1A4	CAS OUT FILENAME
B89C	B1B4	B1B4	CAS OUT BLOCK NUMBER
B89D	B1B5	B1B5	CAS OUT FILE TYPE
B89F	B1B7	B1B7	CAS OUT DATA LENGTH
B8A1	B1B9	B1B9	CAS OUT DATA LOCATION
B8A3	B1BB	B1BB	CAS OUT FIRST BLOCK FLAG
B8A4	B1BC	B1BC	CAS OUT USER FIELDS LOGICAL LENGTH
B8A6	B1BE	B1BE	CAS OUT USER FIELDS ENTRY ADRESS FOR MC
B8CC	B1E4	B1E4	?????
B8CD	B1E5	B1E5	?????
B8CE	B1E6	B1E6	?????
????	B1E7	B1E7	?????
B8D0	B1E8	B1E8	?????
B8D1	B1E9	B1E9	?????
B8D2	B1EA	B1EA	?????
B8D3	B1EB	B1EB	?????
B8DC	B114	B114	EDI CURSOR ON FLAG
B8DD	B115	B115	EDI INSERT/OVERWRITE FLAG
B8DE	B116	B116	EDI COPYCURSOR POS/COLUMN AND ROW
B8E4	B100	B100	RANDOM NUMBER BYTE 0+1
????	B101	B101	?????
B8E6	B102	B102	RANDOM NUMBER BYTE 2+3
B8E8	B104	B104	FAC 1
B8ED	B109	B109	FAC 2
B8F2	B10E	B10E	FAC 3
B8F7	B113	B113	FLAG DEG/RAD
B939	B941	B941	INTERRUPT ENTRY RST 7
B978	B970	B970	?????
????	B972	B972	?????
B97C	B984	B984	LOW ROM OR RAM
????	B9B0	B9B0	?????
B9B1	B9B9	B9B9	JP FAR CALL
B9B2	B98A	B98A	LOW JUMP
B9B9	B9C1	B9C1	KL FAR ICALL
B9BF	B9C7	B9C7	FAR CALL
BA10	BA17	BA17	JP TO A SIDEWAYS ROM
BA16	BA1D	BA1D	CALL TO A SIDEWAYS ROM
BA2E	BA35	BA35	FIRM JUMP LOWER ROM
????	BA45	BA45	?????
????	BA46	BA46	?????
BA4A	BA51	BA51	KL L ROM ENABLE CONT'D
BA54	BA58	BA58	KL L ROM DISABLE CONT'D
BA5E	BA5F	BA5F	KL CURRENT UPPER ROM ENABLE
BA68	BA66	BA66	KL CURRENT UPPER ROM DISABLE
BA72	BA70	BA70	KL ROM RESTORE CONT'D
BA7E	BA79	BA79	KL SELECT AN UPPER ROM
BA83	BA7E	BA7E	KL PROBE ROM CONT'D
BA8C	BA87	BA87	KL STORE PREVIOUS ROM SELECTION
BAA2	BA9D	BA9D	KL 'CURR SELECTION CONT'D
BAA6	BAA1	BAA1	KL LDIR ROMS DISABLED

