Sonderheft Nr. 9903

DM 14,-

HAPPY-COMPUTER

SCHNEIDER-SONDERHEFT

HAPPY-COMPUTER

DAS GROSSE HEIMCOMPUTER-MAGAZIN

CPC 6128, 664, 464: Alle Schneider-Computer im Vergleich

Grundlagen:

★ Basic für Aufsteiger
★ Grafik-Kurs
★ Soundprogrammierung

CP/M: Was nicht im Handbuch steht

Basteleien:

★ Jetzt paßt jeder Drucker
★ Trakball auch für CPC

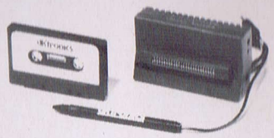
Über 20 neue Listings

Alle Programme auch auf Diskette und Kassette erhältlich

Schneider CPC Hardware & Zubehör

dk'tronics Lightpen für CPC 464

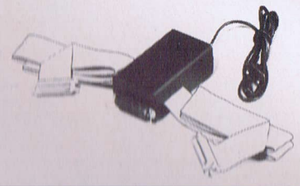
Das Werkzeug für den Grafikprogrammierer: Pixelweise Auflösung, volle Ausnutzung der Farb- und Grafikmöglichkeiten des CPC, auch für Grünmonitor, umfangreiches Hilfsprogramm mit Menuesteuerung über Pictogramme, dadurch kinderleichte Handhabung, kompatibel zu DDI-1 und MP1.
Komplett mit Handbuch und Hardcopyroutine.



Best.-Nr. 6400
nur 99,– DM

NEU **6510 Druckerinterface von Data Media für Schneider CPC 464/664**

2 verschiedene Drucker können wahlweise betrieben werden. Steuerung über die Software, durch Eingabe des jeweiligen Druckercodes. Mit eigener Stromversorgung!



198,– DM

Staubschutzhauben
aus weichem Kunstleder, beigefarben für Computer und Peripherie.

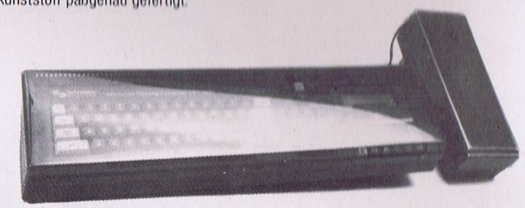
f. Grünmonitor
Best.-Nr. 6409
37,50 DM

f. Farbmonitor
Best.-Nr. 6410
37,50 DM

für Drucker NLQ 401
Best.-Nr. 6408
22,50 DM

f. CPC 464 Best.-Nr. 6406 **22,50 DM**
f. Floppy DDI-1 Best.-Nr. 6407 **19,90 DM**

DATA MEDIA Abdeckhauben
Formschön und stabil! Schützen Sie Ihre wertvolle Hardware vor Staub, Schmutz und Beschädigung. Die Abdeckhauben sind aus durchsichtigem, rauchglasfarbenem Hartkunststoff paßgenau gefertigt.



Best.-Nr. 6110 Abdeckhaube für CPC 464 Computer **35,– DM**
Best.–Nr. 6120 Abdeckhaube für Floppy DDI-1 **29,– DM**

5 1/4"-Laufwerk von Data Media
Das professionelle Speichermedium für Ihren CPC 464!
Einsatzmöglichkeit als eigenständiges Laufwerk unter CP/M. Maximale Speicherkapazität von 3.2 MB bei 4 Laufwerken, eingebauter Controller sowie ein Maschinensprache-Monitor auf Eprom.

Lieferbar ab 10.09.85

Zum Lieferumfang gehören:

– Laufwerke mit Netzteil

– Controller-Karte mit Eprom

– je 1 deutsches Handbuch für CP/M und Bedienungsanleitung des Monitors

Folgende Konfigurationen sind möglich:

Best.-Nr.	Anz. d. Laufwerke	Speicherkapaz.	Preis
6130	2	1.6 MB	ca. 1500,– DM
6140	3	2.4 MB	ca. 2200,– DM
6150	4	3.2 MB	ca. 2900,– DM

Speichererweiterung von Data Media für CPC 464
Lieferbar ab 10.09.85

NEU NEU

Bestell-Nr. 6170	64KB	Speichererweiterung	**198,– DM**
Bestell-Nr. 6171	128KB	Speichererweiterung	**298,– DM**
Bestell-Nr. 6172	256KB	Speichererweiterung	**398,– DM**
Bestell-Nr. 6173	512KB	Speichererweiterung	**498,– DM**

Leerdisketten
3" (10er Pack) **129,00 DM**

Zubehör von John Hall:
Druckerkabel für Schneider-Drucker, 150 cm lang
Best.-Nr. 6402 **DM 79,– DM**

Joystick-Adapter
Zum Anschluß von 2 Standard-Joysticks an Schneider Computer **17,90 DM**
Best.-Nr. 6404

Schneider-Stereo-Kabel
3,5 mm Klinken-Stecker/
2 Chinch Stecker
150 cm lang
Best.-Nr. 6400 **16,50 DM**

Ohne Abbildung:
Schneider-Monitor-Verlängerungskabel (2 Kabel) für CPC Monitor
Best.-Nr. 6403 **19,50 DM**

Ohne Abbildung:
Schneider-Stereo-Kabel
3,5 mm Klinken-Stecker/5 pol. DIN-Stecker, 150 cm lang
Best.-Nr. 6401 **16,50 DM**

The Stick
Völlig neues Steuergefühl durch Einhandbedienung!
Durch die Bewegung der Hand besonders geeignet für schnelle Action.
Mittels Saugfuß auch stellbar.

Fabrikat: John Hall
Best-Nr. 6405 **49,– DM**

Wie bestellen?
Bestellung an DATA MEDIA senden. Lieferung erfolgt umgehend! Zahlung per Verkasse oder Nachnahme zzgl. Porto- und Nachnahmegebühr. (Nachnahme ins Ausland ist nicht möglich.)

data media gmbh MAILORDER

Weitere Artikel in unserem Gesamtkatalog. Bitte anfordern (2,– DM für Rückporto beilegen).
DATA MEDIA GMBH, Bereich Mailorder, Postfach 1263, 4620 Castrop-Rauxel
Telefonische Auskünfte: 0231/125071

Über 120 Seiten Hilfe

Sie halten die erste Schneider-Ausgabe aus der Sonderheft-Reihe von Happy-Computer in den Händen — vollgepackt mit vielen nützlichen Informationen für alle Schneider-Computer-Fans. Die CPCs sind tolle Computer mit fast unbegrenzten Einsatzmöglichkeiten, auf die das Handbuch bei weitem nicht eingeht. Um Ihren Spaß an Ihrem Hobby noch zu vergrößern und Ihren Computer noch mehr auszureizen, haben wir dieses Sonderheft geschrieben.

Übrigens:
Für alle, die keine Zeit (oder Lust) haben, die Listings aus diesem Sonderheft abzutippen, gibt es die Programme auch auf 3-Zoll-Diskette (34,90 Mark, Best.-Nr. LH 85 S2 D), 5¼-Zoll-Diskette (für Vortex-Laufwerk) (34,90 Mark, Best.-Nr. LH 85 S2 V) und Kassette (29,90 Mark, Best.-Nr. LH 85 S2 K). Für Ihre Bestellung benutzen Sie bitte die Zahlkarte in der Mitte des Heftes.

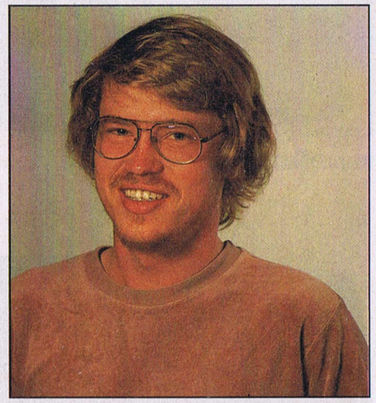
Wie kam es eigentlich zu dem Riesenerfolg von Schneider? Ein Jahr ist es her, da tauchte mit Schneider im Bereich der Heimcomputer ein neuer Name auf. »Schon wieder einer«, war die Reaktion vieler. Inzwischen mußten sich diese Leute eines Besseren belehren lassen. Mehr als 100000 Computer hat Schneider in den deutschsprachigen Ländern Europas verkauft und sich damit hinter Commodore auf Platz zwei der Verkaufs-Hitparade geschoben. Und das zu einer Zeit, wo der Markt lange nicht mehr die Zuwachsraten der Anfangszeit aufweist.

Die Gründe für diese erstaunliche Entwicklung sind vielfältig. Massenproduktion in Fernost (Korea) und billige Bausteine, die ihre Entwicklungskosten schon längst wieder eingespielt haben, machen die Schneider-Geräte so günstig. Dies und jede Menge Kreativität bei der Entwicklung der Firmware haben einen Computer möglich gemacht, der die 8-Bit-Struktur voll ausreizt. Das Betriebssystem und der Basic-Interpreter sind es, die den CPC so interessant machen.

Software-Lösungen, wo früher spezielle Bausteine benötigt wurden, heißt das Geheimnis des Newcomers. So gibt es keinen festen Zeichensatz oder eine immer gleiche Tastaturbelegung. Fast alles kann per Programm umdefiniert werden. Und alles auf einem Niveau, das es auch dem Einsteiger erlaubt, sich seinen eigenen »individuellen« Computer zu »basteln«.

Mit einer Bildschirmauflösung von 640 x 200 Punkten, 27 verschiedenen Farben, 80 Zeichen pro Zeile, Windows, einem Synthesizer mit drei Tonkanälen und einem extrem schnellen Basic, brauchen sich die Computer der CPC-Reihe vor niemandem zu verstecken. Das moderne Konzept der Schneider-Computer nutzt die 8-Bit-CPU Z80A voll aus.

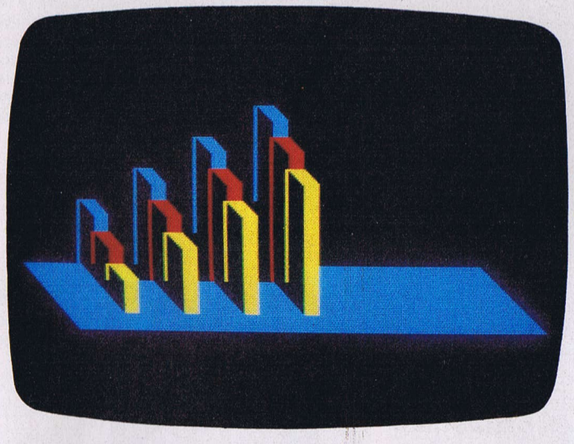
Das haben inzwischen auch die Peripherie-Hersteller erkannt. So gibt es inzwischen verschiedene Speichererweiterungen (bis auf 512 KByte), 5¼-Zoll-Diskettenlaufwerke mit eigenem Controller (wie die 3-Zoll-Station von Schneider werden sie alle mit dem Betriebssystem CP/M geliefert), Lichtgriffel und verschiedene Interfaces. Sogar eine MS-DOS-Karte hat das Entwicklungslabor bereits verlassen und soll die Schneider-Computer IBM-kompatibel machen.



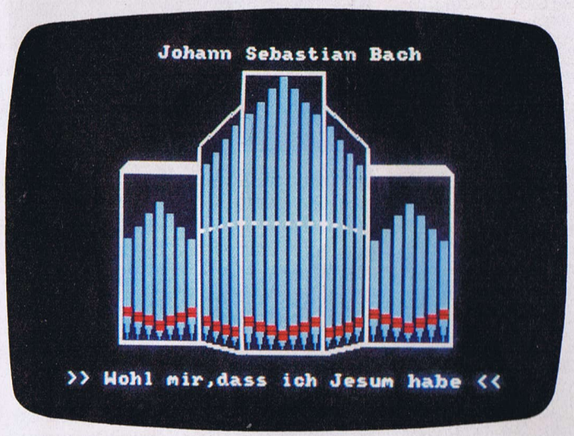
Bei allen Diskettenstationen bekommt man das Betriebssystem CP/M mitgeliefert. Dem Besitzer des Schneider-Computers eröffnet sich dadurch eine fast unüberschaubare Vielfalt von (Anwendungs-) Software, die für zu Hause und kleinere Betriebe vollkommen ausreichend ist. In diesen Tagen haben auch die Softwarehersteller die Bedeutung dieses Marktes erkannt. Als erstes werden »Wordstar«, »dBase« und »Multiplan« für weniger als 200 Mark angeboten. Ein Preis, der oft auch für »ganz normale« Heimcomputerprogramme verlangt wird.

Bei dieser Gelegenheit möchte ich all denjenigen Lesern danken, die uns ihr Wissen und ihre Programme zur Verfügung gestellt haben. Denn keiner kann alles wissen, und so wie »Happy-Computer« soll auch dieses Sonderheft ein Forum für alle »Schneider-Fans« sein. Weiterhin sei allen gedankt, die bei der Zusammenstellung und Produktion dieses Sonderheftes ihren Beitrag geleistet haben.

Andreas Hagedorn



Kurs: Grafik ohne Geheimnisse 17



Unser Beitrag zum Bach-Jahr: Orgelklänge zum Abtippen 62

Und es geht noch: Trakball an CPC angepaßt 134

Warum Schneider?

1000 Argumente für den Schneider 6

Hardware

CPC 6128, 664, 464: Alle Schneider-Computer im Vergleich
Ein Schneider für jeden Zweck 8

Grafik

Grundlagen: Grafik-Kurs
Der Schneider im Reich der Farben 17
Grafik mit Spezialeffekten 23
»Geheimcodes« geknackt 27
Grafik aus dem ROM gelockt 34

Grafik-Listings

Malen wie auf einer Leinwand (Malprogramm) 36
Bildschirmspielereien (Demonstration) 42

Sound

Grundlagen: Soundprogrammierung
Musik und Sound selbst programmiert 43

Sound-Listings

Orgelspielen auf dem Schneider (Synthesizer) 58
Ihr Schneider spielt Bach (Demonstration) 62

Basic

Grundlagen: Basic für Aufsteiger
Basic Special: Besonderheiten und Neues 64
Simultan auf zwei Hochzeiten: Echtzeitverarbeitung auf dem Schneider 67
Dem Basic aufs Byte geschaut 70
Ein-Blick in den Variablenspeicher 74
So funktioniert die Stringverarbeitung 78

Spiele-Listings

Trainieren Sie Ihren Schneider! (Fußballmanager) 83
Retten Sie Ihren Spion! (Geschicklichkeitsspiel) 92
Frisch aus Las Vegas — Olli's Crown (Spielautomat) 98
Wyatt Earp jagt Billy the Kid (Strategiespiel) 100
Kopfüber in die Aktien (Wirtschaftssimulation) 104

Anwendungs-Listings

Daten gut aufbereitet (Dateiverwaltung) 109
Funktionen anschaulich gemacht (Mathematik) 111
Assembler-Disassembler für den CPC 464 (Entwicklungssystem) 112

Tips & Tricks

Nützliche Tips zur Eingabe der Listings (Eingabehilfen) 16
Eine deutsche Schreibmaschinentastatur für Ihren Schneider (Tastaturbelegung) 119
Der neue Befehl: Mode 1, 2, 3 (Grafik-Utility) 120
Verbessern Sie die LIST-Routine (Utility) 122
Wem der Druckerpuffer zu klein ist ... (Utility) 122
Menügesteuerte Diskettenstation (Bedienerhilfe) 124

CP/M

CP/M: Was nicht im Handbuch steht
CP/M — Das zweite Gesicht Ihres Schneiders 127
Wordstar — A Star was born 130

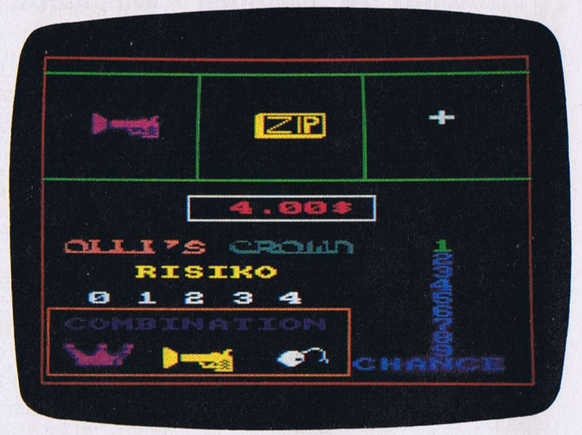
Bastelei

Krach in Stereo: Ein Soundkabel für den Schneider 56
Jetzt paßt jeder Drucker
Centronics-Schnittstelle mit 8 Bit 132
Trakball auch für CPC
Ein Trakball für den Schneider 134

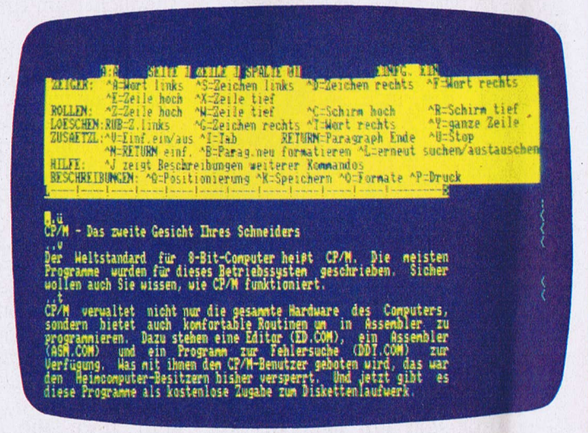
Rubriken

Editorial 3
Inserentenverzeichnis 138
Impressum 138

CPC 6128 — Lohnende Alternative zu 464 und 664? 8



Ihr Schneider wird zum Groschengrab 98



Wordstar: Profiprogramm in der Praxis 130

1000 Argumente für den Schneider

Die einen schwören auf den Commodore 64, die anderen auf MSX, Sinclair oder Atari. So mancher wartet auf den Amiga, viele hätten gern einen Apple oder einen IBM-PC. Viele landen allerdings beim Namen »Schneider«.

Wenn ein absoluter »Newcomer« auf dem Computer-Sektor innerhalb von 12 Monaten mehr Computer verkauft, als andere bekannte Hersteller in fünf Jahren, dann verdient das sicher einige Aufmerksamkeit. Was ist das Besondere an dem Computer, der innerhalb eines Jahres Sinclair und Atari zu Statisten degradierte und der nun dem bisher unangefochtenen Marktführer Commodore das Fürchten lehrt? Was können die mittlerweile drei Schneider-Computer, das andere Heimcomputer nicht können? Die letzte Frage ist schnell beantwortet: eine ganze Menge. Der Schneider basiert auf dem Mikroprozessor Z80A, einem der leistungsfähigsten 8-Bit-Prozessoren überhaupt. Obwohl der Z80A wie alle 8-Bit-Prozessoren nur 64 KByte Speicher adressieren kann, verfügt der Schneider bereits in der Grundversion über 96 KByte Speicher, die sich in 64 KByte (hauptsächlich Anwender- und Bildschirmspeicher) und 32 KByte ROM (Betriebssystem und Basic) aufteilen. Beim CPC 6128 kommen nochmals 64 KByte RAM hinzu. Um damit arbeiten zu können, ist ein sehr modernes Betriebssystem installiert worden, das den Speicher mittels »Bank-Switching« verwaltet. Das bedeutet nichts anderes, als daß die einzelnen Speicherteile verschiedenen Speicherbänken zugeordnet werden, von denen jede maximal 64 KByte groß sein darf. Das Betriebssystem schaltet nun je nach Bedarf die eine oder andere Bank ein, um auf Programme oder Daten zugreifen zu können. Das Basic-ROM (16 KByte) liegt beispielsweise an der gleichen Adresse wie der Grafikbildschirm, jedoch in einer anderen Speicherbank. Anstelle der 16 KByte Basic können auch externe ROM-Module verwendet werden, die dann weiteren Bänken zugeordnet sind. Bis zu 252 ROMs oder RAM-Module können im oberen 16-KByte-Adreßbereich »parallel« geschaltet werden. Das entspricht einem Gesamtspeicher von fast 4 Megabyte ROM oder RAM, dessen Verwaltung das Schneider-Betriebssystem übernimmt.

Doch nicht nur bei der Speicherverwaltung ging man beim Schneider neue Wege. Eine ausgeklügelte Mischung aus Hardware-Interrupt und Software-Statusabfragen ermöglicht zum erstenmal bei einem Heimcomputer die Programmierung echter »High-Level«-Interrupt-Routinen und damit grundsätzliche Multi-Tasking-Fähigkeit, also das gleichzeitige Bearbeiten mehrerer Programme. Im Betriebssystem-Kern sind für diesen Zweck eine ganze Reihe von Service-Routinen vorhanden. Der Dreh- und Angelpunkt bei der Interrupt-Programmierung ist der Hardware-Interrupt, der 300mal in der Sekunde ausgelöst wird. Während eines solchen Interrupts unterbricht der Z80-Prozessor sein gerade laufendes Programm, rettet alle zur späteren Weiterverarbeitung wichtigen Daten und fragt die sogenannte »Fast Ticker«-Warteschlange ab. In dieser Warteschlange befinden sich, vereinfacht gesagt, die Startadressen von Maschinenprogrammen, die zu bestimmten Zeitpunkten aufgerufen werden müssen. Der Benutzer kann eigene Programme in diese Warteschlange einreihen und die Zeit zwischen zwei aufeinander folgenden Aufrufen festlegen. Anschließend braucht man sich um nichts mehr zu kümmern.

Das Basic unterstützt diese Interruptverarbeitung mit den Befehlen AFTER und EVERY — eine absolute Neuheit in der Geschichte der Mikrocomputer. Doch der Schneider bietet nicht nur technische Spitzenleistung zu einem fast unglaublich niedrigen Preis, er ist auch ein sehr vielseitiger Computer zum ständigen Gebrauch. Technologie ist hier kein Selbstzweck, sondern dient ganz effektiv dem Anwender.

Im Gegensatz zu anderen Computern sind die Schneider-Maschinen nicht ausschließlich auf einen Anwendungsbereich eingeschränkt. Ob man gerne selbst programmiert oder lieber fertige Programme anwendet, ob man sich für Computerspiele begeistert oder professionelle Software für den Betrieb braucht — in allen Fällen ist man mit einem Schneider-Computer gut beraten.

Mit einem dreistimmigen Synthesizer, einem frei programmierbaren Zeichensatz, 27 Farben und einer Grafikauflösung von bis zu 200 x 640 Punkten, ist der Schneider ein Heimcomputer einer neuen Leistungsklasse. Auch die Spiele-Hersteller haben das erkannt, und inzwischen erscheinen die meisten neuen Spiele auch sofort in einer Schneider-(Amstrad-)Version. So ist es nur noch eine Frage der Zeit, bis für die Schneider-Computer ein vergleichbares Spiele-Angebot zur Verfügung steht wie für den Commodore 64.

Dazu kommt die Möglichkeit, sich seine eigenen Anwendungen, Spiele, Hobby-Programme etc. mit wenig Aufwand selbst zu programmieren. Der Schneider verfügt über insgesamt 32 KByte ROM für Betriebssystem und Basic, der C 64 beispielsweise muß mit der Hälfte auskommen — und das merkt man: Basic-Programme, die für den C 64 geschrieben wurden, bestehen vielfach im wesentlichen aus einer Aneinanderreihung undurchsichtiger PEEK-, POKE- und SYS-Befehle, die von endlosen DATA-Zeilen mit Zahlenfolgen eingerahmt sind.