BAAC	BAA7	BAA7	KL LDDR ROMS DISABLED
????	BAAD	BAAD	?????
????	BAC2	BAC2	?????
BACB	BAC6	BAC6	RAM LAM
BADC	BAD7	BAD7	RAM LAM (IX)
????	BC0F	BC0F	?????
BD3A	BD5B	BD5E	EDI LINE EDITOR
BD3D	BD5E	BD61	COPY 5 BYTES
BD40	BD61	BD64	REAL ARITH CREAL 1
BD43	BD64	BD67	REAL ARITH CREAL 2
BD46	BD67	BD6A	REAL ARITH CINT
BD49	BD6A	BD6D	REAL ARITH 1 (?)
BD4C	BD6D	BD70	REAL ARITH FIX
BD4F	BD70	BD73	REAL ARITH INT
BD55	BD76	BD79	REAL ARITH 2 (?)
BD58	BD79	BD7C	REAL ARITH ADD
BD5B	BD7C	BD7F	REAL ARITH SUB 1 (?)
BD5E	BD7F	BD82	REAL ARITH SUB 2 (?)
BD61	BD82	BD85	REAL ARITH MULT
BD64	BD85	BD88	REAL ARITH DVD
BD67	????	????	REAL ARITH 3 (?)
BD6A	BD8B	BD8E	REAL ARITH COMPARE
????	BD8E	BD91	?????
BD70	BD91	BD94	REAL ARITH SGN
BD73	BD94	BD97	REAL ARITH SET DEG/RAD
BD76	BD97	BD9A	REAL ARITH PI
BD79	BD9A	BD9D	REAL ARITH SQUARE
BD7C	BD9D	BDA0	REAL ARITH EXP
BD7F	BDA0	BDA3	REAL ARITH LOG
BD82	????	BDA6	REAL ARITH LOG10'
BD85	BDA6	BDA9	REAL ARITH GET EXP
BD88	BDA9	BDAC	REAL ARITH SIN
BD8B	BDAC	BDAF	REAL ARITH COS
BD8E	BDAF	BDB2	REAL ARITH TAN
BD91	BDB2	BDB5	REAL ARITH ATN
BD94	BDB5	BDB8	REAL ARITH ADD
BD9A	BDB8	BDBE	REAL ARITH SEED RANDOM NUMBER
BD9D	BD7C	BD7F	REAL ARITH RANDOMIZE
BDA0	BD88	BD8B	REAL ARITH GET LAST RANDOM NUMBER

erstellt von Lothar Miedel, Stand: 29.10.1985



Die Diskette zum Hekt!

Sparen Sie sich Zeit und Ärger!
Die Programme aus diesem Sonderheft können Sie auch auf einer 3-Zoll-Diskette für Ihren Schneider-Computer erhalten.
Das Abtippen der Programme kostet nicht nur Zeit und Nerven, sondern man macht meist auch noch Tippfehler, die eine zeitaufwendige und ärgerliche Fehlersuche zur Folge haben.
Unter diesem Gesichtspunkt hat sich die Diskette schnell bezahlt gemacht.
Am besten bestellen Sie gleich die Schneider-Diskette zum Sonderheft!
Bestellnummer: 0179-3
Preis: 35 DM

FRANZIS SOFTWARE SERVICE

Franzis'

Franzis-Verlag, Software-Service, Karlstraße 37-41, 8000 München 2,
Telex 5 22 301, Telefax (089) 5117-379, BTX-Leitseite * 30503 #
Telefon direkt: (089) 5117-331

Pascal in der Methode Busch

Nie wird der zweite Schritt vor dem ersten gemacht. Das ist der unbestreitbare Vorzug der „Methode Busch“. Der Autor kann sich nur zu gut daran erinnern, wie er Pascal von Anfang an lernte und weiche Stolperdrähte er erst einmal wegräumen mußte bis schließlich der Pascal-Weg mühelos gangbar war. Systematisch erklärt der Autor zunächst die wichtigsten Sprach-elemente. Diese werden anhand von praktischen Beispielen eingeübt und laufend wiederholt. Ein Grundwortschatz in Pascal entsteht. Kleine Programme werden geschrieben, die sofort nützlich für die praktische Arbeit sind. Das Selbstbewußtsein steigt. Der Neuling ist erstaunt, wie einfach, schlicht und spannend der Autor Pascal-Unterricht gestaltet.

Und was kann der Leser und Benutzer dieses Buches am Ende? Das exakte Definieren von Algorithmen hat er gelernt. Er beherrscht einen Basismfang von Pascal. Grammatik und Syntax sind geläufig. Er hat die unbändige Lust auf mehr und der Karriere als Pascal-Programmierer steht nichts mehr im Wege.

Der sichere Einstieg in Pascal



Der leichte Weg
zum selbständigen
Programmieren
in Pascal.

Von **Rudolf Busch**.