Problemlos programmieren

Wo beim Schneider bequeme Basic-Befehle zur Verfügung stehen, muß man sich beim Commodore 64 mit direkten Zugriffen auf die Hardware behelfen. Statt mit »border 0« wird der Bildschirmrahmen beim Commodore beispielsweise mit »POKE 53280,0« schwarz eingefärbt. Um eine Linie in hochauflösender Grafik zu zeichnen, braucht der Schneider einen einzigen Basic-Befehl — beim Commodore hat man die Wahl zwischen einer POKE-PEEK-Orgie mit Ausführungszeiten zwischen zehn und zwanzig Sekunden und der Alternative, gleich alles in Maschinensprache zu programmieren.

Auch wenn eingefleischte C 64-Freaks gerade diesen Gesichtspunkt als besonders positiv herausstreichen (»Man kommt sofort in Kontakt mit Maschinensprache«) — einen Basic-Befehl tippt man immer noch in wenigen Sekunden ein, für die Entwicklung eines Maschinenprogramms braucht man halt doch ein wenig länger. Außerdem: Wer gerne in Assembler programmiert, der kann das beim Schneider auch gerne tun, wenn er will. Beim C 64 muß er es tun, ob er will oder nicht.

Wer etwas Geld für Software übrig hat, der kann für seinen C 64 diverse Basic-Compiler erhalten. Diese Compiler übersetzen ein Basis-Programm in Maschinensprache und machen es dadurch um einiges schneller — typischerweise etwa zwischen fünf- und zehnmal. Für Schneider-Besitzer ist das aber kein Grund zum Neid: Der Schneider arbeitet Basic-Programme etwa sechs- bis zwölfmal so schnell ab wie der C 64. Das bedeutet im Klartext: Das Schneider-Basic ist beim »normalen« Interpretieren bis auf wenige Spezialfälle schneller als compilierte Programme auf dem C 64.

Wer mehr zu professionellen Anwendungen tendiert, der bekommt für 1598 Mark (CPC 6128) eine schnelle CP/M-Maschine mit einer Speicherkapazität, die weit über das hinausgeht, was vergleichbare Computer zu bieten haben. Aber auch mit den »kleineren« Schneider-Computern (CPC 464/646) braucht man hinsichtlich Professionalität nicht hintenan zu stehen. Alle drei Computer sind CP/M-fähig, und das einzige Hindernis bei der sofortigen Nutzung des umfangreichen Angebotes an CP/M-Software besteht im marktunüblichen 3-Zoll-Diskettenformat. Allerdings sollte man das nicht zu tragisch nehmen: Inzwischen sind bereits die ersten CP/M-Programme in einer speziell angepaßten Schneider-Version auf 3-Zoll-Disketten erhältlich.

Es handelt sich dabei um das Textverarbeitungsprogramm »Wordstar«, das Datenbanksystem »dBase II« und die Tabellenkalkulation »Multiplan«. Es sind dies drei professionelle Programme der Spitzenklasse, die bereits vor einigen Jahren entwickelt wurden, als CP/M-Computer noch 5000 Mark kosteten und fast ausschließlich im kommerziellen Bereich angewendet wurden. Wordstar, dBase II und Multiplan laufen jetzt in der MS-DOS-Version auf dem IBM-PC und dazu kompatiblen Computern und werden von Großfirmen für die täglich anfallenden Arbeiten eingesetzt. In der Leistungsfähigkeit stehen die Schneider-Versionen dieser Programme den IBM-Versionen praktisch in nichts nach.

Und das ist erst der Anfang. Sowohl für den neuen Commodore 128 als auch für die Schneider-Computer wird immer mehr CP/M-Software preiswert zu erhalten sein. Fortran, Cobol, C, Pascal oder Lisp — praktisch alle Programmiersprachen sind unter CP/M erhältlich.

Auch hinsichtlich des Speicherbereiches schneidet der Schneider nicht schlecht ab: Wem die 64 KByte RAM und die 32 KByte ROM nicht reichen, der kann seinen Schneider auf bis zu 512 KByte RAM per Vortex-Karte aufrüsten — der ROM-Bereich ist sowieso auf etliche Megabyte ausbaubar. Und wer gerne mit 16-Bit-Software unter MS-DOS arbeiten würde, aber die Ausgabe von über 5000 Mark für einen IBM-PC ohne Farbe, ohne Grafik und ohne Sound scheut, für den gibt es für zirka 500 Mark die MS-DOS-Karte zum Schneider. Durch 8088-Prozessor, 128 KByte zusätzliches RAM und MS-DOS-Betriebssystem kann man mit dem Schneider auf das gesamte Angebot an MS-DOS-Software zurückgreifen — aber Vorsicht beim Einkauf: Nicht selten ist derartige Software teurer als der ganze Schneider-Computer.

Aber egal ob CP/M oder MS-DOS, der Schneider bietet in jedem Fall wirkliche Personal Computer-Leistung zum Heimcomputer-Preis.

Keine Angst vor großen Tieren

In den letzten Wochen und Monaten sind neue 16-Bit-Computer auf der Basis des 68000-Prozessors ins Gespräch gekommen. Der Atari 520 ST und der Commodore Amiga werden allerorten in den Computerhimmel gelobt. Da wird die fantastische Grafikauflösung genannt, die Unmenge darstellbarer Farben, Bildschirmfenster (Windows), die Fähigkeit, mehrere Programme quasi gleichzeitig ablaufen zu lassen (Multi-Tasking) und die hohe Arbeitsgeschwindigkeit.

Das alles ist ja auch ganz schön und richtig, nur: Die Grafikauflösung beträgt sowohl beim Atari als auch beim Amiga gerade das Doppelte wie beim Schneider.

Und was die anderen Fähigkeiten angeht: Der Schneider kann acht Bildschirmfenster gleichzeitig ansprechen. Multi-Tasking ist selbst von Basic aus durch vier Interruptebenen möglich, und wer für seine Programme mehr als 27 Farben benötigt, der sollte statt mit dem Computer doch lieber gleich analog malen: mit Pinsel und Palette und auf richtiger Leinwand.

Bleibt noch die Geschwindigkeit. Das ist natürlich ein gewichtiges Argument, denn wer hat es schon gerne, wenn er immer auf seinen Computer warten muß? Doch was heißt hier Geschwindigkeit. Wie lange müssen Sie im Durchschnitt nach Betätigung der Enter-Taste warten, bis der Schneider Ihr Kommando ausgeführt hat und »Ready« meldet? Zugegeben, die neuen 16-Bit-er Commodore Amiga und Atari 520 sind ganz ausgezeichnete Computer und für ihre Leistung wirklich nicht zu teuer. Aber es gibt zumindest einen 8-Bit-Computer, der ihnen auf vielen Gebieten kaum etwas nachsteht — der Schneider.

(Volker Everts)

Ein Schneider für jeden Zweck

Drei Computer in sechs verschiedenen Varianten gibt es inzwischen aus Türkheim. Wer sich einen Schneider kaufen will, der hat die Qual der Wahl. Vor dieser Entscheidung sollte man deshalb die Geräte kennen und sich darüber klar sein, für welche Zwecke man den Computer einsetzen will.

Die drei Computer von Schneider sind: der CPC 464, der CPC 664 und der CPC 6128. Alle werden mit Farb- oder Grünmonitor und Massenspeicher geliefert. Der CPC 464 hat einen Kassettenrecorder eingebaut, die beiden anderen eine 3-Zoll-Diskettenstation. Da das Betriebssystem und der Basic-Interpreter sich sehr stark ähneln, sind die Computer annähernd kompatibel. Vollständige Kompatibilität haben auch die Entwickler von Amstrad nicht geschafft, aber der Schritt weg von dem Spruch »Kompatibel nur zu sich selbst« ist gelungen.

Bei dieser Produktreihe werden Erinnerungen an die Automobilindustrie wach, bei der in ein Modell auch verschiedenste Ausstattungsvarianten eingebaut werden können. Doch welche Version ist nun für wen die richtige? Um diese Frage zu beantworten, wollen wir Ihnen zuerst einmal die einzelnen Geräte vorstellen.

CPC heißt »Colour Personal Computer«, die erste Ziffer der Zahl steht für den Massenspeicher (»4« für den Kassettenrecorder und »6« für Diskettenstation) und die restlichen für den Speicherbereich. Somit haben der 464 und der 664 jeweils 64 KByte und der 6128 128 KByte RAM. In Deutschland werden die Geräte nur zusammen mit dem Monitor verkauft, in dem auch das Netzteil mit eingebaut ist, das die 5 Volt Spannung für den Computer und die 12 Volt für die Diskettenstation liefert. Das lästige Kabelwirrwarr vieler anderer Heimcomputer entfällt und ein Kabel zur Steckdose reicht somit für das Grundgerät aus.

Im CPC 464 ist — wie schon erwähnt —, als Massenspeicher ein einfacher Kassettenrecorder eingebaut. Einfach deshalb, weil es sich um einen ganz normalen Recorder handelt ohne komplizierte Elektronik, die für einen Datenrecorder auch absolut unbrauchbar wäre. Die Start-/Stop-Funktion wird vom Computer kontrolliert, so daß nach dem Laden eines Programms oder einer Datei das Band automatisch stehen bleibt. Beim Speichern mehrerer Dateien auf Kassette, die man später an verschiedenen Stellen des Programms laden will, ist dies sehr nützlich, da eine manuelle Betätigung fast nie rechtzeitig erfolgt. Die beiden anderen Computer, der 664 und 6128, haben ein Diskettenlaufwerk eingebaut. Oft kritisiert wurde dabei das unübliche 3-Zoll-Format; man lernt es aber sehr schnell zu schätzen. Die Übertragungsgeschwindigkeit von 250 KBit pro Sekunde ist sehr hoch. Mit ihr sinkt die mittlere Zugriffszeit auf 200 Millisekunden und hat damit einen für Heimcomputer fast traumhaft zu nennenden Wert.

Die gleiche Diskettenstation wird als Peripheriegerät auch für den CPC 464 angeboten. Der Controller (er kann zwei Laufwerke gleichzeitig steuern) ist dann einfach auf den Systembus, der hinten aus dem Computer herausgeführt ist, aufzustecken. Mit der Station, beziehungsweise dem Computer, wird auf Diskette das Betriebssystem CP/M und die Programmiersprache Logo geliefert. CP/M heißt »Control Program for Microcomputers« und ist ein Disketten-Betriebssystem, das sich als Industriestandard für 8-Bit-Computer durchgesetzt hat. Entwickelt wurde es Ende der siebziger Jahre von Digital Research. Obwohl seine Bedeutung mit Einführung der 16-Bit-Geräte stark abnahm, gibt es für CP/M noch heute die größte Auswahl an Software.

Während CP/M-Computer vor kurzem noch mehrere tausend Mark kosteten, ist in diesem Bereich mit dem Schneider ein enormer Preisrutsch eingetreten. So ziehen jetzt auch die Softwarehäuser nach und bieten ihre CP/M-Programme zu Preisen an, die vor wenigen Monaten noch undenkbar schienen. Ob dies die Erkenntnis ist, daß man mit Software, die sich im professionellen Bereich kaum noch verkaufen läßt, so noch einmal etwas im Heimbereich verdienen kann, oder ob man sich sagt, wer zu Hause mit »Wordstar« arbeitet, der benutzt dies auch im Büro in der teuren Ausführung auf seinem 16-Bit-Gerät, ist nicht bekannt. Für uns, die Benutzer, bleibt nur die Erkenntnis, daß Programme, die noch heute in vielen Firmen genutzt werden, jetzt auch zu Hause interessant sind.

Nach so viel Schwärmerei kommt jetzt aber doch ein Dämpfer. Die beiden kleinen Computer von Schneider stellen nämlich für viele CP/M-Programme zu wenig Speicherplatz zur Verfügung. Oft braucht man bis zu 48 KByte, der Schneider hat jedoch nur gut 40 KByte unter CP/M frei. Das liegt daran, daß alle drei Geräte einen 16 KByte großen Bildschirmspeicher benutzen — ein normaler CP/M-Computer hingegen einen bedeutend kleineren. Die vielen Farben kosten eben doch Platz. Der CPC 6128 kennt das Problem nicht. Da er unter CP/M auf die zweiten 64 KByte RAM zurückgreift, hat er über 60 KByte für Programme frei. Aber auch den Besitzern der kleinen Computer wird geholfen. Zwar bietet Schneider selbst noch keine Speichererweiterung an, wohl aber die Firma Vortex. Auf einer Platine, die direkt im Gerät eingesteckt wird, können Speicherbausteine mit einer Kapazität von bis zu 512 KByte hinzugefügt werden. Das reicht dann für jeden Anwendungszweck.

Kehren wir zurück zu den eigentlichen Computern. Der normale RAM-Speicher beträgt also 64 KByte. Auch bei dem 6128, da von Basic aus die zweiten 64 KByte — etwas umständlich — nur mit einem Hilfsprogramm benutzt werden können. Dieses findet man auf einer der beiden Disketten, die man beim Kauf des Geräts miterhält.

Von den 64 KByte hält der Schneider gut 40 KByte für Ihre Programme bereit. Ohne Diskettenstation sind es ein paar Byte mehr, da das DOS (Disc Operating System = Diskettenbetriebssystem) Platz für einige Variable am oberen Ende des Speichers benötigt. Bildschirmspeicher und Systemmeldungen belegen den restlichen RAM-Bereich. Das ROM mit Basic und dem Betriebssystem liegt parallel dazu zwischen den Adressen 0000 hex und 3FFF hex, sowie C000 hex und FFFF hex. Mittels Bankwitching schaltet der Z80 zwischen den Bereichen hin und her.

Zur Bildschirmausgabe gibt es drei verschiedene Betriebsarten. 80, 40 oder 20 Zeichen pro Zeile werden in zwei, vier oder 16 verschiedenen Farben ausgegeben. Die Zeilenzahl ist dabei auf 25 pro Bildschirmseite festgelegt. Für hochauflösende Grafik (übrigens gleichzeitig mit Texten darstellbar) kann man 640, 320 oder 160 mal 200 Punkte ansprechen. Für die Tonausgabe ist ein AY-3-8912 eingebaut, der mit drei verschiedenen

Der Kleine: CPC 464 mit eingebautem Kassettenrecorder

Kanälen arbeitet: dem rechten, dem linken und einem für beide Seiten gemeinsam. Der Stereoeffekt, den diese drei Kanäle bewirken, kommt auf dem eingebauten Lautsprecher leider nicht sonderlich zur Geltung. Dessen Qualität läßt nämlich sehr zu wünschen übrig. Benutzt man deswegen ein Kabel für die heimische Stereoanlage, dann klingt die Computermusik ausgesprochen gut.

Bei der Hardware der Schneider-Computer fällt noch die Tastatur auf, die von Gerät zu Gerät professioneller wurde. Allerdings ist bei 6128 das übersichtliche dreigeteilte Tastenfeld geopfert worden. Ähnlich vieler anderer Personal Computer sind alle Tasten zu einem einzigen großen Feld zusammengefaßt worden; ein Griff und die falsche Taste ist betätigt! Der Tastenanschlag hingegen ist beim 6128 wirklich gut.

Was die Computer aus Türkheim von den meisten anderen unterscheidet, ist das gute Basic und die übersichtliche Konzeption des Betriebssystems. Bei vielen anderen Computern greift der Basic-Interpreter direkt auf die Hardware zu, um seine Anweisungen zu geben, anders beim Schneider. In einer Firmware-Sprungtabelle findet man alle Routinen, die die Hardware ausführen kann. Sie sind dort übersichtlich mit Ein- und Aussprungbedingungen, sowie der Registerübergabe aufgeführt. Dieses System hat mehrere Vorteile. Zum einen können Maschinencode-Routinen des Betriebssystems von jedem Programmierer relativ einfach in seine Programme integriert werden, zum anderen kann man Programme, die nur über diese Sprungtabelle auf die Hardware zugreifen, leicht an eventuelle neue Entwicklungen anpassen. Wenn man nämlich die Sprungtabelle für das neue Gerät umschreibt, läuft jedes Programm, sofern es sauber programmiert wurde (das heißt ohne Direktzugriff auf die Hardware).

Inzwischen gibt es zwei verschiedene Basic-Interpreter, die aber zueinander aufwärtskompatibel sind. Der Schneider CPC 464 arbeitet mit Basic 1.0 von Locomotiv, der 664 und der 6128 mit der Version 1.1. Die beiden unterscheiden sich lediglich durch 15 zusätzliche Befehle in der Version 1.1 und dadurch, daß bei dieser einige kleine Fehler ausgebessert wurden.

Die interessantesten Basic-Befehle findet man mit EVERY und AFTER bei der Interruptsteuerung. Sie bewirken, daß man nach einer bestimmten Zeit ein Programm verlassen und eine Unterroutine bearbeiten kann, bevor im Hauptprogramm fortgefahren wird. Für effektvolle Programme finden sich da sehr viele Anwendungsgebiete: von der Uhr bis hin zur Tonausgabe.

Der Kompakte: CPC 664 mit 3-Zoll-Diskettenstation

Der Speicherriese: CPC 6128 mit 128 KByte RAM

Letztere wird von Basic aus mit den Befehlen SOUND, ENT und ENV gesteuert. ENV und ENT kreieren dabei verschiedene Hüllkurven, die unter dem Befehl SOUND zur Ausführung kommen. Die einzelnen Kanäle werden dabei mit der sogenannten Rendezvous-Technik synchronisiert.

Die Bildschirmgestaltung ist durch sehr viele Befehle steuerbar. Einer der wichtigsten dabei ist MODE, mit dem die Auflösung festgelegt wird. Aus den 27 Farben kann man sich bis zu 16 aussuchen, die dann einzelnen Registern zugeordnet werden. Bei der Wahl von Zeichen- oder Hintergrundfarbe wird dann nicht direkt die Farbe, sondern das Register aufgerufen.

Warum in die Ferne schweifen, wenn es gute deutsche Software zum CPC gibt!

Deutsche CPC Software aus bester Hand

DATAMAT

Deutschlands meistgekaufte Dateiverwaltung

bietet einiges, was in dieser Preisklasse bisher unvorstellbar schien:

- menuegesteuertes Diskettenprogramm, dadurch extrem einfach zu bedienen
- für jede Art von Daten
- völlig frei gestaltbare Eingabemaske
- 80 Zeichen pro Zeile
- Hardcopy
- 50 Felder pro Datensatz
- 512 Zeichen pro Datensatz
- bis zu 4000 Datensätze pro Datei je nach Umfang
- 27 Farben für Rand, Hintergrund und Buchstaben
- Schnittstelle zu TEXTOMAT
- Benutzung von Rechenfeldern
- Anzeige des Disketteninhaltes
- läuft mit ein oder zwei Floppys
- komplett in Maschinensprache, dadurch extrem schnell
- deutscher/amerikanischer Zeichensatz
- fast jeder Drucker ist anschließbar
- duplizieren der Datendiskette
- gute Benutzerführung
- Hauptprogramm komplett im Speicher – kein lästiges Nachladen
- deutsches Handbuch mit Übungslexikon

Sie können:

- jeden Datensatz in wenigen Sekunden suchen
- nach beliebigen Feldern selektieren
- nach allen Feldern, auf- oder absteigend sortieren
- Listen in völlig freiem Format drucken
- Etiketten drucken
- Komplett nur DM 148,– Für CPC 464, 664 und (ab September) 6128 Die richtige Version wird automatisch geladen

TEXTOMAT

Deutschlands meistgekaufte Textverarbeitung

bietet Profileistung zum Hobbypreis! TEXTOMAT in Stichworten:

- Diskettenprogramm durchgehend menuegesteuert
- deutscher/amerikanischer Zeichensatz
- Rechenfunktionen für alle Grundrechenarten
- über 17000 Zeichen pro Text im Speicher
- beliebig lange Texte durch Verknüpfung
- 80 Zeichen pro Zeile
- läuft mit ein oder zwei Floppys
- 27 Farben für Rahmen-Hintergrund-Bildschirmfarbe
- es können Trennvorschläge gemacht werden
- Wordwrap
- Tabulatoren
- Seitennumerierung
- Proportionalschrift auf entsprechendem Drucker
- Zuweisungstabelle für ASCII-Code
- frei definierbare Steuerzeichen, z. B. für Indices, Schriftarten, Unterstreichen, Formate
- umfangreiche Formularanpassungen
- Blockoperationen, 'Suchen und Ersetzen'
- Serienbrieferstellung mit DATAMAT
- formatierte Ausgabe auf dem Bildschirm
- Anpassung an fast jeden Drucker
- ausführliches Handbuch mit Übungslektion
- Komplett nur DM 148,– Für CPC 464, 664 und (ab September) 6128 Die richtige Version wird automatisch geladen

TEXTOMAT PLUS

neues Textverarbeitungsprogramm der Superlative

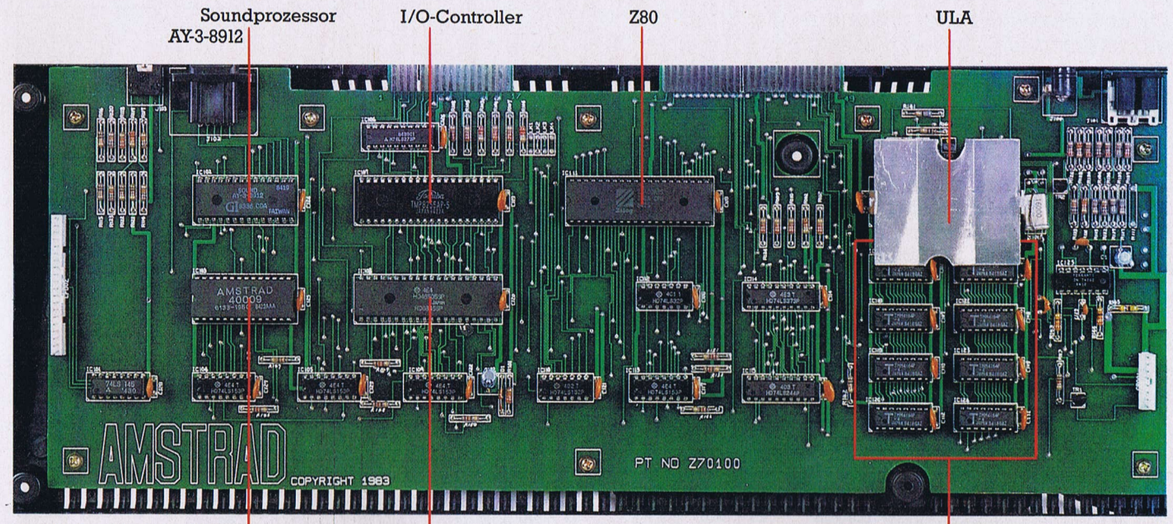
Erheblich erweiterte, leistungsstärkere TEXTOMAT-Version. Bietet alle Möglichkeiten von TEXTOMAT und **zusätzlich:**

+ ergonomische, schreibmaschinenähnliche Texteingabe arbeitet grundsätzlich im 80 Zeichenmodus
+ 2 dynamisch verwaltete Textbereiche im Speicher. Zwischen beiden Texten kann beliebig hin- und hergeschaltet sowie kopiert werden. Wahlweise Menuesteuerung oder schnelle Direktanwahl der Funktionen. 10 Floskeltasten für häufig wiederkehrende Worte oder Redewendungen. Sehr komfortable Cursorsteuerung (vor/zurück – Zeichen/Wort/ Satz/Absatz)
+ Trennvorschläge nach deutscher Grammatik
+ Kopf- und Fußzeilen während des Textes änderbar
+ bedingter Seitenwechsel
+ BASIC Programme können eingelesen, editiert und abgespeichert werden, dabei automatisch ASCII Um- und Rückwandlung
+ Suchen und Ersetzen mit vielen Optionen und Joker (vor/rückwärts – Klein/Großschreibung – ganze Wörter)
+ komplettes Terminalprogramm zum problemlosen Senden und Empfangen von Texten sowohl zum Halb- als auch Vollduplexbetrieb

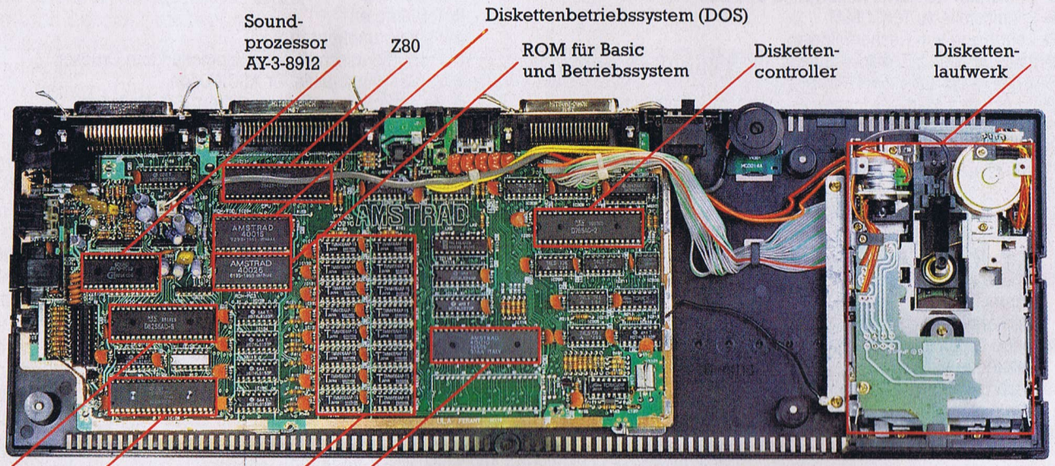
TEXTOMAT PLUS ist ab ca. Ende September für CPC 6128 lieferbar und kostet DM 198,–

DATA BECKER
Merowingerstr. 30 · 4000 Düsseldorf · Tel. (0211) 310010

BESTELL-COUPON
Einsenden an: DATA BECKER · Merowingerstr. 30 · 4000 Düsseldorf 1
Bitte senden Sie mir.
zzgl. DM 5,– Versandkosten
☐ per Nachnahme ☐ Verrechnungsscheck liegt bei
Name und Adresse bitte deutlich schreiben



Die Platine des CPC 464



Die Platine des CPC 6128

Diese Angaben reichen dem Videochip aus, um den Bildschirm farbig zu gestalten. Definiert man nun ein Farbregister um, so ändert sich schlagartig auch die entsprechende Farbe auf dem Bildschirm und das ohne einen einzigen POKE-Befehl.

Der Zeichensatz umfaßt mehr als 200 Symbole, die in einer 8x8 großen Matrix abgebildet werden. Verschiedene Basic-Befehle erlauben aber eine einfache Definition neuer Zeichen, mit denen dann auch die Tastatur belegt werden kann. Im ROM sind der ASCII-Zeichensatz, sowie 128 Grafiksonderzeichen fest einprogrammiert.

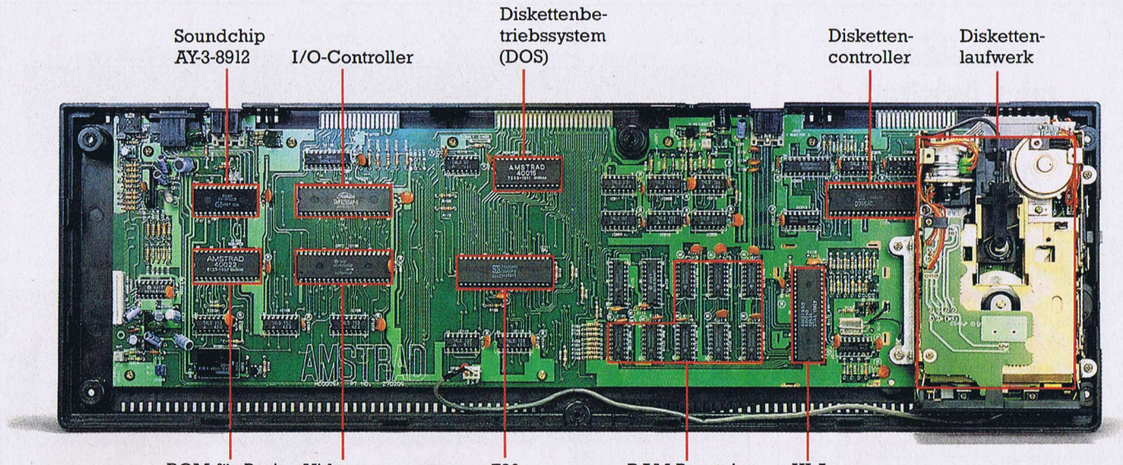
Bemerkenswert ist die Fähigkeit der Schneider-Computer, acht verschiedene Windows (sieben für Text und eins für Grafik) gleichzeitig darzustellen. Die Ausgabe auf ein bestimmtes Fenster steuert eine sogenannte Kanalnummer. Dem gleichen Prinzip folgt übrigens der Drucker. Er besitzt die Kanalnummer 8, während die Bildschirmbereiche mit den Kanälen 0 bis 7 anzusprechen sind.

Wie Sie sehen ähneln sich die drei Computer sehr und auch vom Preis her sind die Unterschiede sehr gering. Preiswert, aber doch mit allem ausgestattet was man braucht, kommt sicherlich der CPC 464 den Bedürfnissen eines Einsteigers am nächsten. Da er ohne weitere Probleme mit Diskettenlaufwerk und anderer Peripherie auszubauen ist, stellt die anfängliche Einschränkung kein Hindernis für spätere professionelle Einsätze dar.

Der semiprofessionelle Anwender findet mit dem CPC 6128 das richtige Gerät. Denn wer CP/M-Software benutzen will, der braucht mehr als 40 KByte Speicherplatz im RAM, die nur der 6128 zu bieten hat. Die zweiten 64 KByte von Basic aus als RAM-Floppy zu benutzen ist so umständlich gelöst, daß sich für jemanden, der unter Basic mehr als 42 KByte Speicherplatz braucht eher die Speichererweiterung von Vortex (sie wird einfach als Laufwerk C angesprochen) anbietet.

Den CPC 664 einzuordnen ist sicher am schwersten. Zum einen ist er mit 100 Mark nur unwesentlich billiger als der 6128, zum andern kann er aber den für CP/M notwendigen Speicherplatz nicht vorweisen. Wer aber nicht mit dem Kassettenrecorder arbeiten will und wem der geringe Speicherplatz ausreicht, der wird wohl auf den 664 zurückgreifen. Legt er hingegen auf »Schönheit« wert, dann ist er doch wieder beim 6128 angelangt, der von der Optik her den ästhetischsten Anblick der drei Geräte bietet.

Ob Farb- oder Grünmotor, das hängt vom Einsatz des Computers ab. Professionelle Anwender bevorzugen sicher den Grünmotor, der auch 80 Zeichen pro Zeile noch sauber wiedergibt. Wer mehr den spielerischen Aspekt seines Computers ausnutzen will, der sollte den Farbspeicher wählen. Die Farben sind zwar brillant, das Auflösungsvermögen des Monitors erweist sich jedoch für eine 80-Zeichen-Wiedergabe zu gering. (hg)



Die Platine des CPC 664

|A	|DRIVE		
|B	|ERA		
|CPM	|REN		
|DIR	|TAPE		
|DISC	|TAPE.IN		
|DISC.IN	|TAPE.OUT		
|DISC.OUT	|USER		

Die zusätzlichen Befehle des DOS

ABS	EOF	MID$	ROUND
AFTER	ERASE	MIN	RUN
AND	ERL	MOD	SAVE
ASC	ERR	MODE	SGN
ATN	ERROR	MOVE	SIN
AUTO	EVERY	MOVER	SOUND
BIN$	EXP	NEXT	SPACE$
BORDER	FILL	NEW	SPC
CALL	FIX	NOT	SPEED
CAT	FN	ON	SQ
CHAIN	FOR	ON BREAK	SQR
CHR$	FRAME	ON BREAK CONT	STEP
CINT	FRE	ON ERROR GOTO	STOP
CLEAR	GOSUB	ON SQ	STR$
CLEAR INPUT	GOTO	OPENIN	STRING$
CLG	GRAPHICS PAPER	OPENOUT	SWAP
CLOSEIN	GRAPHICS PEN	OR	SYMBOL
CLOSEOUT	HEX$	ORIGIN	TAB
CLS	HIMEM	OUT	TAG
CONT	IF	ON»expression«GOSUB	TAGOFF
COPYCHR$	INK	ON»expression«GOTO	TAN
COS	INKEY	PAPER	TEST
CREAL	INKEY$	PEEK	TESTR
CURSOR	INP	PEN	THEN
DATA	INPUT	PI	TIME
DEC$	INSTR	PLOT	TO
DEF	INT	PLOTR	TROFF
DEFINT	JOY	POKE	TRON
DEFREAL	KEY	POS	UNT
DEFSTR	LEFT$	PRINT	UPPER$
DEG	LEN	RAD	USING
DELETE	LET	RANDOMIZE	VAL
DERR	LINE	READ	VPOS
DI	LIST	RELEASE	WAIT
DIM	LOAD	REM	WEND
DRAW	LOCATE	REMAIN	WHILE
DRAWR	LOG	RENUM	WIDTH
EDIT	LOG10	RESTORE	WINDOW
EI	LOWER$	RESUME	WRITE
ELSE	MASK	RESUME NEXT	XOR
END	MAX	RETURN	XPOS
ENT	MEMORY	RIGHT$	YPOS
ENV	MERGE	RND	ZONE

Die Basic-Befehle der Schneider-Computer (die farbig hinterlegten Anweisungen gibt es nur beim CPC 664 und 6128).

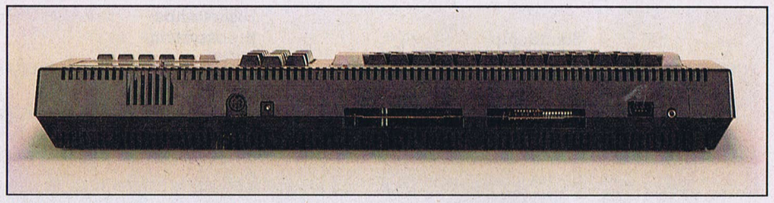
Schneider oder Amstrad — wie paßt denn das zusammen?

Genau ein Jahr ist es her, da stellte die Firma Schneider aus Türkheim mit dem CPC 464 ihren ersten Computer vor. Bis heute sind es drei, die gemeinsam mit der englischen Firma Amstrad auf den europäischen Markt gebracht wurden. Den neuesten, den CPC 6128, will Amstrad jetzt auch dem amerikanischen Publikum präsentieren. Nach Timex/Sinclair, die nur einen kurzlebigen Erfolg in den USA hatten, startet damit zum zweiten Mal ein Hersteller von Heimcomputern aus Großbritannien einen Versuch über den großen Teich zu gehen.

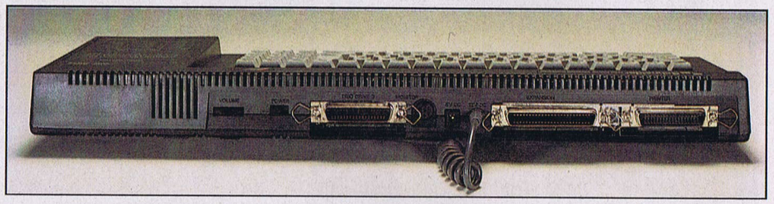
Schneider hingegen wird die Computer weiterhin nur im deutschsprachigen Raum (und einigen angrenzenden Ländern) verkaufen, da die beiden Firmen sich den Markt entsprechend aufgeteilt haben. Bis heute — und in naher Zukunft wird sich daran nichts ändern — sind die Geräte auch ausschließlich in Brentwood, dem Sitz von Amstrad, entwickelt worden. In Türkheim mußte man sich darauf beschränken, das, was aus England angeboten wurde, dem deutschen Markt anzupassen. Es blieben dann beispielsweise die blauen und grünen Tasten auf der Strecke, die den Amstrad CPC 464 deutlich vom Schneider CPC 464 unterscheiden. Grundsätzliches konnte aber nicht verändert werden. So zeigen — wie fast alle Computer aus England — auch die Geräte von Schneider Eigenheiten, die einige Gewöhnung erfordern. Es sei nur der sieben Bit breite Datenbus für den Drucker erwähnt.

Schneiders Einstieg in das Geschäft mit dem Computer erfolgte zu einer Zeit als der erste große Boom vorbei war. Die hohen Gewinne der Gründerzeit sind Vergangenheit, und einige — sogar größere bekannte Firmen — schreiben heute rote Zahlen. So war es nicht verwunderlich, daß viele der Schneider Computer Division im letzten Jahr wenig Chancen einräumten. Heute erweist sich das Gegenteil: Nach Commodore ist Schneider die Nummer 2.

Gemäß der Aussage von Bernhard Schneider ist sein Engagement bei den Heimcomputern nur einem glücklichen Zufall zu verdanken. »Vor wenigen Monaten bin ich von einem englischen Geschäftsfreund gefragt worden, ob wir die Vermarktung des Computers auf dem Kontinent übernehmen wollen. Die damalige Antwort eröffnete einen neuen Produktionszweig, der in der zweiten Stufe auch die Entwicklung eines eigenen Computers beinhalten soll«, waren seine Worte im letzten Jahr anläßlich der Vorstellung des CPC 464. Es bleibt abzuwarten, ob und wann die Einbahnstraße von Amstrad zu Schneider andersherum befahren wird. Bis dahin »muß« man sich mit den englischen Geräten zufrieden geben, was aber dank ihrer Leistungsfähigkeit, nicht schwer fällt. (hg)



Wenig stabil: Die Schnittstellen sind beim 464 und 664 nur einfach Platinenstecker



Gut gelöst: Die Schnittstelle des 6128 mit richtigen Büchsen

Name	CPC 464	CPC 664	CPC 6128
Massenspeicher:	Kassettenrecorder	Diskettenstation	Diskettenstation
Preis mit Grünmonitor:	899 Mark	1498 Mark	1598 Mark
Preis mit Farbmonitor:	1398 Mark	1998 Mark	2098 Mark
CPU (Taktfrequenz):	Z80A (4 MHz)*		
ROM:	32 KByte (auf über 4 MByte erweiterbar)*		
RAM:	64 KByte	64 KByte	128 KByte
Bildschirmauflösung:			
Modus 0:	160x200 (20 Zeichen pro Zeile, 16 Farben)*		
Modus 1:	320x200 (40 Zeichen pro Zeile, 4 Farben)*		
Modus 2:	640x200 (80 Zeichen pro Zeile, 2 Farben)*		
Zahl der Farben:	27*		
Soundprozessor:	AY-3-8912 (3 Stimmen)*		
Basic-Version (Zahl der Befehle):	1.0 (157)	1.1 (172 und 13 DOS-Befehle)	1.1 (172) und 13 (DOS-Befehle)
CP/M (Version):	nein	ja (2.2)	ja (2.2 und 3.1)
Schnittstellen:	Joystickport* Centronics (7 Bit)* Datenbus* Stereoanschluß*		
	—	DIN-Buchse für Tonband	DIN-Buchse für Tonband
	—	Anschluß für 2. Laufwerk	Anschluß für 2. Laufwerk
Tastatur:	drei Blöcke ASCII Ziffernblock Cursortasten	drei Blöcke ASCII Ziffernblock Cursortasten	ein Block ASCII mit Ziffernblock und Cursortasten

Die wichtigsten Daten der 3 Schneider-Computer *: Diese Ausgaben gelten für alle 3 Modelle

Schneider CPC Spiele-Software

Spiele-Software

Titel		Best.-Nr.	Preis
Gems of Stradus	(Amsoft)	1215	39,– DM
Laserwarp	(Amsoft)	1216	39,– DM
Haunted Hedges	(Amsoft)	1217	39,– DM
Galactic Plague	(Amsoft)	1223	39,– DM
Splatt	(Amsoft)	1224	39,– DM
Jammin	(Amsoft)	1226	39,– DM
Fruit Machine	(Amsoft)	1228	39,– DM
Mutant Monty	(Amsoft)	1229	39,– DM
Loopy Laundry	(Amsoft)	1230	39,– DM
Blagger	(Amsoft)	1231	39,– DM
Classic Racing	(Amsoft)	1232	39,– DM
Crazy Golf	(Amsoft)	1233	39,– DM
Roland on the run	(Amsoft)	1234	39,– DM
Detective (Cludo)	(Amsoft)	1235	39,– DM
Hunter Killer	(Amsoft)	1237	39,– DM
Jet Boot Jack	(Amsoft)	1238	39,– DM
Ghouls	(Micro Power)	1242	29,– DM
The Prize	(Amsoft)	1243	39,– DM
Sultans Maze	(Amsoft)	1245	39,– DM
Oh Mummy	(Amsoft)	1247	39,– DM
3D Invaders	(Amsoft)	1249	39,– DM
Alien Break in	(Romik)	1250	27,– DM
Atom Smasher	(Romik)	1251	27,– DM
3D Monster Chase	(Romik)	1252	27,– DM
Dragons Gold	(Romik)	1253	27,– DM
Star Commando	(Terminal)	1255	27,– DM
Electro Freddy	(Amsoft)	1257	39,– DM
Punchy	(Amsoft)	1259	39,– DM
Stockmarket	(Amsoft)	1260	39,– DM
Bridge it	(Amsoft)	1261	39,– DM
Cubit	(Amsoft)	1262	39,– DM
Grand Prix Drive	(Amsoft)	1263	39,– DM
Killer Gorilla	(Micro Power)	1270	49,– DM
Sorcery	(Virgin)	1273	29,– DM
Map Rallye	(Amsoft)	1239	34,– DM
Traffic	(Amsoft)	1241	34,– DM
Xanagrams	(Amsoft)	1256	34,– DM
Admiral Graf Spee	(Amsoft)	1264	34,– DM
Roland goes digging	(Amsoft)	1265	34,– DM
Space Hawks	(Amsoft)	1267	34,– DM
Quack a Jack	(Amsoft)	1268	34,– DM
Classic Adventure	(Amsoft)	1271	34,– DM
American Football	(Amsoft)	1275	34,– DM
Frank-N-Stein	(Amsoft)	1276	34,– DM
Super Pipeline II	(Amsoft)	1277	34,– DM
The Scouts steps out	(Amsoft)	1278	34,– DM
Stud Poker/Stud Jack	(Data Media)	1211	39,– DM
Disk 3"		1411	49,– DM
Disk 5 1/4"		1611	49,– DM
Skat/Bauernskat	(Data Media)	1212	39,– DM
Disk 3"		1412	49,– DM
Disk 5 1/4"		1612	49,– DM
Poker/17+4	(Data Media)	1213	39,– DM
Disk 3"		1413	49,– DM
Disk 5 1/4"		1613	49,– DM
Zargon	(Data Media)	1214	39,– DM
Disk 3"		1414	49,– DM
Disk 5 1/4"		1614	49,– DM
Codename Mat	(Amsoft)	1218	39,– DM
Disk 3"		1418	49,– DM
Snooker	(Amsoft)	1220	39,– DM
Disk 3"		1420	49,– DM
Roland ahoy	(Amsoft)	1222	39,– DM
Disk 3"		1422	49,– DM
Pyjamarama	(Amsoft)	1236	39,– DM
Disk 3"		1436	49,– DM
Spannerman	(Amsoft)	1246	39,– DM
Disk 3"		1446	49,– DM
Masterchess	(Microgen)	1248	39,– DM
Disk 3"		1448	49,– DM
Roland in time	(Amsoft)	1266	39,– DM
Disk 3"		1466	49,– DM

Amsoft-Produkte werden exklusiv von Data Media in Deutschland, Österreich und der Schweiz vertrieben

ER*Bert	(Microbyte)	1272	39,– DM
Disk 3"		1472	49,– DM
Disk 5 1/4"		1672	49,– DM
Future World	(Data Media)	1289	39,– DM
Disk 3"		1489	49,– DM
Disk 5 1/4"		1689	49,– DM
Enterprise	(Data Media)	1294	49,– DM
Disk 3"		1494	59,– DM
Disk 5 1/4"		1694	59,– DM
Der Rote Baron	(Data Media)	1295	39,– DM
Disk 3"		1495	49,– DM
Disk 5 1/4"		1695	49,– DM
Das Geheimnis der 4 Juwelen	(Data Media)	1296	39,– DM
Disk 3"		1496	49,– DM
Disk 5 1/4"		1696	49,– DM
Pittmans Typing Tutor	(Amsoft)	1240	49,– DM
Disk 3"		1440	54,– DM
Harrier Attack	(Amsoft)	1244	39,– DM
Disk 3"		1444	49,– DM
Amsgolf	(Amsoft)	1269	34,– DM
Disk 3"		1469	44,– DM

Neu **Marco Polo I** (Data Media)

Die Marco Polo Saga. Ein deutsches Grafik-Adventure.