192 Seiten,
52 Abbildungen,
Lwstr-geb. DM 48.-
ISBN
3-7723-7861-7

Franzis'

der große Fachverlag für angewandte Elektronik und Informatik

Franzis-Verlag, München



Alles über Mikro- Compu



DAS BASIC-SONDERHEFT
BASIC-Programme für Commodore-, Apple-, MS-DOS- und andere Rechner
NEU
Das BASIC-SONDERHEFT
BASIC-Programme und Programmiertips aus allen Anwendungsbereichen, u. a. für Commodore-, Apple- und MS-DOS-Rechner. In der schon erschienenen Ausgabe wurden z. T. verbesserte und für andere Rechnerarten angepaßte, z. B. Datenbank-Programm, Texteditor, Expertensystem u. a. für alle ernsthaften Anwender von Computern.
20 Seiten, 24 DM



DAS MODEM-SONDERHEFT
Datenübertragung mit Mikrocomputern
Das MC-MODEM-Sonderheft behandelt Hard- und Software zur Datenübertragung per Telefon – Schaltungstechnik von Modems. Übertragungs-Software, Grundlagen der Datenfernübertragung, Schnittstellen an Computern, Übertragungsprotokolle, öffentlich zugängliche Datenbanken. Überwiegend Aufsätze aus MC. Für Profis und Hobbyisten.
64 Seiten, 16 DM.



Klartext für den C 64
Einführung in die Maschinensprache
NEU
Ein stark praxisorientierter Lehrgang über das Programmieren des C 64 in Maschinensprache. Das Heft ist eine überarbeitete Zusammenfassung der gleichlautenden FUNKSCHAU-Serie. Neu hinzugekommen ist die achteilige Serie „Schach dem Video-Chip“, die zu einem Maschinenprogramm für hochauflösende Grafik führt.
Für Anfänger.
4 Seiten, 18 DM



ELEKTRONIK-Sonderheft Daten-Kommunikation
NEU
Alles Wichtige, was die ELEKTRONIK bisher über dieses Spezialgebiet schrieb, finden Sie jetzt komplett in einem Heft. Damit gibt der Franzis-Verlag allen Interessierten ein vielseitiges Informationsmittel in die Hand, das alle wesentlichen Aspekte der Datenübertragung transparent macht.
3., ergänzte Auflage.
Für Einsteiger und „alte Hasen“.
160 Seiten, 23 DM

Kennenlern-Angebot

für eine von vier Franzis-Zeitschriften
zum vorteilhaften Jahresabonnementspreis

Ich möchte Ihre Zeitschrift (nachstehend angekreuzt) kostenlos probelesen. Informiere ich Sie nach Erhalt des Heftes nicht anders, abonniere ich diese Zeitschrift.

Die Bestellung kann ich innerhalb von 10 Tagen nach Erhalt des kostenlosen Heftes beim Franzis-Verlag, Postf. 37 02 80, 8000 München 37, widerrufen. Zur Wahrung der Frist genügt rechtzeitiges Absenden.

Datum	Unterschrift
Wenn Sie nichts mehr von mir hören, möchte ich ab _____ Ihre Zeitschrift	
☐ Elektronik	26 Hefte pro Jahr, Jahresabonnementspreis DM 126,-, im Ausland DM 150,-
☐ Funkschau	26 Hefte pro Jahr, Jahresabonnementspreis DM 105,-, im Ausland DM 129,-
☐ ELO	12 Hefte pro Jahr, Jahresabonnementspreis DM 55,20, im Ausland DM 64,20
☐ MC	12 Hefte pro Jahr, Jahresabonnementspreis DM 66,-, im Ausland DM 72,-

abonnieren. Senden Sie die Hefte an folgende Anschrift:

Name/Vorname	
Beruf	
Straße	
PLZ/Ort	
Datum	Unterschrift

In den genannten Abonnementspreisen sind sämtliche Nebenkosten, einschließlich Porto, enthalten (Preis Stand 1/1986).
Die Kündigung ist jederzeit zum Ende des bezahlten Zeitraumes möglich.
Die Abonnementsgebühr ist nach Erhalt der Rechnung fällig.

G7C2

Kennenlern-Angebot

für eine von vier Franzis-Zeitschriften
zum vorteilhaften Jahresabonnementspreis

Ich möchte Ihre Zeitschrift (nachstehend angekreuzt) kostenlos probelesen. Informiere ich Sie nach Erhalt des Heftes nicht anders, abonniere ich diese Zeitschrift.