Marco Polo 1	Disk 3"	1497	49,– DM
	Disk 5 1/4"	1697	49,– DM
Marco Polo 2	Disk 3"	1498	49,– DM
	Disk 5 1/4"	1698	49,– DM
Marco Polo 3	Disk 3"	1499	49,– DM
	Disk 5 1/4"	1699	49,– DM

Die Thorr-Trilogie (Data Media)

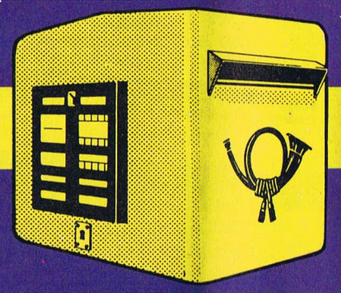
Eine der komplexesten deutschen Textadventure-Trilogien. Erleben Sie Thorr, die Welt der Zauberer, Ritter und der Schwarzen Magie.
Die Thorr-Adventures sind auch einzeln zu erhalten:

Die letzten Tage von Burg Ghorrodt		1290	39,– DM
Disk 3"		1490	49,– DM
Disk 5 1/4"		1690	49,– DM
Die Fluch nach Thyrros		1291	39,– DM
Disk 3"		1491	49,– DM
Disk 5 1/4"		1691	49,– DM
Das Geheimnis von Thyrros		1292	39,– DM
Disk 3"		1492	49,– DM
Disk 5 1/4"		1692	49,– DM
Die Thorr-Trilogie komplett		1293	99,– DM
Disk 3"		1493	109,– DM
Disk 5 1/4"		1693	109,– DM

Bücher

Huslik:	
Inside out, 428 Seiten	**59,00 DM**
Data Becker:	
Das Floppy-Buch zum CPC, ca. 250 Seiten	**49,00 DM**
CPC Hardware-Erweiterungen, ca. 300 Seiten	**49,00 DM**
Peeks & Pokes zum CPC, ca. 220 Seiten	**29,00 DM**
CPC 464 Graphik & Sound, 220 Seiten	**39,00 DM**
CPC 464 Intern, 548 Seiten	**69,00 DM**
Das Maschinensprachebuch zum CPC 464, 330 Seiten	**39,00 DM**
CPC Tips & Tricks, 263 Seiten	**39,00 DM**
Adventures – und wie man sie auf dem CPC 464 programmiert, 320 S.	**39,00 DM**
CPC 464 Basic-Programme, 185 Seiten	**39,00 DM**
Das Basic-Trainingsbuch zum CPC 464, 285 Seiten	**39,00 DM**
Das Schulbuch zum CPC 464, 389 Seiten	**49,00 DM**

Wie bestellen?
Bestellung an DATA MEDIA senden. Lieferung erfolgt umgehend! Zahlung per Vorkasse oder Nachnahme zzgl. Porto- und Nachnahmegebühr. (Nachnahme ins Ausland ist nicht möglich.)



data media gmbh MAILORDER

Weitere Artikel in unserem Gesamtkatalog. Bitte anfordern (2,– DM für Rückporto beilegen).
DATA MEDIA GMBH, Bereich Mailorder, Postfach 1263, 4620 Castrop-Rauxel
Telefonische Auskünfte: 0231/125071

Nützliche Tips zur Eingabe der Listings

Daß man die Tastatur und den Zeichensatz der Schneider-Computer sehr leicht umdefinieren kann, hat dazu geführt, daß fast jeder mit einer anderen Belegung arbeitet. Für dieses Heft haben wir uns auf einen Standard geeinigt.

Viele Besitzer eines CPC arbeiten mit einer deutschen Tastatur. Da das gesamte Tastenfeld durch entsprechende Basic-Befehle sehr leicht umdefiniert werden kann, ist solch eine Erweiterung auch kein Problem. Schwierigkeiten treten erst dann auf, wenn man ein Listing ausdrucken und an andere weitergeben will. Denn ein Drucker, der mit dem ASCII-Zeichensatz arbeitet, bringt anstelle der deutschen Sonderzeichen nur ungewohnte Symbole (teilweise sind diese auf der Tastatur gar nicht zu finden) zu Papier.

Bei den Listings in diesem Heft haben wir uns deshalb darauf geeinigt, alle Listings mit dem ASCII-Zeichensatz auszudrucken. Statt der deutschen Sonderzeichen sind deshalb die des amerikanischen Zeichensatzes zu finden (Klammern, senkrechter Strich und so weiter). Mit einer Ausnahme sind sie alle auf der normalen Tastatur angegeben. Nur das »~« als Äquivalent zum deutschen »ß« muß etwas umständlich durch gleichzeitiges Drücken der »2« und der Ctrl-Taste eingegeben werden. Wer seinen Computer mit einer kleinen Routine »eingedeutscht« hat (zum Beispiel mit dem Programm von Michael Straßer in diesem Heft), der findet auch auf dem Monitor die gewünschten deutschen Buchstaben.

Der Pfeil nach oben (»↑«) des CPC wird vom Drucker nur als »^« wiedergegeben. Sofern nichts anderes angegeben ist, können die Programme auf allen drei Schneider-Computern benutzt werden. In der folgenden Liste finden Sie noch einmal zusammengefaßt die amerikanischen Sonderzeichen und ihre deutschen Äquivalente. (hg)

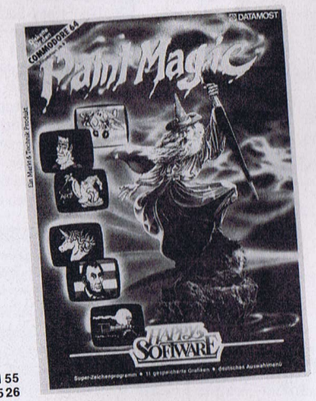
Sonderzeichen

amerikanische	deutsche	Bemerkungen
@	§	---
[	Ä	---
\	Ö	---
]	Ü	---
{	ä	---
|	ö	---
}	ü	
~	ß	Ctrl-Taste und »2«

Das Symbol »^« steht für »↑«.

Paint Magic
Das magische Zeichenprogramm aus den USA für Ihren Commodore 64

- elf gespeicherte »Traumbilder«
- gleichzeitiges Malen auf zwei Bildschirmen
- einfache Bedienung durch übersichtliche Menütechnik
- eigenes Farbmenü (16 Farben)
- umfangreiche Diskettenbefehle (Speichern, Löschen, Laden)
- 100% Maschinensprache



★ Deutsches Auswahlmenü auf Diskette
★ Deutsches Anleitungsheft

DM 59,- inkl. MwSt. unverbindliche Preisempfehlung (Sfr. 54,50/öS 460,20) Bestell-Nr. MD 230 A

Markt & Technik-Programme erhalten Sie bei Ihrem Buchhändler.

Bestellkarten bitte an Ihren Buchhändler oder an eine unserer Depotbuchhandlungen. Adressenverzeichnis am Ende des Heftes. Beim Markt & Technik Verlag eingehende Bestellungen werden von den Depot-Händlern ausgeliefert.

Markt&Technik
Verlag Aktiengesellschaft
Unternehmensbereich Buchverlag

Hans-Pinsel-Straße 2, 8013 Haar bei München
Schweiz: Markt & Technik Vertriebs AG, Kollerstraße 3, CH-6300 Zug, ☎ 042/22 31 55
Österreich: Rudolf Lechner & Sohn, Heizwerkstraße 10, A-1232 Wien, ☎ 0222/67 75 26

Werden Sie mit den »magischen Malereien« zum »elektronischen Künstler!«
Sie brauchen Ihren Commodore 64 — ein Diskettenlaufwerk — Joystick.

Der Schneider im Reich der Farben

CPC — Colour Personal Computer, das ist ein Titel, der Ansprüche weckt. Und der Schneider wird ihnen auch in vielfältiger Weise gerecht. Diagramme, Blockgrafik und Sonderzeichen lernen Sie in diesem Artikel kennen.

Ein Bild sagt mehr als tausend Worte«, so lautet ein altes chinesisches Sprichwort, das damit daran erinnert, wie Informationen »menschengerecht« aufgearbeitet sein sollten. Um so erstaunlicher ist es, daß man bei der Entwicklung von Computern auf der Eingabe- wie der Ausgabeseite fast nur seitenlange Texte findet. Diese »Grundregel« gilt auch für den Schneider. Die Sprache dominiert, sei es als Programm oder erklärender Text. Nun hat man aber im Grafikbereich beim Schneider eine Grenze überschritten, jedenfalls in dieser Preisklasse. Die Rede ist davon, Grafik und Text ohne Umschaltung oder besondere Tricks gleichzeitig auf den Schirm zu bringen. Damit sind eigentlich die idealen Voraussetzungen geschaffen, um Programme auch ohne ausgefuchste Konstruktionen, mit einfachen Diagrammen und Zeichnungen aufzuwerten, sie benutzbarer und komfortabler zu gestalten. Im folgenden wollen wir uns damit beschäftigen, was beim Schneider im Bereich der Grafik machbar ist, und mit welchen Tricks man noch etwas mehr herausholen kann.

Als erstes Gebiet und gleichsam Paradebeispiel für den Grafikeinsatz, fällt einem meist die grafische Statistik ein. Ihre Aufgabe besteht darin, abstrakte Zahlenwerte absolut oder in ihrem Verhältnis zueinander darzustellen. Wir alle kennen die dabei angewandten bildlichen Darstellungsmittel aus den täglichen Presseberichten im Fernsehen oder in Zeitungen. Kein Wirtschaftsbericht kann heute bestehen, ohne das obligatorische Säulendiagramm, das die Kennwerte für das letzte halbe Jahrzehnt widerspiegelt und optisch einen Eindruck von der Entwicklung der diskutierten Größe vermittelt. Bei Bundestagswahlen faszinieren immer wieder die Torten- oder Kreisdiagramme, die, blitzartig auf den Bildschirm gezaubert, die voraussichtliche Sitzverteilung im neuen Parlament wiedergeben. Die Anwendung grafischer Darstellungsmittel ist jedoch nicht auf Wirtschaft und Politik, auch nicht auf die Verdeutlichung abstrakter Zahlen beschränkt. Schon von der Schule her sind uns noch andere Anwendungen, speziell im Bereich der grafischen Analyse, vertraut. Eine der bekanntesten ist dabei die grafische Darstellung einer Funktion. Diese liefert einen Überblick über den Verlauf der Funktion sowie bestimmter Eigenschaften (Maxima, Minima, Nullstellen etc.).

Schauen wir uns einmal an, welche Darstellungsarten wir beim CPC benutzen können und für welchen Zweck sie jeweils geeignet sind. Den Anfang soll das am häufigsten benutzte Diagramm machen, das Balkendiagramm. Bei ihm werden verschiedene Werte durch unterschiedlich hohe Rechtecke beziehungsweise Balken repräsentiert. Die Breite der Figuren kann sich dabei beispielsweise aus der verstrichenen Zeit (X-Achse) ergeben, oder ist fest vorgegeben. Mit der Breite der Figuren hat man meist auch die geringeren Probleme. Viel größere Sorgen bereitet die Wahl der maximalen Darstellungshöhe. Da diese Schwierigkeit bei allen Darstellungsarten auftritt, soll es hier als erstes beschrieben werden.

Grafikhöhe selbst bestimmt

Nehmen wir einmal an, Sie wollen ein Programm schreiben, das es erlaubt, zehn verschiedene Werte, die in einem Feld W(i) als w(1), w(2)... w(10) abgelegt sind, darzustellen. Wenn der Wertebereich, in dem die darzustellenden Zahlenwerte liegen, nicht fest vorgegeben ist, stellt sich das Problem des Maßstabs. Wählt man einen relativ großen (kleiner Maximalwert auf der Y-Achse) und gibt diesen fest vor, so werden zwar die Unterschiede in der Höhe der Balken sehr deutlich erkennbar, aber die Gefahr von Bereichsüberschreitungen ist sehr groß. Die einzelnen Blöcke wachsen dann über den oberen Bildschirmrand hinaus. Setzen wir den Maßstab dagegen zu klein an, so haben wir zwar die Gewißheit, daß alle Blöcke richtig abgebildet werden, aber gegebenenfalls sind die Unterschiede nur noch schwach erkennbar und im Extremfall reduzieren sich unsere Blöcke auf kleine Striche an der Bildunterkante. Wir haben es hier also mit einem Optimierungs-Problem zu tun, das nur, speziell wenn die Werte relativ weit schwanken, durch eine automatische Maßstabswahl lösbar ist. Der optimale Punkt ist dann erreicht, wenn der größte Wert das für die Darstellung vorgewählte Bildschirmfenster gerade ausfüllt. Dazu müssen wir das Maximum und das Minimum der Funktionswerte kennen. Dies ist mit einer FOR...TO...-Schleife und zwei speziellen Befehlen im Schneider-Basic relativ einfach zu bewerkstelligen. Das Kommando MAX berechnet das Maximum einer Folge von Zahlen, MIN das Minimum.

```
100  Maßstabswahl; en=max.Anzahl der Werte; Werte in w(i) gespeichert
110  en=10:mi=1E37:ma=-1E37
120  FOR i=1 TO en:ma=MAX(ma,w(i)):mi=MIN(mi,w(i)):NEXT
```

Nach Durchlauf der Schleife haben wir die Extremwerte, die in dem Feld w(i) gespeichert sind, bestimmt. Setzen Sie einmal ein paar Werte für (w)i ein und geben Sie dann GOTO 100 ein. Die Variablen und mi und ma enthalten danach die gewünschten Extremwerte. Mit deren Hilfe können wir nun zwischen zwei Darstellungsarten wählen. Im ersten Fall bildet die Nullinie die Untergrenze unserer Grafik, im anderen ist der Bezugspunkt das Minimum unserer Werte. Jede Variante hat ihre Vor- und Nachteile. Die Nullinie bildet einen eindeutigen Bezugspunkt, der nicht nur die relativen Unterschiede, sondern auch die absoluten Werte direkt verdeutlicht. Allerdings ist diese Darstellungsart nicht geeignet, wenn auch negative Zahlen dargestellt werden sollen oder die wiederzugebenden Werte nur geringe Änderungen auf dem hohen Niveau aufweisen. Dies ist zum Beispiel bei der Darstellung des Bruttosozialproduktes der Fall.

Hier empfiehlt sich die Verwendung der zweiten Alternative, die allerdings auch ihre Schwächen hat, da sie die Aussage verfälscht. In der folgenden Anwendung beider Darstellungen, können Sie von Fall zu Fall entscheiden, welche günstiger für Sie ist.

Von den Werten zur Grafik

Zunächst einmal zur Realisierung der ersten Variante: Beim CPC gibt es grundsätzlich zwei Möglichkeiten Balkendiagramme zu erzeugen. Die Verwendung der hochauflösenden Grafik und die Benutzung von Windows, was Ihnen vielleicht auf den ersten Blick »etwas spanisch« vorkommt. Die wohl schnellste Art beim CPC ein Rechteck auf dem Bildschirm zu erzeugen, bilden ohne Zweifel die Windows. Definieren Sie einmal nach dem Einschalten »WINDOW #3,10,20,10,20:PAPER #3,2:CLS #3«

Als Ergebnis sehen Sie ein hellblaues Rechteck auf dem Bildschirm. Spielen Sie dieselbe Befehlsfolge mit PAPER #3,3 durch, »sieht« der CPC rot. Sie können aber auch die Parameterangaben im Window-Kommando verändern und erhalten dann an anderer Position (oder auch derselben) ein neues Rechteck. Damit haben Sie schon alle Informationen, die Sie brauchen, um ein Balkendiagramm mit Windows zu erzeugen. Bleibt nur noch das Problem verschiedene Windows nebeneinander zu definieren, wobei bei jedem die Farbe gewechselt werden sollte, um eine optimale Übersichtlichkeit und Trennung zu gewährleisten. Eine Überschneidung ist tunlichst zu vermeiden.

Bei der Definition mit Windows sind aber noch ein paar andere Punkte zu beachten. Der Schneider definiert die Zeilen und Spalten so, daß der Punkt 1,1 in der linken oberen Ecke liegt. Wir benötigen aber eine umgekehrte Richtung, da schließlich bei größeren Werten die Ausdehnung nach oben erfolgen soll. Bei 24 Zeilen müßte also Zeile 24 die höchste Zeile sein, was aber leicht zu machen ist. Subtrahieren wir nämlich unseren Ergebniswert von 25, erhalten wir gerade diesen umgekehrten Verlauf.

Die Entscheidung wie breit ein Balken ausfallen soll ist abhängig vom ausgewählten Mode. Bei Mode Null und zehn darzustellenden Werten würde man hier wahrscheinlich eine 2 wählen. Das erste Window liegt dann auf den Spalten 1 und 2, das zweite benutzt 3 und 4, Window Nummer 10 wird in den letzten beiden Spalten eröffnet. Im Mode 1 kann man hier problemlos die doppelte Spaltenzahl benutzen. Erst dann wird der gesamte Bildschirm ausgefüllt. Als letzter Punkt wäre noch die Farbwahl in den verschiedenen Modi zu klären. Gehen wir davon aus, daß der Hintergrund mit INK 0 beschrieben ist; uns stehen dann die weiteren Farbregister vollständig zur Verfügung. Der Farbwechsel ist mit der Modulo-Funktion elegant zu lösen. Diese Funktion liefert als Ergebnis den Rest einer Division. Wenn wir zum Beispiel eine Zahl MOD 3 nehmen, so schwankt das Ergebnis immer zwischen 0 und 2. Analog dazu die Situation bei MOD 15, wobei das Ergebnis auf den Bereich von 0 bis 14 beschränkt ist. Addieren wir zu diesen Werten jetzt jeweils eine 1, so erhalten wir Zahlen zwischen 1 und 3 beziehungsweise 1 und 15, und damit genau die Farbregisteraufrufe, die wir für die Modi 1 und 0 benötigen. Da die MOD-Funktion bei einer fortlaufenden Zahlenfolge zyklischen Charakter hat (beim Überschreiten der 14 bei MOD 15 geht es wieder von vorne los), brauchen wir uns um die Farbumschaltung auch bei größeren Zahlenmengen keine Sorge mehr zu machen. Das leistet der Schneider dann selbsttätig. Ein Programm, das nach diesen Gedanken aufgebaut ist, finden Sie in Listing 1. Es zeigt grafisch die Umsätze eines Jahres. Diese sind dabei in dem Feld m(i) abgespeichert, in m$(i) die zugehörigen Monatsnamen. Die Beschränkung auf 12 Werte resultiert dabei aus dem Monatsrhythmus. Man beginnt mit der Eingabe der Einzelergebnisse, zu denen in Zeile 140 und 150 die Maximumberechnung durchgeführt werden. Ab Zeile 170 beginnt die eigentliche Grafikausgabe. Dies geschieht in einer Schleife, wobei die Farbe (f) mit Hilfe der MOD-Funktion bestimmt und dann als Farbe des WINDOW #4 mit dem PAPER-Kommando definiert wird. Es folgt die Berechnung der Höhe nach folgender Formel: (Wert/Maximum der Werte) x 16. Durch die Window-Definition wird das Fenster im Bereich von Zeile 4 bis Zeile 20 gesetzt, so daß oberhalb und unterhalb unserer Grafik noch etwas Platz für die Ausgabe von Titeln oder Achsenbenennungen bleibt.

Mit der Beschränkung auf Windows erscheint unsere Grafik zwar sehr schnell auf dem Schirm, man ist jedoch in puncto Auflösung und Anzahl der darzustellenden Werte eingeengt. Um dem abzuhelfen, muß man auf die hochauflösende Grafik zurückgreifen. Das Vorgehen ist ähnlich, wie bei der Window-Grafik, nur daß hier die Rechtecke nicht mehr am Stück definiert werden können. Dafür ist aber die Möglichkeit gegeben, Linien zu ziehen. Und aus mehreren Parallelen erhalten wir wiederum ein Rechteck. Das ist auch der ganze Trick, der uns zum nächsten Programm führt. Anwendungsbeispiel ist diesmal, etwas haushaltsnäher, eine Stromkostenabrechnung.

Nach den üblichen Vorbereitungsarbeiten, wie Einlesen der Texte, Ausgabe des Titels »Balkendiagramm« (Listing 2), der Farbgebung und der Abfrage der entsprechenden Stromkosten in einer Schleife, kommen wir zur Bestimmung der Maxima. Es schließt sich die Abfrage über die Blockbreite an, wo festgelegt wird, wieviele Linien ein Block haben soll. Der Bereich sinnvoller Eingaben beschränkt sich dabei auf die Werte zwischen 5 und 50. Der nächste Teil (ab Zeile 200) bildet die eigentliche Darstellungsroutine. Zunächst setzen wir das Zentrum unserer Grafik mit »ORIGIN 0,0« in die linke untere Ecke und löschen den Bildschirm. Die Grafik selbst wird in den nächsten drei Zeilen in den geschachtelten FOR...TO...-Schleifen gemalt. Die äußere durchläuft dabei die Monate, die innere die Linien eines Blocks. Für jeden Monat wird zunächst die Farbe festgelegt und danach die Höhe des entsprechenden Blocks aus den eingegebenen Werten und dem vorher festgelegten Maßstab errechnet. Weiter geht es mit der inneren Schleife, in der mittels der Laufvariablen i und j und der Balkenbreite der Fußpunkt der zu zeichnenden Linie auf der X-Achse ermittelt wird. Danach bewegt das MOVE-Kommando den Grafik-Cursor auf diese Position. Mit dem relativen DRAW-Befehl ziehen Sie eine Linie von entsprechender Höhe in der Farbe F. Das Ganze wiederholt sich so lange, bis der Block mit der gewünschten Breite fertiggestellt ist. Danach wird das Ganze für den nächsten Monat wiederholt.

Wenden wir uns nun den anderen Darstellungsarten zu. Das Stabdiagramm (Listing 3) basiert auf demselben Prinzip wie das Balkendiagramm. Auch hier repräsentiert die Höhe der Stäbe die Größe der entsprechenden Werte. Das Stabdiagramm ist immer dann als Darstellungsmittel geeignet, wenn eine

Vielzahl von Werten wiedergegeben werden soll. Wie können wir nun ein Stabdiagramm erzeugen? Der einzige Unterschied zum Balkendiagramm besteht darin, daß sich nicht die verschiedenen Blöcke untereinander abdecken, die insgesamt die volle Breite einnehmen, sondern, daß sich Stäbe und Hintergrundfarbe einander abwechseln. Was auf den ersten Blick schwierig aussieht, ist in Wirklichkeit jedoch eine Vereinfachung des eben betrachteten Programms. Löschen wir den Bildschirm und zeichnen wir statt eines ganzen Blockes nur eine Linie, so erhalten wir genau diesen Effekt. Wenn Sie in Zeile 260 j nur noch von 1 bis 1 laufen lassen, ist das Ergebnis genau diese Ausgabe. Sie können j auch über zwei Punkte laufen lassen, dann wird der Stab deutlicher sichtbar.

Vor etwas größere Probleme stellt uns die Ausgabe von Daten in Form eines Kuchendiagramms. Die Schwierigkeit liegt darin, daß wir eine Kreisfläche verschiedenfarbig unterteilen, was mittels Linien relativ schwer realisierbar ist, besonders

```
10 REM ********************
20 REM ** Saeulengrafik **
30 REM ********************
40 INK 0,27:INK 1,21:PAPER 0:PEN 1:BORDE
R 0
50 DATA Januar,Februar,Maerz,April,Mai,J
uni
60 DATA Juli,August,September,Oktober,No
vember,Dezember
70 DIM m$(12),m(12)
80 FOR i=1 TO 12:READ m$(i):NEXT i
90 CLS
100 FOR i= 1 TO 12
110 PRINT"Bitte geben Sie den Umsatz fue
r den Mo-"
120 PRINT"nat ":m$(i):" ein"::INPUT m(i)
130 NEXT i
140 ma=0:mi=10000000
150 FOR i= 1 TO 12:ma=MAX(ma,m(i)):mi=MI
N(mi,m(i)):NEXT i
160 CLS
170 LOCATE 7,2:PRINT"W I N D O W - G r a
 f i k"
180 INK 0,0:INK 1,2:INK 2,15:INK 3,21:BO
RDER 0
190 FOR i= 1 TO 12
200 f=(i MOD 3)+1:PAPER#4, f
210 h=m(i)/ma*16
220 WINDOW#4,3*i-1,3*i+1,20-h,20:CLS#4
230 NEXT i
240 PEN 2
250 LOCATE 2,22:PRINT" J  F  M  A  M  J
 J  A  S  O  N  D"
260 GOTO 260
```

Listing 1. Daten lassen sich leicht in einer Säulengrafik veranschaulichen

```
10 REM ****************************
20 REM ** Saeulengrafik mit DRAW **
30 REM ****************************
40 INK 0,27:INK 1,21:PAPER 0:PEN 1:BORDE
R 0
50 DATA Januar,Februar,Maerz,April,Mai,J
uni
60 DATA Juli,August,September,Oktober,No
vember,Dezember
70 DIM m$(12),m(12)
80 FOR i=1 TO 12:READ m$(i):NEXT i
90 CLS
100 FOR i= 1 TO 12
110 PRINT"Bitte geben Sie die Stromkoste
n fuer denMonat ";
120 PRINT m$(i);" ein"::INPUT m(i)
130 NEXT i
134 REM ******************
135 REM ** Maximalwert **
136 REM ******************
140 ma=0:mi=10000000
150 FOR i= 1 TO 12:ma=MAX(ma,m(i)):mi=MI
N(mi,m(i)):NEXT i
160 CLS
170 LOCATE 7,2:PRINT"B A L K E N D I A G
 R A M M"
180 PRINT"Bitte geben Sie die Breite der
 Bloecke  an (5-50)"::INPUT b
190 IF b<5 OR b>50 THEN PRINT"Fehler":GO
TO 180
200 CLS:LOCATE 7,2:PRINT"B A L K E N D I
 A G R A M M"
210 INK 0,0:INK 1,2:INK 2,15:INK 3,21:BO
RDER 0
220 ORIGIN 0,0
230 FOR i= 1 TO 12
240 f=(i MOD 3)+1
250 h=m(i)/ma*250
260 FOR j=1 TO b:MOVE b*i+j,80:DRAWR 0,h
,f
270 NEXT j,i
280 PEN 2
290 GOTO 290
```

Listing 2. Mit DRAW ist eine Säulengrafik viel genauer zu gestalten

```
10 REM ****************************
20 REM ** Stabdiagramm mit DRAW **
30 REM ****************************
40 INK 0,27:INK 1,21:PAPER 0:PEN 1:BORDE
R 0
50 DATA Januar,Februar,Maerz,April,Mai,J
uni
60 DATA Juli,August,September,Oktober,No
vember,Dezember
70 DIM m$(12),m(12)
80 FOR i=1 TO 12:READ m$(i):NEXT i
90 CLS
100 FOR i= 1 TO 12
110 PRINT"Bitte geben Sie die Stromkoste
n fuer denMonat ";
120 PRINT m$(i);" ein"::INPUT m(i)
130 NEXT i
140 REM ******************
150 REM ** Maximalwert **
160 REM ******************
170 ma=0:mi=10000000
180 FOR i= 1 TO 12:ma=MAX(ma,m(i)):mi=MI
N(mi,m(i)):NEXT i
190 INK 0,0:INK 1,2:INK 2,15:INK 3,21:BO
```

Listing 3. Unterschiedliche Funktionswerte werden oft als Stabdiagramm ausgegeben

```
RDER 0
200 CLS:LOCATE 10,2:PRINT"S T A B D I A
G R A M M"
210 INK 0,0:INK 1,2:INK 2,15:INK 3,21:BO
RDER 0
220 REM ********************
230 REM ** Staebe ziehen **
240 REM ********************
250 ORIGIN 0,0
260 FOR i= 1 TO 12
270 f=(i MOD 3)+1
280 h=m(i)/ma*250
290 MOVE 25*i+166,80:DRAWR 0,h,f
300 MOVE 25*i+168,80:DRAWR 0,h,f
310 MOVE 160,80:DRAWR 330,0,2
320 NEXT I
330 PEN 2
340 GOTO 340
```

Listing 3. Unterschiedliche Funktionswerte werden oft als Stabdiagramm ausgegeben (Schluß)

```
10 REM ********************
20 REM ** Kuchendiagramm **
30 REM ********************
40 INK 0,27:INK 1,21:PAPER 0:PEN 1:BORDE
R 0
50 DATA Januar,Februar,Maerz,April,Mai,J
uni
60 DATA Juli,August,September,Oktober,No
vember,Dezember
70 DIM m$(12),m(12),re(12)
80 FOR i=1 TO 12:READ m$(i):NEXT i
90 CLS
100 FOR i= 1 TO 12
110 PRINT"Bitte geben Sie die Stromkoste
n fuer denMonat ";
120 PRINT m$(i);" ein"::INPUT m(i)
130 NEXT i
140 REM ********************************
150 REM ** relativen Anteil berechnen **
160 REM ********************************
170 sum=0:CLS
180 FOR i=1 TO 12:sum=sum+m(i):NEXT i
190 FOR i=1 TO 12:re(i)=m(i)/sum:NEXT i
200 re(1)=re(1)+re(2)+re(3):re(2)=re(1)+
re(4)+re(5)+re(6)
210 re(3)=re(2)+re(7)+re(8)+re(9):re(4)=
re(3)+re(10)+re(11)+re(12)
220 REM ********************
230 REM ** Kreis zeichnen **
240 REM ********************
250 CLS
260 INK 0,0:INK 1,24:INK 2,6:INK 3,2
270 PEN 3
280 LOCATE 7,3:PRINT"K r e i s d i a g r
 a m m"
290 ORIGIN 300,200
300 re(1)=re(1)*2*PI:re(2)=re(2)*2*PI
310 re(3)=re(3)*2*PI:re(4)=re(4)*2*PI
320 FOR i=0 TO re(1) STEP 0.0051:a=SIN(i
)*50
330 b=COS(i)*50:MOVE 0,0:DRAWR a,b,1:NEX
T i
340 FOR i=re(1) TO re(2) STEP 0.0052:a=S
IN(i)*50
350 b=COS(i)*50:MOVE 0,0:DRAWR a,b,2:NEX
T i
360 FOR i=re(2) TO re(3) STEP 0.0052:a=S
IN(i)*50
370 b=COS(i)*50:MOVE 0,0:DRAWR a,b,1:NEX
T i
380 FOR i=re(3) TO re(4) STEP 0.0052:a=S
IN(i)*50
390 b=COS(i)*50:MOVE 0,0:DRAWR a,b,3:NEX
T i
400 GOTO 400
```

Listing 4. Feine Unterschiede sind mit dem Kuchendiagramm schlecht zu erkennen

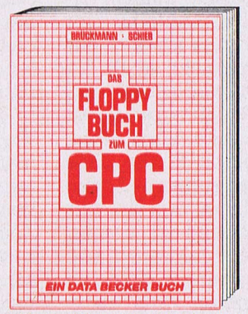
```
10 ' ********************
20 ' 3-D-3-Ebenengrafik
30 ' ********************
40 INK 0,0:INK 1,11:INK 2,6:INK 3,24
50 CLS:LOCATE 10,2:PRINT"3-D-3-EBENENGRA
FIK":PRINT:PRINT
60 INPUT"Bitte geben Sie den  Wert einer
  Einheitin der Vertikalen an";diff
70 INPUT"Wieviele Saeulen pro Ebene (2-6
)";zs:'Zahl der Saeulen
80 INPUT"Wieviele Ebenen (1-3)";ze:'Zahl
 der Ebenen
90 DIM w(ze,zs)
100 FOR i=0 TO ze-1
110 FOR j= 1 TO zs
120 PRINT:PRINT"Bitte geben Sie den Wert
 fuer Ebene";i+1;"Saeule";j;"ein";:INPUT
 w
130 IF INT(w/diff)>19 THEN PRINT:PRINT"W
ert zu gross!!":GOTO 120
140 w(i,j)=w
150 NEXT j,i
160 CLS
170 PEN 1:FOR i=0 TO 3:LOCATE 5+i,21+i:P
RINT CHR$(213);STRING$(29,CHR$(143));CHR
$(215)
180 NEXT i
190 FOR h=0 TO ze-1
200 PEN h+1
210 FOR i=zs TO 1 STEP-1
220 IF INT(w(h,i)/diff)=0 THEN w(h,i)=di
ff
230 FOR j=1 TO INT(w(h,i)/diff)+1
240 LOCATE 8+4*(i-1)+h,22+h-j
250 IF j=1 THEN PEN 1:PAPER 0:PRINT CHR$
(143);CHR$(215);:PEN h+1:PRINT CHR$(143)
260 IF j>1 AND j<INT(w(h,i)/diff)+1 THEN
 PRINT CHR$(209);" ";CHR$(143)
270 IF j=INT(w(h,i)/diff)+1 THEN PRINT C
HR$(209);CHR$(213);CHR$(215)
280 NEXT j,i,h
290 FOR i=0 TO zs-1
300 FOR j=1 TO ze
310 LOCATE 8+4*i+j,20+j
320 IF j>1 OR w(2,i)=0 OR (j=1 AND ze=0)
 THEN PEN 1:PRINT CHR$(215)
330 NEXT j
340 NEXT i
350 GOTO 350
```

Listing 5. Mit Blockgrafik zum 3D-Effekt

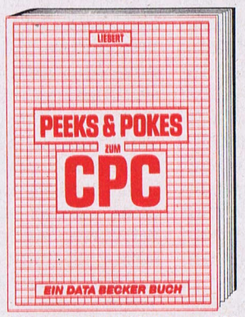
Deutschlands größter Verlag für Computerbücher bringt die richtigen Bücher zum CPC 464, 664 und 6128

Die CPC Bibliothek

Endlich CP/M beherrschen! Von grundsätzlichen Erklärungen zu Speicherung von Zahlen, Schreibschutz oder ASCII-System über Schnittstellen, andere Betriebssysteme und Anpassung von CP/M bis hin zur ausführlichen Behandlung des Schwerpunktes Dateien. Außerdem CP/M-Hilfsprogramme und ihre Anwendung. Für CPC 464, 664, 6128.
CP/M Trainingsbuch zum CPC
ca. 250 Seiten, DM 49,–

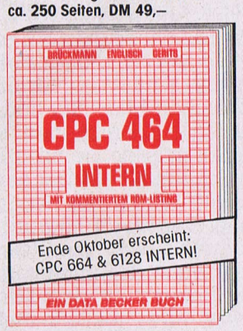


Alles über Diskettenprogrammierung. Mit ausführlichem ROM-Listing (Betriebssystem), einer komfortablen Dateiverwaltung, Disk-Monitor und einem nützlichen Disk-Manager. Dazu verschiedene Programme und Hilfsroutinen, die das Buch für jeden Floppy-Anwender zur Pflichtlektüre machen.
Das Floppy-Buch zum CPC
353 Seiten, DM 49,–



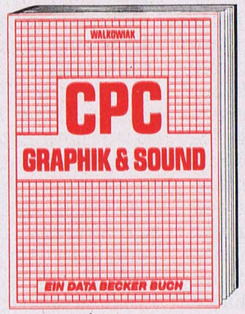
Wichtige Peeks und Pokes zum CPC Kennenlernen. Vom Adreßbereich des Prozessors über Betriebssystem und Interpreter bis hin zur Einführung in Maschinensprache. Präzise Programmierhilfen, sinnvolle Routinen sowie reichlich Material zu Grafikfunktionen, Massenspeicherung und Peripherie, Tricks und Formeln in BASIC, RAM-Pages.
Peeks und Pokes zum CPC
180 Seiten, DM 29,–

Speziell für den Hobbyelektroniker! Von nützlichen Tips zur Platinenherstellung über Adressdecodierung, Adapterkarten und Interfaces bis zu EPROM-Programmierboard und -Programmiernetzteil oder Motorsteuerung für Gleich- und Schrittschaltmotoren werden Erweiterungen praxisnah beschrieben. Am besten gleich anfangen!
CPC Hardware-Erweiterungen
445 Seiten, DM 49,–

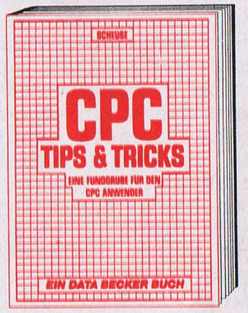


Das Standardwerk zum CPC! Neben dem ausführlich dokumentierten und kommentierten BASIC-ROM-Listing enthält es umfangreiche Kapitel zu Speicheraufteilung, Prozessor, Besonderheiten des Z 80, Gate Array, Video-Controller und Video-Ram, Soundchip, Schnittstellen, Betriebssystem, Routinennutzung, Character-Generator, BASIC-Interpreter und mehr.
CPC 464 INTERN, 548 Seiten, DM 69,–

Von den Grundlagen der Maschinenspracheprogrammierung über die Arbeitsweise des Z 80-Prozessors und einer genauen Beschreibung seiner Befehle bis zur Benutzung von Systemroutinen ist alles mit vielen Beispielen erklärt. Dazu Assembler, Disassembler und Monitor als komplette Anwenderprogramme. Maschinensprache leichtgemacht!
Das Maschinensprachebuch zum CPC
330 Seiten, DM 39,–

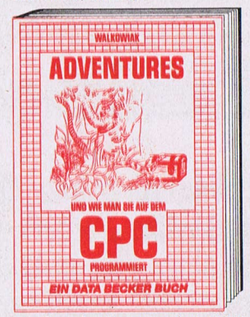


Nutzen Sie Graphik und Sound des CPC! Interessante Beispiele und Hilfsprogramme finden Sie in diesem Buch: Grundlagen der Grafikprogrammierung, Sprites, Shapes und Strings, mehrfarbige Darstellungen, Koordinatentransformation, Verschiebungen, Drehungen, Rotation, 3-D-Funktionsplotter, CAD, Synthesizer, Miniorgel, Hüllkurven u. v. m.
CPC Graphik & Sound
220 Seiten, DM 39,–

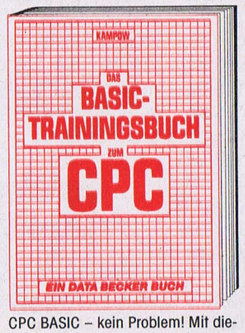


Anregungen und Hilfen zum CPC. Von Hardwareaufbau, Betriebssystem, BASIC-Tokens, Zeichnen mit dem Joystick, Anwendungen der Windowtechnologie und vielen Programmen wie einer umfangreichen Dateiverwaltung, Soundeditor, komfortablem Zeichengenerator bis zu kompletten Listings spannender Spiele bietet das Buch eine Fülle von Möglichkeiten.
CPC Tips & Tricks
263 Seiten, DM 39,–

Lernen macht Spaß! Für Schüler der Mittel- und Oberstufe: Satz des Pythagoras, geometrische Gleichungen, Molekülbildung, Vokabeln, unregelmäßige Verben, exponentielles Wachstum. Grundlagen der EDV, wichtige BASIC-Elemente und eine Einführung in die Grundzüge der Problemanalyse vervollständigen das Ganze.
Das Schulbuch zum CPC, 389 Seiten, DM 49,–



Ein Führer in die phantastische Welt der Abenteuer-Spiele: Adventures wie sie funktionieren, wie man sie spielt und selbst programmiert. Der Clou: neben vielen fertigen Adventures ein kompletter ADVENTURE-GENERATOR.
Adventures – und wie man sie auf dem CPC programmiert
320 Seiten, DM 39,–



CPC BASIC – kein Problem! Mit diesem Trainingsbuch lernen Sie von Grund auf die einzelnen Befehle und ihre Anwendungen, einen richtig sauberen Programmierstil. Von der Problemanalyse über den Flußplan bis zum fertigen Programm. Viele Übungsaufgaben mit Lösungen und Beispielen.
Das BASIC-Trainingsbuch zum CPC 285 Seiten, DM 39,–

Mehr darüber und über weitere Bücher und Programme im neuen DATA BECKER Spezialkatalog Schneider, den wir Ihnen gerne zusenden.

Schon die neue DATA WELT mit dem Supertest des CPC 6128 gelesen?

BESTELL-COUPON
Einsenden an: DATA BECKER · Merowingerstr. 30 · 4000 Düsseldorf 1
Bitte senden Sie mir:
zzgl. DM 5,– Versandkosten
☐ per Nachnahme ☐ Verrechnungsscheck liegt bei
Name und Adresse bitte deutlich schreiben

DATA BECKER
Merowingerstr. 30 · 4000 Düsseldorf · Tel. (0211) 310010

wenn man ein Ausmalen der Fläche fordert. Je größer wir dabei den Radius unseres Kreises wählen, desto mehr Linien müssen wir zeichnen, und desto enger müssen diese Linien nebeneinander liegen. Dies macht das Entwerfen der Grafik zum Geduldspiel, es heißt Kompromisse zu schließen. Wir wollen deshalb eine mittlere Kreisgröße wählen, die eine vernünftige Auflösung bietet, ohne daß man deswegen gleich eine Kaffeepause einlegen muß.

Der Anfang des Programms ähnelt den schon betrachteten. Neu ist der Teil ab Zeile 150. Bei der Kreisdarstellung (Listing 4) können wir nicht wie bei den vorgenannten Darstellungsarten nur mit den Funktionswerten operieren. Da am Ende ein Kreissegment das Verhältnis der entsprechenden Zahl zum Gesamten darstellen soll, müssen wir zunächst den relativen Anteil jeder Zahl am Ganzen berechnen. Dazu bilden wir die Summe über alle Werte und ermitteln mit Hilfe dieser in einer zweiten Schleife den relativen Anteil jedes einzelnen Teils. Da eine Darstellung aller zwölf Angaben aufgrund des geringen Umfanges unseres Kreises eine unbefriedigende Auflösung bringt, fassen wir die relativen Anteile zu Quartalsangaben zusammen. Hierbei dürfen sich die einzelnen Kuchenstücke nicht überdecken, sondern sollen schön nebeneinander auf der Kreisfläche abgebildet werden. Die abgebildeten Werte werden noch mit 2π multipliziert, um den ganzen Kreisumfang abzudecken. Vier FOR...TO...-Schleifen, die den eigentlichen Zeichen-Algorithmus enthalten, laufen über ein Segment des Kreises, und zwar in relativ kleinen Schritten. Zur Berechnung des Punktes auf der Kreislinie wenden wir die Sinus- und Konsinusfunktion an, aus der mit 50 multipliziert der richtige Abstand resultiert. Danach wird der Grafik-Cursor in den Nullpunkt unseres Kreises gesetzt und die entsprechende Linie mit der ersten Farbe gezogen. Dasselbe wiederholt sich für die anderen Kreissegmente.

Zum Abschluß wollen wir noch kurz auf eine weitere Darstellungsart eingehen, den Funktionsgraph. Dieser bietet sich an, wenn man es mit einer Vielzahl von Punkten zu tun hat. Es ist wieder nur eine Variante desselben Themas. In unserem Programm Stabdiagramm (Listing 3) brauchen wir nämlich nur die Zeilen mit dem Befehl DRAWR durch »DRAW 50 * i,h:NEXT i« ersetzen. Statt der Stäbe werden jetzt die einzelnen Funktionspunkte einzeln gesetzt und durch Linien miteinander verbunden. Allerdings ist dies nur eine Methode. Man kann auch die Verbindungslinien weglassen und statt des DRAW PLOT verwenden. Während DRAW eine Linie zieht, setzt PLOT nur einen Punkt. Dies führt zu einer etwas anderen Abbildung. Wenn eine Funktion mit nur wenigen Werten definiert wurde, setzt sich auch der Funktionsgraph sichtbar aus mehreren geraden Teilstücken zusammen, was eventuell etwas störend wirkt. Während hier die Einzelpunktdarstellung Vorteile hat, wird andererseits durch die Linienführung der Zusammenhang der Punkte besser verdeutlicht. Welche Methode nun die bessere ist, hängt von der Anwendung und dem Geschmack des Benutzers ab.

Mit den gerade beschriebenen Grafiken haben wir natürlich erst grundsätzliches Rüstzeug für anspruchsvolle Diagramme kennengelernt. Daher noch ein paar Worte zu weiteren Ergänzungen. Wir sind nämlich nicht auf die Nullinie als Bezugspunkt angewiesen. Wir können den Bezugspunkt beliebig wählen, zum Beispiel das Minimum einer Funktion. Um eine Extremwert-Darstellung zu erhalten, brauchen wir nur ein paar kleine Änderungen in unserem Grundkonzept vorzunehmen. Wir subtrahieren einfach von allen Werten den minimal möglichen, der in der Variablen mi nach dem ersten Durchlauf zur Verfügung steht. Entweder verwendet man dazu eine separate Schleife nach der Extremwert-Berechnung oder wir ändern die Berechnung der einzelnen Werte ab. Bei der Säulengrafik mit DRAW wäre beispielsweise die Zeile 250 mit »h=((m(i)-mi)/(ma-mi))*250« zu ersetzen und schon erscheint die neue Grafik auf dem Schirm.

Auf dem Weg zur Komfort-Grafik

Eine weitere interessante Erweiterung ist das Einfügen von Achsenbezeichnungen in das Diagramm. Die Begrenzungslinien lassen sich ebenfalls mit dem Kommando DRAW zeichnen. Wer noch etwas mehr will, kann die Achseneinteilungen mit kleinen Strichen nachbilden, und diese dann auch noch beschriften. Gut eignet sich dazu das PRINT USING-Kommando in der Exponentialdarstellung. Diese Form des PRINT-Befehls hat den Vorteil, daß wir mit der genau vorgegebenen Länge der ausgegebenen Zeichenkette kalkulieren können. Wollen wir die Benennung der Y-Achse mit sieben Zeichen erlauben, aber in bezug auf die darzustellenden Werte kein Format festlegen, so geht man folgendermaßen vor. Wir nehmen an, wir wollten in unserem Balkendiagramm die verschiedenen Zeilen bezeichnen. Der höchste vorkommende Wert ist ma, der niedrigste mi. Die Darstellung erfolgt in 16 Zeilen Höhe, so daß wir (ma-mi) durch 16 teilen müssen, um den Abstand zwischen den einzelnen Zeilen zu erhalten. Mit einer einfachen Schleife und PRINT USING können wir dann die ganze Benennung auf einmal machen. Die Zeile lautet

```
275 diff=(ma-mi)/16:FOR i=4 TO 20:LOCATE1,i:PRINT USING"# # # .# ↑↑↑↑";mi+(20.5-i)*diff:NEXT i
```

Diese Gleichung gilt natürlich nur, wenn Sie schon die gerade beschriebene Änderung zur Darstellung zwischen Minimum und Maximum in Zeile 250 eingebaut haben. Ansonsten müssen Sie für mi 0 einsetzen. In jedem Fall aber sorgt die neue Zeile 275 für die richtige Benennung. Der Wert 20,5 wurde deshalb gewählt, weil die Reihe von Bildpunkten in der untersten Zeile, unserer Benennung, dem Minimum entspricht. Der Zeilenmittelpunkt dagegen liegt wertmäßig schon 0.5*diff höher.

Wir können noch eine ganze Reihe anderer nützlicher Änderungen, zum Beispiel die gleichzeitige Darstellung von Durchschnittswerten, einbauen. Hier sind Ihrer Phantasie und Ihrem Gestaltungswillen keine Grenzen gesetzt. Komfort und Aussagequalität sind immer verbesserungsfähig.