Die Bestellung kann ich innerhalb von 10 Tagen nach Erhalt des kostenlosen Heftes beim Franzis-Verlag, Postf. 37 02 80, 8000 München 37, widerrufen. Zur Wahrung der Frist genügt rechtzeitiges Absenden.

Datum	Unterschrift
Wenn Sie nichts mehr von mir hören, möchte ich ab _____ Ihre Zeitschrift	
☐ Elektronik	26 Hefte pro Jahr, Jahresabonnementspreis DM 126,-, im Ausland DM 150,-
☐ Funkschau	26 Hefte pro Jahr, Jahresabonnementspreis DM 105,-, im Ausland DM 129,-
☐ ELO	12 Hefte pro Jahr, Jahresabonnementspreis DM 55,20, im Ausland DM 64,20
☐ MC	12 Hefte pro Jahr, Jahresabonnementspreis DM 66,-, im Ausland DM 72,-

abonnieren. Senden Sie die Hefte an folgende Anschrift:

Name/Vorname	
Beruf	
Straße	
PLZ/Ort	
Datum	Unterschrift

In den genannten Abonnementspreisen sind sämtliche Nebenkosten, einschließlich Porto, enthalten (Preis Stand 1/1986).
Die Kündigung ist jederzeit zum Ende des bezahlten Zeitraumes möglich.
Die Abonnementsgebühr ist nach Erhalt der Rechnung fällig.

GARANTIE: Wenn Sie von unserem Kennenlern-Angebot Gebrauch machen, können Sie innerhalb von 10 Tagen nach Erhalt des kostenlosen Heftes das Abonnement widerrufen! Zur Wahrung dieser Frist genügt rechtzeitiges Absenden an den Franzis-Verlag, Postfach 37 02 80, 8000 München 37. Ein Jahresabonnement kostet: ELEKTRONIK DM 126,-, Ausland DM 150,-; FUNKSCHAU DM 105,-, Ausland DM 129,-; ELO DM 55,20, Ausland DM 64,20; MC DM 66,-, Ausland DM 72,-. Verbilligtes Jahresabonnement für Auszubildende und Studenten: ELEKTRONIK DM 108,-, Ausland DM 132,-; FUNKSCHAU DM 91,20, Ausland DM 115,80; ELO DM 48,-, Ausland DM 57,-; MC DM 54,-, Ausland DM 60,-.