Vorstoß in neue Dimensionen

Mit diesen Änderungen, sauber ausgeführt, haben wir auch schon die Grenze dessen erreicht, was mit einfacher Grafik machbar ist. Wie gut würde es aber aussehen, wenn wir statt der Rechtecke richtige Balken auf dem Bildschirm hätten. Auch könnten wir noch ein paar zusätzliche Darstellungsmittel gebrauchen, um mehrere Werte gleichzeitig im Zeitablauf darstellen zu können. Dadurch würden Parallelentwicklungen deutlich, wie zum Beispiel die Entwicklung des Bruttosozialproduktes und gleichzeitig die der Inflation. Dies alles ist zwar möglich, nur kommen wir dann nicht mehr mit simplen Grafiken aus. Gefragt ist dann vielmehr der Übergang in die dritte Dimension. Um dies aufzuzeigen, beschränken wir uns auf ein konkret durchgeführtes Beispiel, ein dreidimensionales Balkendiagramm in drei Ebenen. Bei einem normalen Balkendiagramm werden nur die einzelnen Rechtecke perspektivisch gezeichnet. Dies wäre die erste Ebene. Man kann aber in der Perspektive noch eine Stufe weiter gehen und statt einer Reihe von Säulen nebeneinander, auch noch Säulen voreinander zeichnen. In unserem Beispiel operieren wir mit drei Ebenen. Wir können folglich drei unterschiedliche Entwicklungen parallel gegeneinander darstellen, wobei jede Ebene sechs Säulen enthält. Dies ist durch die Parameter festgelegt. Sie sind jedoch nicht auf eine bestimmte Anzahl von Säulen beschränkt. Ab einem gewissen Punkt werden Sie jedoch die Grafik vor lauter Säulen nicht mehr sehen. Als andere Möglichkeit, anstelle der hochauflösenden Grafik, bietet der CPC noch seinen komfortablen Zeichensatz. Daß sich dieser für solche Probleme sehr gut eignet, das sehen Sie, wenn Sie einmal kurz einen Blick auf das nachfolgende kleine Programm (Listing 5) werfen. Obwohl nicht besonders lang, beinhaltet es doch alle gerade beschriebenen Funktionen. Tippen Sie es einmal ein und lassen Sie sich überraschen! Das Ergebnis kann sich sehen lassen. Um das Programm übersichtlich zu halten, wurde bewußt auf jede Art von Benennung, Achsendarstellung, -definition und ähnlicher verzichtet. Sie haben also das reine Rumpfprogramm vor sich, das Sie vor keine Probleme mehr stellen sollte.

Schauen wir uns das Programm etwas näher an. Am Anfang finden wir einen Definitionsteil. Dieser dient zur Eingabe der für die Darstellung benötigter Parameter, die in das Feld w(i,j) eingelesen werden. Der erste Index bezeichnet dabei die Ebene, der zweite die Säule darin. Zunächst wird die Anzahl der Ebenen (1-3) abgefragt und in ze zwischengespeichert. Gleiches passiert danach mit der Anzahl der Elemente je Ebene. Anschließend wird das Bodenfeld gezeichnet (Zeile 170-180), wozu ein String mehrmals hintereinander auszugeben ist. Ab Zeile 190 befinden wir uns dann im eigentlichen Hauptteil. Die einzelnen Balken, die sich aus den Grafikzeichen zusammensetzen, werden hier ausgegeben. Die Erklärung jeder einzelnen Zeile würde zu weit führen. Wollen Sie sich mit dem Ablauf des Programms näher auseinandersetzen, fügen Sie eine Zeitschleife als Zeile 275 ein:

```
275 FOR t=1 TO 1000:NEXT t
```

Die Bearbeitung der Zeilen von 190 bis 280 erfolgt nun im Zeitlupentempo. Sie können jetzt ganz genau verfolgen, wie die Balken aufgebaut werden. Sind diese fertiggestellt, ist zu sehen, daß noch immer einige freie Stellen übrig sind, die von dem Hauptprogramm nicht beschrieben wurden und deshalb nachträglich gesetzt werden müssen. Dies macht eine zweite Schleife, auf die wir bei den Zeilennummern 290 bis 340 stoßen. Sollten Sie sich dafür interessieren, aus welchen Zeichen die einzelnen Balken aufgebaut sind, so schlagen Sie im Handbuch nach. Im Anhang III findet sich der gesamte Standard-Zeichensatz des Computers. Sie können sich die relevanten Zeichen aber auch selbst mit der Funktion CHR$ auf den Bildschirm ausgeben lassen.

```
FOR i=32 TO 255:PRINT i,CHR$(i):NEXT i
```

« heißt der ganze Befehl.

Wenn wir einzelne Zeichen zu Körpern zusammensetzen, eröffnet sich ein weiterer interessanter Bereich der Grafik. Zu Unrecht fällt beim CPC meist die Blockgrafik etwas unter den Tisch und man fliegt nur durch die »höheren Sphären der Hi-Res«. Speziell wenn man nur wenig Platz im Speicher hat und gar nicht auf den optimalen Effekt, sondern nur eine ausreichend genaue Darstellung Wert legt, sollte man sich der Blockgrafik entsinnen. Kombiniert man dazu noch beide Grafikkarten miteinander, überraschen einen oft die Resultate positiv.

Blockgrafiken können, da sie ja im Endeffekt Zeichenketten darstellen, problemlos in Strings gespeichert und sehr schnell auf dem Schirm ausgegeben werden. Man kann die Grafikausgabe für einen viertel Bildschirm in einen einzigen String abspeichern. Der Speicherbedarf ist dabei sehr gering. Die hochauflösende Grafik beansprucht dagegen achtmal soviel Platz. Speziell das Zeichnen relativ komplizierter Figuren, ist mit einer Definition neuer Grafikzeichen (mit SYMBOL) meist viel einfacher zu lösen. Schließlich ist es mit einem Trick noch möglich, auch die normalen Zeichen in Multicolor-Darstellung auf den Schirm zu bringen. Man muß also nicht immer auf die hochauflösende Grafik zurückgreifen.

(Carsten Straush)

Grafik mit Spezialeffekten

Das Schneider-Basic stellt einige Befehle zur Erzeugung von verschiedenen Farben auf dem Bildschirm zur Verfügung. Damit lassen sich viele verblüffende Effekte erreichen, die wir Ihnen hier vorstellen.

Für die Farbdefinition beim CPC gibt es zwei Möglichkeiten. Zum einen kann man mit dem dritten Parameter in den Kommandos DRAW und PLOT eine Farbe festlegen, zum anderen definieren PEN und PAPER Schrift- und Hintergrundfarben. Dabei verwendet der CPC ein etwas ungewöhnliches System. Man schreibt nämlich nicht mit einer Farbe, beziehungsweise einer Farbnummer, sondern mit der Nummer eines Farbregisters. Ändert man nachträglich den Inhalt dieses Registers, so ändert sich mit ihm auch die Farbe auf dem Bildschirm. Probieren Sie einmal die Befehlsfolge »PEN 2:PRINT"Dies ist Farbregister 2"« aus. Sie erhalten den Satz in der Farbe auf dem Bildschirm, die in Farbregister 2 steht. Geben Sie nun »INK 2,21« ein, so erscheint die Ausgabe in grün. Dies ist nun ein sehr nützlicher Effekt. Folgende zwei Beispiele sollen seine Anwendung verdeutlichen.

Beispiel 1: Titelgrafik. Oft, besonders bei Spielprogrammen, gehört eine aufwendige Titelgrafik zum guten Ton. Das geht jedoch in jedem Fall auf Kosten der Zeit. Auf dem Schirm ist dann ein halbfertiges Bild zu sehen, was keinen besonders professionellen Eindruck macht. Die Lösung: man entwirft erst einmal den kompletten Titel. Danach fügt man vor dem Aufbau eine Schleife

ein, die alle 16 Farbregister auf 0 setzt (»FOR i=0 TO 15:INK i,0:NEXT i«). Nach dem Titel fügt man wiederum eine oder mehrere Zeilen ein, die die Farbregister wieder zurücksetzen (zum Beispiel: »INK 1,6:INK 2,21« und so weiter). Der Effekt: da die Titelgrafik schwarz auf schwarz geschrieben wird (Farbregister auf 0), sieht man anfangs nichts! Erst durch die Farbdefinition am Schluß erhält man die gewünschte Farbzusammenstellung. Die Grafik erscheint dann auf einen Schlag. Dieser Trick wird übrigens bei vielen professionellen Spielen für den Schneider benutzt.

Beispiel 2: Laterna magica. Unter diesem lateinischen Ausdruck versteht man einen Effekt, der sich auf die Unfähigkeit des menschlichen Auges bezieht, schnelle Änderungen als Einzelbewegungen zu erkennen. Statt dessen werden die nacheinander liegenden Schritte als laufende Änderung interpretiert. Wenn man dieses Phänomen auf Farbwechsel anwendet, kann man mit dem Weiterschalten von Farben eine Bewegung erzeugen. Wie dies geht, demonstriert ein kleines Beispielprogramm (Listing 1). Definieren wir dazu einen Kreis, der aus sechs gleichen Teilen besteht. Jeder Kreisabschnitt läuft über 1/3 π. Dies ist erst einmal alles, was »Laterna magica« am Anfang tut. Es zeichnet mittels DRAW die sechs Kreissegmente (Zeile 80-140) vorerst schwarz auf schwarz (siehe Zeile 70). Die Auflösung und damit der Grad, mit dem die Flächen abgedeckt werden, wird dabei mit der Variablen »auf« gesteuert. Je größer die Schrittweite, desto schneller ist die Zeichnung fertig, desto größer sind aber auch die nicht ausgefüllten Flächen. Noch eine Anmerkung für Besitzer des CPC 664 und des CPC 6128. Da dieses Verfahren nicht sonderlich schnell ist, sollten Sie mit Ihrem Computer besser einen etwas anderen Weg beschreiten. Ziehen Sie die sechs Begrenzungslinien in der Mitte des Kreises und den Kreisrand mit PLOT. Wenn Sie dann den Cursor mit MOVE in die einzelnen Teilstücke bewegen, können Sie mit »FILL 1« für das erste, »FILL 2« für das zweite und »FILL 3« für das dritte Segment den Schneider anweisen, das Ausfüllen selbst zu machen.

Ab Zeile 160 beginnt die Farbumschaltung. Zunächst bringt man die Umschaltgeschwindigkeit mit der Zeit-Abfrage in Erfahrung, danach wird bei der Weiterschaltung mit jeder der drei »INK-Schleifen« die Farbe um genau eine Position weitergesetzt. Was Farbe 1 war, wird jetzt Farbe 2, Farbe 2 wird Farbe 3 und, um den Kreis komplett zu machen, wird Register 1 mit Farbe 3 geladen. Dieser Prozeß wiederholt sich dreimal. Dann ist der Ausgangszustand wieder erreicht. Zwischen den Umschaltungen sind noch Zeitschleifen eingefügt, um sicherzustellen, daß die Farbwechsel nicht zu schnell vor sich gehen. Ansonsten wären sie nämlich unsichtbar. Wenn Sie jetzt für die Zeit den Wert 50 eingeben, so erhalten Sie ein sich drehendes buntes Rad. Mit einem weiteren Tastendruck kommen Sie zurück in die Zeitabfrage und können das ganze Spiel mit einem neuen Wert noch einmal durchspielen. Besonders interessant wird es, wenn Sie sehr kleine Zahlen eingeben (im Bereich von 1 bis 10). Dann ergeben sich interessante Interferenzen bis hin zu einem reinen Weiß. Der Punkt, an dem Ihr Auge den Farbwechsel nicht mehr unterscheiden kann, wird erreicht und das Auge »schaltet um«.

Das Auge mag Bewegung

Mit dieser Art der Farbumdefinition ist es sehr einfach auch, Bewegungen, wie zum Beispiel einen Wasserfall oder ähnliches, darzustellen. Da das menschliche Auge sehr viel stärker auf Bewegungen als auf stehende Objekte reagiert, kann durch Blinken oder eine simulierte Bewegung mittels Laterna-magica-Effekt auch auf einem voll

```
10 ' ***************
20 ' laterna magica
30 ' ***************
40 auf=0.01122
50 CLS:BORDER 0
60 ORIGIN 320,200
70 FOR i=0 TO 3:INK i,0:NEXT
80 FOR i=0 TO PI/3 STEP auf:MOVE 0,0:DRA
W 50*SIN(i),50*COS(i),1:NEXT
90 FOR i=PI/3 TO 2*PI/3 STEP auf:MOVE 0,
0:DRAW 50*SIN(i),50*COS(i),2:NEXT
100 FOR i=2*PI/3 TO PI STEP auf:MOVE 0,0
:DRAW 50*SIN(i),50*COS(i),3:NEXT
110 FOR i=PI TO 4*PI/3 STEP auf:MOVE 0,0
:DRAW 50*SIN(i),50*COS(i),1:NEXT
120 FOR i=4*PI/3 TO 5*PI/3 STEP auf:MOVE
 0,0:DRAW 50*SIN(i),50*COS(i),2:NEXT
130 FOR i=5*PI/3 TO 2*PI STEP auf:MOVE 0
,0:DRAW 50*SIN(i),50*COS(i),3:NEXT
140 INK 1,24:INK 2,11:INK 3,6
150 WINDOW#2,1,40,23,25
160 INPUT#2,"Zeit";zeit
170 INK 1,24:INK 2,11:INK 3,6
180 FOR t=1 TO zeit:NEXT t
190 INK 1,6:INK 2,24:INK 3,11
200 FOR t=1 TO zeit:NEXT t
210 INK 1,11:INK 2,6:INK 3,24
220 FOR t=1 TO zeit:NEXT t:IF INKEY$=""
THEN 170 ELSE 150
```

Listing 1. Das Programm »Laterna magica« ist Grundlage vieler grafischer Effekte

```
10 ' ************
20 ' Eingabedemo
30 ' ************
40 SPEED INK 40,5
50 FOR i= 1 TO 4
60 CLS
70 INK 2,6,24
80 PAPER#2,2
90 PRINT"              Eingabe Person Nr.";i

100 LOCATE 3,5:PRINT CHR$(24)+" Name   :
"+CHR$(24):LOCATE 3,7:PRINT CHR$(24)+" V
orname:"+CHR$(24)
110 LOCATE 3,10:PRINT CHR$(24)+" Strasse
:"+CHR$(24)+"                  "+CHR$(24
)+" Nr.:"+CHR$(24)
120 WINDOW#2,13,40,5,5:CLS#2:INPUT#2,"",
n$(i,1)
130 PAPER#2,0:CLS#2:PRINT#2,n$(i,1)
140 PAPER#2,2:WINDOW#2,13,40,7,7:CLS#2:I
NPUT#2,"",n$(i,2)
150 PAPER#2,0:CLS#2:PRINT#2,n$(i,2)
160 PAPER#2,2:WINDOW#2,13,28,10,10:CLS#2
:INPUT#2,"",n$(i,3)
170 PAPER#2,0:CLS#2:PRINT#2,n$(i,3)
180 PAPER#2,2:WINDOW#2,36,40,10,10:CLS#2
:INPUT#2,"",n$(i,4)
190 PAPER#2,0:CLS#2:PRINT#2,n$(i,4)
200 NEXT i
```

Listing 2. So einfach kann die Eingabe-Routine einer Dateiverwaltung gehalten werden

Die **DATEN RIESEN** für den Schneider CPC 464/664 sind da:

CPC-COMPUTER DICTIONARY 464/664

(Computer-»Wörterbücher« für den SCHNEIDER)

★ Über **20000 (Zwanzigtausend)** gespeicherte Vokabeln
★ Über **10000 (Zehntausend)** Stichwörter
★ **Minimale Zugriffszeiten**
★ Kein langes Suchen mehr im oder nach dem Wörterbuch
★ Individuell **ERWEITERBAR** durch eigene Angaben
★ Volle Ausnutzung der CPC u. Floppy-Kapazität
★ Eine große Hilfe in **Schule** und **Beruf**

DEUTSCH/ENGLISCH auf 3-Zoll-Diskette	DM 69,90
ENGLISCH/DEUTSCH auf 3-Zoll-Diskette	DM 69,90
DEUTSCH/ITALIENISCH auf 3-Zoll-Diskette	DM 69,90

(Versionen auf 5¼-Zoll-Disketten sind in Arbeit)

ERHÄLTLICH SIND »CPC DICTIONARY 464/664« IM GUTEN FACHHANDEL
oder direkt bei:

ZS-Soft Microtrading

Peter Herzog, Postfach 2361 — Doktorberg 20
8240 Berchtesgaden
Tel. 08652/61199-2691

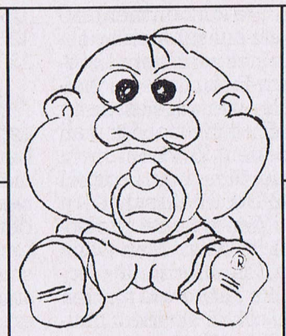
Bitte fordern Sie unseren **kostenlosen** Gesamtkatalog an.
ES LOHNT SICH!!!

HÄNDLERANFRAGEN ERWÜNSCHT

GREMLINS (K) DM 40,—

AMERICAN FOOTBALL (K) DM 47,-

FRANK BRUNO'S BOXING (K) DM 33,-



BRUCE LEE (K) DM 40,-

MULTITEXT (K) DM 70,-

EXPLOA-DING FIST (K) DM 36,-

COMPUTER-SHOP
Landsberger Str. 104
8000 München 2
Tel. 5022463

MULTIDATEI (K) DM 63,-

WORLD SERIES BASEBALL (K) DM 24,-

Lieferung per Nachnahme oder per Scheck
zzgl. DM 2,50 Versandkosten

MULTI-ADRESS (K) DM 63,-

HOTLINE 0211 — 6801403

Joysoft
4000 Düsseldorf 1 · Humboldtstraße 84

ABACOMP
Wieder neue Preis-Hits
Sonderpreise solange Vorrat
Bestellungen bitte nur schriftlich an: **ABACOMP GmbH,** Kransberger Weg 24, 6000 Frankfurt/M. 50
Tel. Auskunft Mo-Sa 8-9.30 Uhr unter 069/700308
Ladenöffnung Mo-Fr. 10-12 u. 14-18 Uhr. Ginnheimer Landstr. 1

CPC 464 + CPC 664
sowie Zubehör in unserem Angebot
bitte erfragen Sie die aktuellen Preise

Schneider CPC 464/664
Riesenangebot

Über 400 Artikel lieferbar	
z.B. CP/M Software:	
Turbo Pascal 3.0 auf 3 Zoll	**220 DM**
Turbo Pascal m. Grafik	**280 DM**
Wordstar m. Mailmerge	**199 DM**
dBase II	**199 DM**
Multiplan	**199 DM**
für dBase II und Multiplan nötig:	
SP 64 von Vortex	**275 DM**
Vortex-Disk-Laufwerke:	
F 1-S	**1198 DM**
F 1-D	**1698 DM**
F 1-Z	**698 DM**
weitere Artikel von Vortex lieferbar.	
dk'tronics Light pen	**99 DM**
DFÜ Paket inkl.	**450 DM**
Akustikkoppler m. FTZ-Nummer Netzteil und Teleterminal 300S	

Liste gegen 2,50 DM in Briefm.

H.-J. Janke
Tel. 0202/475521, Postfach 150173, 5600 Wuppertal 12

gefüllten Bildschirm noch Aufmerksamkeit für eine neue Information erregt werden. Sie können mehrere Objekte schwarz auf schwarz zeichnen und dann eines nach dem anderen durch Farb-Definition erscheinen lassen. Oder wie wäre es mit einer Codewort-Eingabe, bei der man die eingegebenen Zeichen nicht mehr sieht, wie dies zum Beispiel bei einigen Out-Door-Bankschaltern (Geld aus der Wand) heute professionell angewandt wird. Man wählt einfach für die Schriftfarbe des Codewerts dieselbe Farbe wie für den Hintergrund — schon sieht ein neugieriger Betrachter nichts mehr.

Wir können das Spiel mit Farbe und Bewegung auch noch in einem etwas anderen Bereich einsetzen, nämlich bei sogenannten Laufschriften. Im folgenden Programm wollen wir eine einfache, möglichst große Laufschrift erzeugen. Als Text haben wir dafür »Schneider CPC 464« gewählt. Dieser ist als String in der Variablen »z$« in Zeile 70 gespeichert. Um am Anfang und am Ende die erwünschte Anzahl von Leerzeichen zu erhalten, werden an diesen String in Zeile 80 mit dem Kommando SPACE$ einige davon definiert. Die nachfolgenden Zeilen (von 90 bis 120) bilden eine Schleife, in der der gerade erst zusammengebaute String leicht wieder »zerpflückt« wird. Mit dem MID$-Kommando trennen wir aus dem String, beginnend mit dem durch I vorgegebenen Zeichen, einen Teil-String mit der Länge 19 heraus. Da I von 1 bis 17 läuft, enthält unser Teil-String mit zunehmendem I immer weniger Leerstellen und einen immer größeren Teil unseres Ausgabe-Strings. Dieser wird durch das LOCATE-Kommando in Zeile 100 immer an derselben Stelle ausgegeben. Die später gedruckten Strings überdecken damit die früheren. Dies erzeugt den Eindruck der laufenden Schrift. Zeile 115 verzögert diesen Prozeß durch eine Zeitschleife, um eine gute Lesbarkeit der Schrift sicherzustellen. Wenn Ihnen der Aufbau zu schnell oder zu langsam ist, können sie problemlos den Schleifenendwert ändern.

```
10  REM*****************
20  REM** Laufschrift **
30  REM*****************
40  INK 0,0:INK 1,2
50  BORDER 0:PAPER 0:PEN 1
60  MODE 0
70  Z$='SCHNEIDER CPC 464"
80  Z$=SPACE$(19)+Z$+SPAC-
    E$(25)
90  FOR I= 1 TO 19
100 LOCATE 1,12
110 PRINT MID$(Z$,I,19);
115 FOR T=1 TO 150:NEXT T
120 NEXT I
130 GOTO 90
```

Das nächste Programm (»Laufschrift 2«) stellt eine Variante desselben Themas dar. Wir geben uns nicht mehr damit zufrieden, nur einen wandernden String auszugeben, der eine einfarbige Laufschrift erzeugt. Unser Ziel ist es, diese Laufschrift mehrfarbig zu gestalten. Basis ist dabei das schon bekannte Programm, das wir allerdings in einigen wichtigen Punkten erweitern müssen. Wir geben den mit MID$ erzeugten String jetzt nicht mehr direkt aus, sondern speichern ihn in der Variable Z2$. In Zeile 110 und 120 wird dieser Teil-String jetzt noch einmal Buchstabe für Buchstabe »auseinandergenommen«. Jeder Buchstabe erhält dabei eine andere Farbe. Dies erreichen wir, indem wir auf unsere Zählvariable wieder die bekannte Modulo-Funktion anwenden. Nehmen wir J modulo 5 und addieren dazu eins, erhalten wir für unser Farbregister Werte zwischen eins und fünf. Diese benutzen wir, um mit dem nachfolgenden PEN-Kommando die Schriftfarbe zu bestimmen. So erhält man ein Schriftband, in dem sich Buchstabe für Buchstabe die Farbe ändert.

```
10  REM*****************
20  REM**Laufschrift 2**
30  REM*****************
40  INK 0,0:INK 1,2:INK 2,6:INK
    3,21:INK 4,7:INK 5,24
50  BORDER 0:PAPER 0:PEN 1
60  MODE 0
70  Z$="C P C 4 6 4"
80  Z$=SPACE$(19)+Z$+SPAC-
    E$(25)
90  FOR I=1 TO LEN(Z$)—21
100 Z2$=MID$(Z$,I,20):LOCATE
    1,12
110 FOR J=1 TO 19
115 FOR T = 1 TO 150:NEXT T
120 m=(J MOD 5)+1:PEN m:
    PRINT MID$(Z2$,i,1);:NEXT J
130 NEXT I
140 GOTO 90
```

Im Spiel von Licht und Farben stellt auch das Blinken eine weitere hübsche Variante dar. Beim Schneider ist der Farbwechsel ja relativ problemlos mit dem INK-Kommando machbar. Dazu definiert man einfach ein Farbregister auf zwei Farben, den Rest erledigt der CPC. Wozu ist neben Spieleffekten ein Farbwechsel nun überhaupt nützlich? Eine sinnvolle Anwendung ist beispielsweise die Hervorhebung eines Eingabefeldes im Rahmen einer Bildschirmmaske und damit die Lösung eines Problems bei der Datenverwaltung. Erwartet diese eine größere Anzahl von Eingaben pro Datensatz, so hat man zwei Möglichkeiten. Entweder Sie beschränken sich auf die laufende Darstellung auf nur wenige Werte. Dann haben Sie zwar Übersicht über Ihren Datenbestand, Sie wissen aber nicht mehr, was Sie zuvor eingegeben haben. Denn diese Information ist ja schon wieder gelöscht. Sie können natürlich auch alle benötigten Daten neben- oder untereinander auf dem Bildschirm darstellen. Dann sind zwar alle Informationen sichtbar, aber Sie wissen nicht, was aktuell an der Reihe ist. Eine brauchbare Lösung ist also vonnöten, um gleichzeitig alle Informationen parat zu haben und dabei trotzdem nicht den Überblick zu verlieren. Wie dies mit Hilfe von blinkenden Eingabefeldern machbar ist, soll anhand eines Beispiels gezeigt werden. Im Listing 2 ist der Eingabeteil einer Dateiverwaltung definiert. Es liest vier Personendaten in das Feld a$(i,j). In einer Schleife gibt der erste Index dabei die Person wieder.

Zuerst wird die Eingabemaske erstellt (Zeile 90-110). Dann kommt der für uns interessante Teil, die Benutzerführung mittels blinkender Eingabefelder. Definieren wir ein Window (»WINDOW#2«) auf die gewünschte Größe unseres Eingabefeldes. Dabei wird auch die Hintergrundfarbe jedesmal neu festgelegt und zwar auf Register 2. Nach dem Löschen mit CLS blinkt dann das gewünschte Eingabefeld. Der Benutzer weiß damit genau, wo er Daten eintragen soll. Ist die Eingabe abgeschlossen und mit ENTER quittiert, so wird der gerade eingelesene String (a$(i,j)) noch einmal im Eingabefenster ausgegeben, jetzt aber mit einer anderen Hintergrundfarbe. Der Hintergrund des Windows wurde nämlich davor auf schwarz (»INK 0«) zurückgesetzt. Dieser Vorgang wiederholt sich für alle weiteren Eingaben. Dieses kurze Programm erhebt nun nicht etwa den Anspruch, eine komplette Dateiverwaltung zu sein. Es soll nur das grundsätzliche Vorgehen, die Definition von blinkenden Eingabefeldern demonstrieren. Sie können dieses Fragment aber problemlos auf eigene Eingaben erweitern. Wenn Sie dann noch eine gute Editierroutine einfügen, haben Sie bereits ein gutes Datenverwaltungsprogramm. (Carsten Straush)

»Geheimcodes« geknackt

Neben den Zeichen für Buchstaben, Zahlen und Sonderzeichen gibt es beim Schneider noch sehr nützliche Systemsteuerzeichen. Sie stellen eine Kurzform längerer Anweisungen — meist zur Bildschirmgestaltung — dar. Mit ihnen kann man sich viel Programmierarbeit sparen und den optischen Eindruck seiner Programme verbessern.

Die Systemsteuerzeichen sind eine Kurzform der Aufrufe bestimmter Betriebssystem-Routinen. Sie ersetzen damit einige der umständlichen CALL-Befehle, mit denen man sonst auf das Betriebssystem zurückgreift. Im wesentlichen beziehen sich alle Steuerzeichen auf die Bildschirmausgabe. Deshalb werden sie auch einfach mit PRINT angesprochen.

Stößt nun der Computer in einem Ausgabetext auf ein solches Zeichen, so kann er dies nicht drucken und ruft deshalb die entsprechende Routine des Betriebssystems auf. Geben Sie beispielsweise »PRINT "<Cursor runter>" « ein, so geht der Cursor eine ganze Zeile nach unten. »<Cursor runter>« entspricht CTRL-Taste und gleichzeitigem Druck auf »J«. Auf dem Bildschirm ist dann ein Pfeil nach unten zu sehen. Eine nachfolgende Ausgabe erfolgt mit diesem Befehl also eine Zeile niedriger. Die meisten der Steuerzeichen sind aber über Tastatur direkt verfügbar und zwar in der Control-Ebene.

Ein Beispiel: Drücken der Taste »G« bringt ein kleines »g« auf den Bildschirm. Drücken der Shift-Taste und »G« erzeugt ein großes »G«. Drücken der CTRL-Taste und »G« bringt dagegen ein Glockensymbol auf dem Bildschirm.

Dieses Zeichen ist für die Systemsteuerzeichen untypisch, da es mit der Bildschirmgestaltung nichts zu tun hat. Es sorgt vielmehr dafür, daß der CPC einen kurzen Piepser erklingen läßt.

Da es sich bei den Steuercodes letztendlich auch um Zeichen handelt, können wir die damit verbundenen Funktionen nicht nur mit dem direkten PRINT-Befehl, sondern auch über den Umweg mit CHR$ erreichen. CHR$(7) entspricht dabei unserem »CTRL G«. Im folgenden bezeichnet »CTRL G« immer die Eingabe eines Wertes mit gedrückter CTRL-Taste. »PRINT CHR$(7)« erzeugt nun denselben Effekt. Dieser läßt sich entweder als Warnton gut verwenden, aber beispielsweise auch, um auf die Ausgabe eines Textes aufmerksam zu machen. Tippen wir »PRINT CHR$(7)+"NAME" «, so wird das Wort »Name« ausgegeben und gleichzeitig ertönt ein Piepser aus dem Lautsprecher. Diesen Effekt hätten wir auch mit »PRINT" <CTRL G>NAME'« erzielen können.

Zeichen mit CTRL

Die Zahl in der CHR$-Funktion ist dabei mit der Nummer des Buchstabens in der Control-Funktion im Alphabet identisch. »G« ist der siebte Buchstabe im Alphabet und somit liefert CHR$(7) den entsprechenden Code. Analog dazu können wir das Drücken von »CTRL F« durch ein CHR$(6) ersetzen und so weiter. Jedoch ist dabei zu beachten, daß »CTRL C« für die Character-Codes 2 und 3 gleichzeitig gilt. Somit ist dann »CTRL A« dem CHR$(0) und »CTRL B« dem CHR$(1) äquivalent.

Doch schauen wir uns nun die anderen Systemsteuerzeichen an. Beim Ausprobieren und Kennenlernen sollten Sie dabei möglichst kein Programm im Speicher haben, da der CPC sich bei mehrmaligen Fehlbedienungen gelegentlich »aufhängt«, das heißt auf keine Taste mehr reagiert. Ein im Speicher befindliches Programm ist dann verloren.

Insgesamt gibt es 32 dieser Systemsteuerzeichen. Einige sind von geringerem Interesse, da sie nur ein Äquivalent zu einfach anzuwendenden Basic-Kommandos sind. Es gibt jedoch auch eine Reihe von Codes, die ungeahnte Effekte auslösen. Diese wollen wir nun untersuchen. Beginnen wir mit dem niedrigsten für den Benutzer erlaubten Kommando. Code 4 hat denselben Effekt, wie das MODE-Kommando. Mit einem zweiten Character geben wir dabei den gewünschten Modus an. Wenn bei einem Systemsteuerzeichen noch Parameter benötigt werden, so müssen diese natürlich auch mit der CHR$-Funktion geliefert werden. Es muß dabei das Zeichen übergeben werden, das den ASCII-Code hat, der mit der zu übergebenden Zahl übereinstimmt. Soll der CPC also den Wert 65 erkennen, so muß als zweite Angabe CHR$(65), (ein großes »A«) mitgegeben werden. Bei CHR$(4) gibt es jedoch auch eine »Default«-Bedingung (Voreinstellung). Wird kein Parameter spezifiziert, so setzt der Computer selbst den Wert 1 ein. »PRINT CHR$(4)«, »PRINT "<CTRL D>" « und »MODE 1« machen daher dasselbe, genau wie »PRINT CHR$(4)+CHR$ (1)« oder »PRINT "<CTRL D>" «. »PRINT CHR$(4)+CHR$(0)«, beziehungsweise »PRINT CHR$(4) +CHR$(2)« sorgen für die Umschaltung in den Modus 0, beziehungsweise den Modus 2.

Mit CHR$(5) kommen wir zu einem Zeichen, das schon etwas mehr leistet. Wie Sie sicher wissen, ist es mit dem CPC möglich, die Zeichenausgabe durch den Grafik-Cursor zu dirigieren. Dies wird mit dem TAG-Befehl gemacht. Wenn man nach »TAG« ein PRINT-Kommando ausführt, so wird die dadurch definierte Zeichenkette nicht wie gewohnt an der Position des Text-Cursor gedruckt, sondern der Grafik-Cursor gibt an, wo das Zeichen dargestellt werden soll. Maßgeblich ist dabei die linke, obere Ecke. Dieser Punkt ist mit der Eckposition des auszugebenden Symbols identisch. Mit CHR$(5) können wir diese Art der Ausgabe nun für ein Zeichen erheblich einfacher darstellen. Wenn wir CHR$(5)+CHR$(65) ausdrucken, so wird ein großes »A« an der Stelle auf dem Schirm ausgedruckt, an der sich der Grafik-Cursor gerade befindet.

CHR$(7), den Steuerbefehl für einen Piepser, haben wir schon kennengelernt. Wir kommen nun zu Steuerzeichen, die den ehemaligen Commodore-Benutzern sehr bekannt sein dürften, den Cursorsteuerzeichen.

»PRINT CHR$(8)« oder »PRINT "<CTRL H>'« bewegt den Cursor um eine Stelle nach links, CHRS(9) oder <CTRL J> nach rechts. Geben wir »PRINT CHR$(10)« oder »PRINT "<CTRL J>« ein, so geht der Cursor eine Zeile nach unten; mit CHR$(11) oder <CTRL K> geht

es in die Gegenrichtung. Die letzten beiden Paare bewegen aber dabei nicht nur den Cursor. Befindet sich der Cursor auf der letzten Zeile, das heißt in Zeile 25, so wird mit CHR$(10) auch der Bildschirm um eine Zeile nach oben geschoben. Diesen Effekt kennen wir schon vom einfachen Beschreiben der letzten Zeile. Interessant ist jedoch, daß wir den Bildschirm mit CHR$(11) in der obersten Zeile zum Scrollen in der Gegenrichtung, das heißt nach unten, bewegen können. Dies ist sehr nützlich, wenn wir Texte nach unten oder oben rollen wollen, wobei jeweils eine neue Textzeile dazu kommen soll. Eine nützliche Anwendung für Kommando bilden beispielsweise Textverarbeitungsprogramme.

Einen ganz anderen Effekt hat »PRINT CHR$(12)« oder »PRINT "<CTRL L>"«. Bei Eingabe dieses Codes führt der CPC ein Clear-Home aus. Das heißt, er löscht den Bildschirm und setzt den Cursor auf den Anfang der obersten Bildschirmzeile. Dieser Befehl stellt also das Gegenstück zu dem Basic-Befehl CLS dar.

Cursor-Bewegungen

»PRINT CHRS(13)« oder »PRINT "<CTRL M>" bewegen den Cursor an den linken Rand der aktuellen Zeile. Damit sind sehr schnelle Cursorbewegungen möglich. Auch das Überschreiben von Texten kann auf diese Weise geschehen, da kein Zeilenvorschub erfolgt. Wir befinden uns also nach der Ausführung des Befehls wieder am Anfang der Zeile, in der wir gerade geschrieben haben. Probieren Sie einmal »PRINT "ab";chr$(13);"abcd"« aus. Sie erhalten nur die Ausgabe »abcd«, da die zuerst geschriebenen zwei Buchstaben durch die Rücksetzung des Cursors überschrieben wurden. Mit CHR$(13) ist es also zum Beispiel in einem Textverarbeitungsprogramm leicht möglich, die aktuelle Zeile nach einer Änderung ohne LOCATE zu überschreiben.

Wenn man die Cursorsteuerzeichen einzeln benutzt, kann man schon interessante Effekte erzielen. Ihre ganze Stärke zeigen diese Kürzel jedoch erst in der Kombination. Geben Sie zum Beispiel einmal »PRINT CHR$(12)+"HIER ANFANG"+STRING$(CHR$(10),5)+"5 ZEILEN RUNTER"
+CHR$(10)+STRING$(CHR$(8),3)
+"DANN ZURÜCK"+CHR$(10)+CHR$(13)+"UND AN DEN ANFANG DER NAECHSTEN ZEILE"« ein. Mit relativ wenigen Befehlen haben Sie so eine schon sehr komplexe Bildschirmausgabe gemacht.

Ein besonders wertvoller Vorteil dieser Art der Steuerung liegt darin, daß die Steuercodes allgemein wie ein normales Zeichen behandelt werden. Wir können also eine grafische Ausgabe mit den zugehörigen Kontrollsequenzen in einem String speichern, Teile aus Strings herauslösen und neu zusammensetzen. Man hat damit sehr effektive Werkzeuge, Grafiken und Texte zusammenzubauen und auf den Bildschirm zu bringen. Auch können mehrmalig hintereinander auftauchende Steuercodes abgekürzt mit Hilfe von STRING$ eingegeben werden. In dem letzten Beispiel haben wir dies gemacht, um die »fünf Zeilen runtergehen« mit Hilfe eines einzigen Befehls zu erzeugen. Wir können jetzt also eine Zeichenkette mit Hilfe von Steuercodes leicht positionieren und so Bewegungen über den ganzen Bildschirm programmieren. Damit haben wir die Grenzen der Codes jedoch noch lange nicht erreicht.

Als nächstes stehen Strings auf dem Programm, die ihre Farbe selbständig ändern. Dies läßt sich mit den nächsten beiden CHR$-Nummern (14 und 15) erreichen. Sie stellen das Äquivalent zu den Basic-Befehlen PAPER und PEN dar. »PRINT CHR$(14) oder »PRINT "<CTRL N>"« beziehungsweise »PRINT CHR$(15)« oder »PRINT "<CTRL O>"« sind mit den Befehlen »PAPER 1« und »PEN 1« identisch. Durch Anhängen eines zweiten Characters erhalten wir die anderen Farbregister.

»PRINT CHR$(15)+CHR$(2)« führt also das gleiche wie »PEN 2« aus. Farbumschaltungen mit in den Text zu integrieren ist nur eine Anwendung für diese Steuercodes. Ein wenig Umdenken in bezug auf die gewohnte Art der Farbwahl ist aber nötig. Geben Sie doch einmal die nachfolgenden Befehle ein: »a$="abcd"+CHR$(15)+CHR$(2)+"efgh"+CHR$(15)+CHR$(3)+"ijkl": PRINT a$«.

Sie sollten im Modus 1 in der Ausgangsfarbzusammenstellung ausprobiert werden. Gegebenenfalls muß der Computer in den Einschaltzustand zurückgesetzt werden. Damit sind wir nun in der Lage, Farbänderungen in eine Zeichenkette einzubauen. Sogar ein blinkender String ist machbar. Dazu definieren Sie einfach INK 3,11,6. Geben Sie nun »PRINT CHR$(14)+CHR$(3)+"Warnmeldung"+CHR$(14)+CHR$(0)« ein, so erhalten Sie eine effektvolle Warnung. Das zweite CHR$-Kommando dient dabei übrigens dazu, die Untergrundfarbe für die normale Ausgabe wieder auf Dunkelblau zurückzuschalten. Unterläßt man dies, so blinkt nach der Anweisung der ganze Schirm. Besonders im Modus 0 ergeben mehrere Farbumschaltungen nacheinander in demselben String farbenfrohe Texte. Damit kann beispielsweise in einer Programmüberschrift jeder Buchstabe einzeln mit einer anderen Farbe dargestellt werden.

Zeichen-Bandwurm

Auf einen weiteren wichtigen Punkt sollte in diesem Zusammenhang noch hingewiesen werden. Er betrifft den Speicherbedarf. Obwohl die Definition mit CHR$ einen ziemlich langen »Zeichen-Bandwurm« ergibt, werden in dem String nur wenige Zeichen durch die Kontrollsequenzen belegt. Da nämlich alle Steuercodes als normale Zeichen betrachtet werden, benötigen sie auch jeweils nur den Platz eines Zeichens. Dazu muß nur noch der Platz für die Parameter addiert werden. Damit ist das Laden solcher einmal definierter Strings mit Sicherheit die schnellste und speicherplatzsparendste Methode, um den Bildschirm schnell zu füllen.

Noch interessanter als die Farbsteuerbefehle sind die Codes »PRINT CHR$(19)« oder »PRINT "<CTRL S>"« sowie »PRINT CHR$(20)« (»PRINT "<CTRL T>"«).

Mit diesen kann man, in Abhängigkeit von der Position des Cursors, einen Teil des Bildschirms löschen. Mit CHR$(19) löschen wir dabei den Bildschirm über dem Cursor, mit CHR$(20) den unter der aktuellen Cursorposition. Anfangs- beziehungsweise Endpunkt ist immer die aktuelle Cursorposition. Dies ist sehr nützlich, wenn man nur Teile des Bildschirms löschen will.

Der Punkt, an dem sich der Cursor momentan befindet, wird dabei als letzter Punkt mitgelöscht. Wird zum Beispiel ein Text in den Bildschirmzeilen 1 bis 10 nicht mehr benötigt, so setzt man den Cursor mit LOCATE auf das Ende der zehnten Zeile, das heißt je nach eingeschaltetem Modus in Spalte 20, 40 oder 80 und führt dann ein PRINT "<CTRL S>" aus. Damit wird dann der obere Teil des Bildschirms gelöscht. Man braucht also nicht (wie es mit Basic-Befehlen nötig wäre) erst ein Window auf diesen Bereich zu definieren und dieses — gegebenenfalls nach einer erneuten Definition der Hintergrundfarbe — mit PAPER und CLS zu löschen.

Sie können natürlich auch hier wieder kombinieren. Wenn Sie zum Beispiel vor der Ausführung dieser Befehle mit CHR$(14) die Hintergrundfarbe umdefinieren, wird mit dieser neuen Farbe gelöscht, das heißt ein Teilbereich des Bildschirms wird farbig. Und dann kann man natürlich auch diese Kommandos in Strings einbauen. Mit den Steuercodes 17 und 18 wird übrigens das gleiche, allerdings nur in einer Zeile, gemacht.

Die vier nächsten Codes überspringen wir vorerst. Sie werden weiter hinten besprochen. Mit Steuerzeichen Nummer 25 ist es möglich, das SYMBOL-Kommando nachzubilden. Dazu müssen aber mit CHR$(25) nun weitere Angaben an den Computer übergeben werden. Einmal die Nummer des zu definierenden Zeichens und natürlich die Werte für die acht Punktzeilen, aus denen jedes Zeichen besteht. Der erste Wert definiert dabei, wie vom Basic-Befehl gewohnt, die oberste Pixelreihe, der zweite die nächstfolgende und so weiter.

Farben blinken immer

Ähnlich wirkt auch CHR$(26). Mit diesem Code ersetzen wir das Window-Kommando. Die mit zu übergebenden vier Werte stehen für die Begrenzungen des Bildschirmfensters. Mit CHR$(28) und CHR$(29) werden INK und BORDER imitiert. Dabei ist zu beachten, daß die Farben beim Schneider grundsätzlich blinkend definiert sind. Wenn Sie also zum Beispiel »INK 2,2« eingeben und den Hintergrund auf diese Farbe gesetzt haben (»PAPER 2«), so stellt der Computer nicht einen blauen Hintergrund dar, sondern er wechselt ständig zwischen zwei Farben ab. Da diese aber beide auf Blau definiert sind, fällt es nicht auf. »INK 2,2« wird also vom CPC als »INK 2,2,2« angesehen. Definieren wir dagegen zwei verschiedene Farben, so blinkt die Darstellung, wobei wir mit »SPEED INK« die Geschwindigkeit beeinflussen. Bei den Steuercodes 28 und 29 müssen immer zwei Farben mit übergeben werden, damit die Ausgabe fehlerfrei ist.

Neben WINDOW und SYMBOL werden aber auch noch andere Basic-Kommandos entbehrlich. Mit »PRINT CHR$(30)« oder »PRINT "<Ctrl ↑>"« wird ein »Cursor-Home« ausgeführt; der Cursor wird also in die oberste, linke Bildecke gesetzt. Im Gegensatz zu »PRINT CHR$(12)« wird dabei der Bildschirm allerdings nicht gelöscht. Mit diesem Kommando kommt man mit dem Cursor schnell in den oberen Teil des Bildschirms. Soll dieser dagegen in die Mitte, so hilft »PRINT CHR$(31)« mit den zugehörigen Parametern. Dieser Befehl entspricht dem LOCATE-Kommando. In Basic benötigt LOCATE zwei Angaben — die Cursorspalte und die -zeile. Wir müssen also zwei Parameter für Zeilen- und Spaltennummer eingeben. Unterlassen wir dies, so sind die Werte 1 und 12 voreingestellt. Damit wird der Cursor automatisch auf den Beginn der Zeile 12 gesetzt. Die weitere Steuerung des Cursors kann mit den einfachen Cursorsteuer-Befehlen (CHR$(8) bis CHR$(11)) problemlos erfolgen.

Nun sind wir am Ende der Steuerbefehls-Liste angelangt. Die »Creme de la Creme« haben wir aber zwischendurch — übrigens ganz bewußt — ausgelassen. Die Steuercodes 21 bis 24 gehören in den Bereich der Farbänderungen. Es werden aber nicht einfach Farben definiert, sondern es geht eher um Spezialeffekte. Die Kommandos sind sehr weitreichend, so daß es sinnvoll ist, zuerst noch ein wenig Übung mit der Anwendung dieser Steuercodes zu erlangen.