HINWEIS

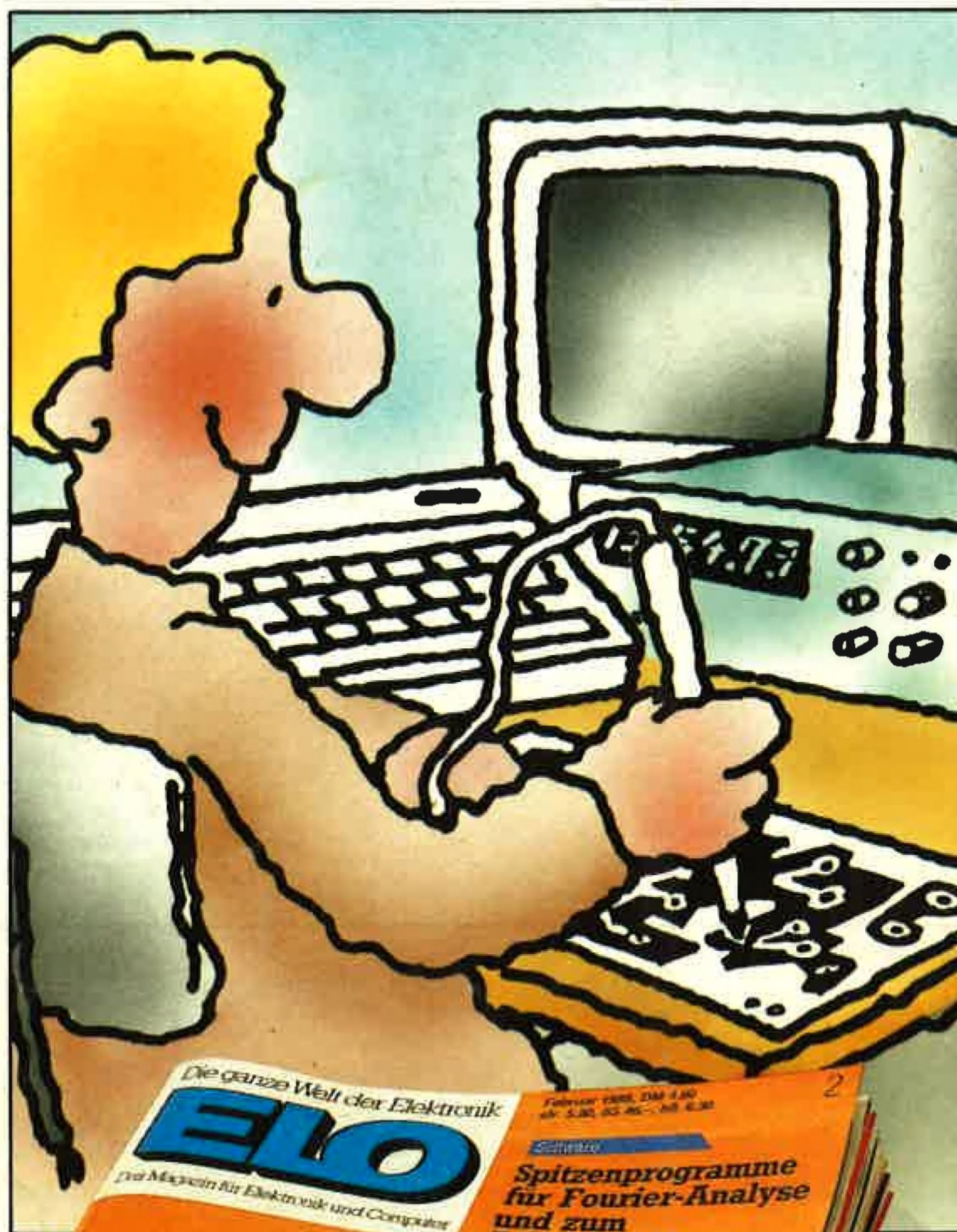
Elektronik verstehen, Computer verstehen – mit der ELO fängt der Spaß an moderner Technik erst so richtig an.

Weil Sie in der ELO leicht zu realisierende **Bauanleitungen** finden, die Ihnen Elektronik praktisch nahe bringen und oft **neue Einsatzbereiche für den Computer** erschließen: Messen, Steuern, Regeln, Erfassen, Auswerten.

Und weil ELO Ihnen helfen wird, Hardware in Ihrem Sinne zu verändern – auch als Anfänger.

Daneben finden Sie leicht verständliche **Beiträge zu Grundlagen der Elektronik**, die sehr schnell dazu beitragen, technische Zusammenhänge zu erkennen und zu nutzen.

Im Magazinteil der ELO und in ELO-Reports wird berichtet, **was mit Elektronik und Computern alles möglich ist**. In der Forschung, in der Industrie, im Verkehrswesen, im Umweltschutz...



Die Rubrik ELO-Service bringt regelmäßig **Marktübersichten**, die Ihnen wichtige Orientierungshilfe im immer unüberschaubarer werdenden Markt elektronischer Produkte sein werden.

Und natürlich finden Sie **ELO-Testberichte**: Über Schach- und Homecomputer, über Video und HiFi-Geräte, über Meßgeräte und alles, was damit im Zusammenhang steht.

Die ELO verbindet die Elektronik mit dem Computer. Wer endlich richtig einsteigen möchte und den Spaß an moderner Technik neu entdecken will, für den gibt's die ELO für DM 5,50 an allen größeren Zeitschriften-Verkaufsstellen.

Noch besser: Sie machen von unserem vorteilhaften Kennenlern-Angebot Gebrauch.

Eine Kennenlern-Karte finden Sie an der Umschlagklappe.



ELO

Das Magazin für Elektronik und Computer