Mit »PRINT CHR$(24)« oder »PRINT "<Ctrl X>"« erreichen wir eine reverse Darstellung. Der Befehl vertauscht die Schrift- mit der Hintergrundfarbe. Wenn wir also an irgendeiner Stelle eine Zeichenkette besonders hervorheben wollen, so ist dies mit »PRINT CHR$(24)+"TEXT"+CHR$(24)« sehr einfach. Sie dürfen aber nicht die Rückschaltung (mit dem zweiten Befehl) vergessen, sonst bleibt Ihnen die Revers-Darstellung auch für den Rest des Bildschirms erhalten. Dieser Steuercode arbeitet unabhängig von irgendwelchen Windows. Es ist also egal, wo man die gerade beschriebene Zeichenfolge ausgibt, oder wie groß die einzelnen Fenster sind. In jedem Fall wird die Schriftfarbe des durch PRINT angesprochenen Bereichs mit der zugehörigen Hintergrundfarbe vertauscht. Das Umdefinieren von PAPER und PEN entfällt also.

Mit CHR$(23) wird der Modus des Grafikfarbstiftes gesetzt. Wenn Sie normalerweise mit DRAW eine Linie ziehen, so wird dabei die Vordergrundfarbe neu auf die einzelnen Bildpunkte gesetzt. Diese Darstellung ist unabhängig von eventuellen vorherigen Befehlsfolgen. Mit »PRINT CHR$(23)+CHR$(1)« wird die Ausgabe von Bedingungen abhängig gemacht. Betrachten wir zunächst den einfachsten Fall, nämlich daß Sie mit der Farbe 1 auf der Farbe 0 schreiben (»PEN 1«, »PAPER 0«). War an der Stelle, die Sie mit DRAW oder PLOT jetzt beschreiben wollen bereits ein Punkt gesetzt, so wird er zurückgesetzt. War der Punkt dagegen mit der Hintergrundfarbe belegt, so wird jetzt die Vordergrundfarbe gemalt. Schwieriger ist es, wenn man mit mehreren Farben arbeitet. Dann tritt bei gesetzten Punkten nämlich ein Farbwechsel auf. Diese Farbänderungen entstehen dadurch, daß der CPC die vorhandenen Punktwerte mit den neuen mit XOR verknüpft. Die XOR-Funktion setzt dann einen Punkt, wenn die Werte der beiden Angaben unterschiedlich sind. Bei identischen Angaben wird in jedem Fall der Punkt gelöscht. Dieser Zusammenhang gilt zumindest für Modus 2. Eine weitere Farbkombination erhält man mit CHR$(23)+CHR$(2). Hier wird die AND-Funktion auf die einzelnen Punkte angewandt. Ein Punkt wird dabei nur gesetzt, wenn er schon vorhanden ist und auch weiterhin gesetzt sein soll. In drei von vier Fällen hat diese Verknüpfung also als Ergebnis die Hintergrundfarbe. Gibt man dem CHR$(23) eine 3 als Parameter mit auf den Weg, so erhält man wieder die XOR-Verknüpfung. Mit einer 0 erreichen wir wieder den Normalzustand.

Wozu Steuercodes?

Wozu können diese Kommandos nun nützlich sein? Der Computer benutzt diese Techniken beispielsweise, um die Cursorführung sichtbar zu machen. Verknüpft man nämlich auf einer Stelle des Schirms zwei Informationen nacheinander zweimal mit der XOR-Funktion, so ändert sich im Prinzip nichts. Dennoch ist bei einmaliger Anwendung sichergestellt, daß sich die so bearbeitete Stelle deutlich vom Restschirm abhebt. Um dies auszuprobieren, sollten Sie Listing 1 eintippen.

Es stellt zuerst in der linken oberen Ecke des Bildschirms ein »A« dar. Als nächstes, das heißt nach Tastendruck, wird dieses mit CHR$(143), dem ausgefüllten Qua-

drat, XOR-verknüpft. Nochmaliger Tastendruck schaltet wieder in den Ausgangszustand zurück. Probieren Sie das Programm doch einmal in den verschiedenen Modi, um sich mit der Funktion dieser Verknüpfung vertraut zu machen!

Der wohl interessanteste Steuercode ist CHR$(22). Er spricht die Transparent-Option an. Normalerweise legt der CPC bei jedem ausgedruckten Zeichen eine Matrix von 8x8 Punkten im Grafikspeicher ab (Beginn bei C000 hex). Dazu werden die einzelnen Punkte mit dem Wert des Farbregisters geladen, das für die Farbdarstellung benutzt wird. Im Grafikspeicher findet man also immer noch die Nummer der zu benutzenden Farbe. Erst bei der Ausgabe der einzelnen Punkte durch die Videobausteine wird dann die Farbe eingesetzt. Dadurch ist es möglich, durch die Umdefinierung eines Farbregisters die Farbe auf dem gesamten Schirm auf einen Schlag zu ändern.

Wenn der Schneider nun ein Zeichen ausgibt, dann setzt er einen Hintergrundpunkt auf die PAPER-Farbe und für die Darstellung des Vordergrundes die Schriftfarbe. Bei eingeschalteter Transparent-Option unterbleibt nun das Schreiben des Hintergrundes. Es wird nur noch der für die Schrift gewählte Farbwert gesetzt. Das hat zur Folge, daß alte Zeichen auf dem Bildschirm stehenbleiben. Um Ihnen einen Eindruck von diesem Effekt zu vermitteln, sollten Sie das folgende kleine Experiment durchführen. Geben Sie einmal in einer Zeile ein paar Buchstaben ein und führen Sie dann in einer anderen, möglichst freien Zeile, den Befehl »PRINT CHR$(22)« aus. Versuchen Sie nun einmal den oben geschriebenen Text mit DELETE zu löschen, oder durch Überschreiben zu vernichten.

Die Zeichen lassen sich nämlich nicht mehr löschen. Dies bewirkt der Überlagerungseffekt, der es ermöglicht, Zeichen übereinander zu drucken. Wir können dies nun in vielen Bereichen anwenden. Wollen wir zum Beispiel ein neues Zeichen aus zwei alten entstehen lassen, so ist dies mit diesem Code einfach durch Überschreiben möglich. Sie geben zweimal hintereinander verschiedene Zeichen an derselben Position aus, und schon haben Sie Ihr neues Bildschirmsymbol definiert. Dies ist in manchen Fällen einfacher, als mit SYMBOL umständlich ein neues Zeichen zu definieren.

Multicolor-Zeichen

Bis jetzt haben wir nur mit einer Farbe gearbeitet. Dies ist aber keineswegs zwingend. Wir können auch zwischen dem Setzen der Zeichen die Schriftfarbe ändern. Aus einem zweifarbigen Symbol erhalten wir dann durch Überdrucken ein Multicolor-Symbol. Treffen die beiden Farben den gleichen Punkt, so bleibt aber nur der zuletzt gesetzte erhalten. Noch weitergehende Effekte erhalten wir, wenn wir CHR$(22) mit CHR$(23) kombinieren. Da es wie schon gesagt, problemlos möglich ist, verschiedene Zeichen durch geeignete Cursorsteuerbefehle verbunden, nacheinander auszugeben, ist ein solches »Gebilde« als String speicherbar. Wir können damit auch in Basic mit Shapes (=Software-Sprites) arbeiten. Als Beispiel hier ein dreifarbiges Raumschiff-Shape:

```
10 SYMBOL 250,&x00011000,&x
00011000,&x00011000,&x00011000,
&x10011001,&x10111101,&x11111111,
&x11011011
20 SYMBOL 251,&x0,&x00100100,
&x0,&x10000001,&x0,&x0,&x0,&x
00100100
30 MODE 0:BORDER 0:INK 0,0:INK
1,27:INK 2,24,6:CLS
40 PRINT CHR$(22):LOCATE 10,12:
PEN 1:PRINT CHR$(250)+CHR$(15)
+CHR$(2)+CHR$(8)+CHR$(251)
+CHR$(22)+CHR$(0)
50 IF INKEY$="" THEN 50 ELSE
PEN 1:END
```

Bild 1 zeigt Ihnen die Symbole aus denen das Raumschiff entsteht. Für jeden gesetzten Punkt ist eine 1 im SYMBOL-Befehl nötig. Die anderen Punkte werden mit 0 ausgefüllt. Dies ist jedoch kein Problem, erfordert aber etwas Kreativität. In Zeile 30 werden dann die Eckdaten für die Ausgabe gesetzt. Die eigentliche Zeichenausgabe, wobei bereits in breitem Rahmen auf die Systemsteuercodes zurückgegriffen wird, steht in Zeile 40. Zunächst wird die Transparent-Option eingeschaltet und die Ausgabeposition festgelegt. Die Vordergrundfarbe wird auf 1 gesetzt. Für die letzten beiden Angaben wurden dabei Basic-Befehle benutzt. Wir hätten aber genausogut mit CHR$(31) und CHR$(15) arbeiten können (wie hätten dann die Characterfolgen gelautet?). Dann folgt ein »CHR$-Bandwurm«. Schauen wir uns die dabei ablaufenden Funktionen einmal Zeichen für Zeichen an. Zunächst wird in der gerade gesetzten Grundfarbe die erste Zeichenebene definiert: das Rumpfraumschiff. Die nächsten beiden Characters bewirken mittels PEN-Umschaltung das Setzen der zweiten Vordergrundfarbe, mit der jetzt einige Effekte — wie das Triebwerksfeuer und die Kanzelbeleuchtung, überlagert werden. Aus der blinkenden Farbe 2 ergibt sich der »realitätsnahe« Effekt. Bei der Ausgabe der zweiten Vordergrundfarbe ist darauf zu achten, daß der Cursor zuerst wieder zurückbewegt werden muß (CHR$(8)). Ansonsten wird das zweite Zeichen nämlich nicht über, sondern hinter das erste gesetzt. Abschließend sollten Sie übrigens nie vergessen, die Transparent-Option wieder abzuschalten.

```
10 '*****************
20 'Grafik-Mode-Demo
30 '*****************
40 CLS:PRINT CHR$(23)+CHR$(1)
50 LOCATE 1,1:PRINT"A"
60 IF INKEY$="" THEN 60
70 MOVE 0,399:TAG:PRINT CHR$(143);
80 IF INKEY$="" THEN 80
90 MOVE 0,399:PRINT CHR$(143);
```

Listing 1. Ein »A« wird transparent

```
10 SYMBOL 250,&X11000,&X11000,&X11000,&X
11000,&X10011001,&X10111101,&X11111111,&
X11011011
20 SYMBOL 251,&X0,&X100100,&X0,&X1000000
1,&X0,&X0,&X0,&X100100
30 MODE 1:INK 0,0:INK 1,27:INK 2,24,6:BO
RDER 0
40 PEN 1:PRINT CHR$(22)+CHR$(1):LOCATE 1
0,12:PRINT CHR$(250)+CHR$(15)+CHR$(2)+CH
R$(8)+CHR$(251)+CHR$(22)+CHR$(0)
45 FOR i=1 TO 50:PLOT 640*RND(1),400*RND
(1),2:NEXT i
50 IF INKEY$="" THEN 50 ELSE PEN 1:END
```

Listing 2. Ein dreifarbiges Raumschiff im Weltall

CHR$(4)	Ersatz für MODE-Befehl
CHR$(7)	Bell; es ertönt ein Piepser im Lautsprecher
CHR$(8)	Cursor links
CHR$(9)	Cursor rechts
CHR$(10)	Cursor runter; gegebenenfalls Scrollen nach oben
CHR$(11)	Cursor rauf; gegebenenfalls Scrollen nach unten
CHR$(12)	CLEAR/COME; entspricht CLS
CHR$(13)	RETURN; Sprung auf Zeilenanfang
CHR$(14)	Ersatz für PAPER: ein Parameter; zu setzende Farbe
CHR$(15)	Ersatz für PEN: ein Parameter: zu setzende Farbe
CHR$(17)	Löschen vom Zeilenanfang bis zum Cursor
CHR$(18)	Löschen vom Cursor bis zum Zeilenende
CHR$(19)	Löschen von Anfang des Windows bis zum Cursor
CHR$(20)	Löschen von Cursor bis Ende des Windows
CHR$(22)	Transparent-Option: ein Parameter 1=EIN, 2=AUS
CHR$(23)	Grafikstiftmodus: ein Parameter:1 oder 3=XOR, 2=AND, 0=normal
CHR$(24)	Revers: Austausch von PEN und PAPER-Farbe
CHR$(25)	SYMBOL-Ersatz: neun Parameter: Nummer und acht Punktzeilen
CHR$(26)	WINDOW-Ersatz: vier Parameter: rechter, linker Rand, Zeilenangaben
CHR$(28)	INK-Ersatz: drei Parameter für Nummer und zwei Farbangaben
CHR$(30)	HOME; Cursor auf Bildschirmanfang
CHR$(31)	Ersatz für LOCATE: zwei Parameter für x,y-Koordinate

Nützliche Systemsteuerzeichen

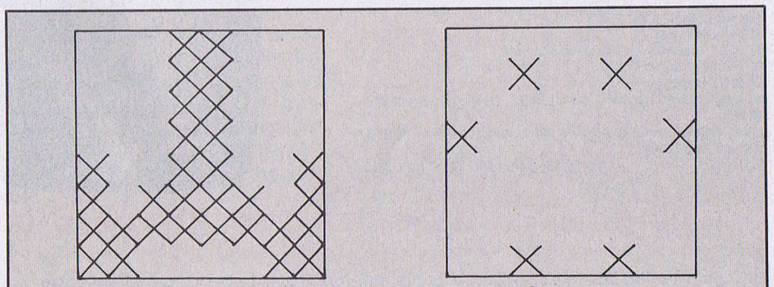
Auf eine Besonderheit bei der Transparent-Option soll an dieser Stelle noch aufmerksam gemacht werden. Da der Schneider jedes Zeichen im Grafikspeicher effektiv in Punkten abgelegt hat, kann er also den »Überdruck« nicht mehr nachträglich erkennen. Wenn Sie also versuchen, einen überdruckten Text, der einen Befehl enthält mit Copy übernehmen zu wollen, so geht dies nicht. Der CPC kennt die neuen Zeichen nämlich nicht. Mit Hilfe dieser Methode können nun natürlich auch problemlos mehrfarbige Zeichen beliebiger Größe definiert werden. So kann zum Beispiel der Figurensatz eines Schachspiels definiert werden, sogar in 3D-Effekt (mit Schattengebung oder ähnlichen Effekten). Die Definition der Symbole sieht auf den ersten Blick etwas schwierig aus, aber mit einem kleinen Editor ist sie problemlos möglich. Wenn ein Zeichen erst einmal definiert ist, kann es mehrmals gleichzeitig auf den Bildschirm gebracht werden, ohne daß viel Speicherplatz gebraucht würde.

Man ist aber beim CPC natürlich nicht auf die einseitige Verwendung der Blockgrafik festgelegt. Kombination ist Trumpf. Es wäre zum Beispiel völlig unökonomisch nun etwa eine Linie mit Hilfe dieser Art von Grafik ziehen zu wollen. Dazu gibt es ja schließlich DRAW. Aber auch eine punktweise Änderung ist in einigen Dingen durchaus sinnvoll. So kann in einem fertigen Zeichen zum Beispiel noch nachträglich ein Punkt mit PLOT gesetzt werden. Aber auch die Gestaltung des Umfeldes kann auf hübsche Weise verbessert werden. Man sollte sich also vorab bei der Grafik damit beschäftigen, auf welche Weise man welche gewünschte Ausgabe effektiv erreichen kann. Das Rüstzeug dazu haben Sie jetzt erworben.

Wenn Ihnen das Raumschiff beispielsweise zu einsam in des Weltalls weiter Ferne schwebt, so kann bereits eine kurze PLOT-Schleife helfen und der Himmel ist voller blinkender Sterne:

```
45 FOR i=1 TO 50:PLOT
640*RND(1),400*RND(1),2:NEXT i
```

(Carsten Straush)



Das Raumschiff... ...mit Farbeffekten

SYBEX neu

376 Seiten, 124 Abb.
Best.-Nr. **3602,** ca. DM **42,–**

Buch + Kassette
mit Z80-Assembler
Best.-Nr. **3412,** DM **64,–**

288 Seiten, 113 Abb.
Best.-Nr. **3603,** ca. DM **38,–**

SYBEX **Überall, wo es gute Computerbücher gibt!**

Unglaublich!

Spitzen-Software zum populären Markt&Technik-Preis für Schneider CPC

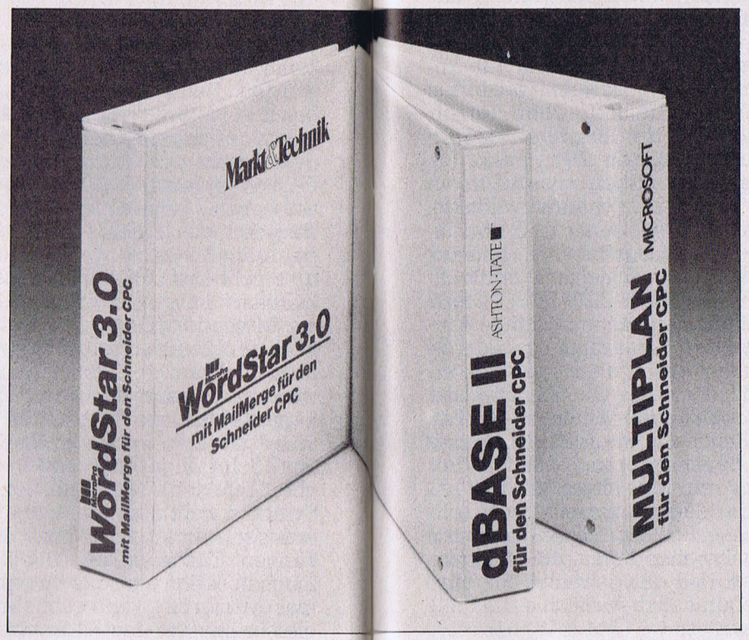
MicroPro® ASHTON·TATE MICROSOFT®

WordStar, dBASE II und MULTIPLAN

WordStar 3.0 mit MailMerge Der Bestseller unter den Textverarbeitungsprogrammen für PCs bietet Ihnen bildschirmorientierte Formatierung, deutschen Zeichensatz und DIN-Tastatur sowie integrierte Hilfstexte. Mit MailMerge können Sie Serienbriefe mit persönlicher Anrede an eine beliebige Anzahl von Adressen schreiben und auch die Adreßaufkleber drucken.
WordStar/MailMerge für den Schneider CPC 464*, 664*
Bestell-Nr. MS 101 (3″-Diskette)
Bestell-Nr. MS 102 (5¼″-Diskette im VORTEX-Format)
WordStar/MailMerge für den Schneider CPC 6128
Bestell-Nr. MS 104 (3″-Diskette)
Hardware-Anforderungen: Schneider CPC 464*, CPC 664* oder CPC 6123, beliebiger Drucker mit Centronics-Schnittstelle
* Der Standard-Speicherplatz beim CPC 464/664 erlaubt ohne VORTEX-Speichererweiterung Blockverschiebe-Operationen nur bedingt und Simultan-Drucken gar nicht.
DM 199,–* (sFr. 178,-) **Ab 15. Oktober: DM 299,–* (sFr. 260,-)**

dBASE II, Version 2.41 dBASE II, das meistverkaufte Programm unter den Datenbanksystemen, eröffnet Ihnen optimale Möglichkeiten der Daten- u. Dateihandhabung. Einfach u. schnell können Datenstrukturen definiert, benutzt und geändert werden. Der Datenzugriff erfolgt sequentiell oder nach frei wählbaren Kriterien, die integrierte Kommandosprache ermöglicht den Aufbau kompletter Anwendungen wie Finanzbuchhaltung, Lagerverwaltung, Betriebsabrechnung usw.
dBASE II für den Schneider CPC 464*
Bestell-Nr. MS 301 (3″-Diskette)
Bestell-Nr. MS 302 (5¼″-Diskette im VORTEX-Format)
dBASE II für den Schneider CPC 6128
Bestell-Nr. MS 304 (3″-Diskette)
Hardware-Anforderungen: Schneider CPC 464*, 6128, beliebiger Drucker mit Centronics-Schnittstelle
* dBASE II für den Schneider CPC 464 ist nur lauffähig mit der VORTEX-Speichererweiterung auf 128 KByte. Diese erhalten Sie direkt bei der Firma VORTEX oder bei Ihrem Computerhändler.
DM 199,–* (sFr. 178,-) **Ab 15. Oktober: DM 299,–* (sFr. 260,-)**

MULTIPLAN, Version 1.06 Wenn Sie die zeitraubende manuelle Verwaltung tabellarischer Aufstellungen mit Bleistift, Radiergummi und Rechenmaschine satt haben, dann ist MULTIPLAN, das System zur Bearbeitung »elektronischer Datenblätter« genau das richtige für Sie! Das benutzerfreundliche und leistungsfähige Tabellenkalkulationsprogramm kann bei allen Analyse- und Planungsberechnungen eingesetzt werden, wie z.B. Budgetplanungen, Produktkalkulationen, Personalkosten usw. Spezielle Formatierungs-, Aufbereitungs- und Druckanweisungen ermöglichen außerdem optimal aufbereitete Präsentationsunterlagen!
MULTIPLAN für den Schneider CPC 464*
Bestell-Nr. MS 201 (3″-Diskette)
Bestell-Nr. MS 202 (5¼″-Diskette im VORTEX-Format)
MULTIPLAN für den Schneider CPC 6128
Bestell-Nr. MS 204 (3″-Diskette)
Hardware-Anforderungen: Schneider CPC 464*, 6128, beliebiger Drucker mit Centronics-Schnittstelle
* MULTIPLAN für den Schneider CPC 464 ist nur lauffähig mit der VORTEX-Speichererweiterung auf 128 KByte.
DM 199,–* (sFr. 178,-) **Ab 15. Oktober: DM 249,–* (sFr. 225,-)**



Dies sind die ersten drei weltbekannten Software-Produkte für Schneider CPC. Weitere folgen in Kürze!

Sie erhalten jedes **WordStar-, dBASE II-** und **MULTIPLAN-Programm** für Ihren Schneider CPC fertig angepaßt (Bildschirmsteuerung und Druckerinstallation). **Jeweils Originalprodukte**! Jedes Programmpaket enthält außerdem ein ausführliches Handbuch mit kompakter Befehlsübersicht. Die VORTEX-Speichererweiterung für den Schneider CPC 464 erhalten Sie direkt bei der Firma VORTEX oder bei Ihrem Computerhändler.

Und dazu die richtige Literatur:

Mit diesem Buch haben Sie eine wertvolle Ergänzung zum WordStar-Handbuch: Anhand vieler Beispiele steigen Sie mühelos in die Praxis der Textverarbeitung mit **WordStar** ein. Angefangen beim einfachen Brief bis hin zur umfangreichen Manuskripterstellung zeigt Ihnen dieses Buch auch, wie Sie mit Hilfe von MailMerge Serienbriefe an eine beliebige Anzahl von Adressen mit persönlicher Anrede senden können.
WordStar für den Schneider CPC
Best.-Nr. MT 779, ISBN 3-89090-180-8

Zu einem Weltbestseller unter den Datenbanksystemen gehört auch ein klassisches Einführungs- und Nachschlagewerk! Dieses Buch von dem deutschen Erfolgsautor Dr. Peter Albrecht begleitet Sie mit nützlichen Hinweisen, die nur von einem Profi stammen können, bei Ihrer täglichen Arbeit mit **dBASE II**. Schon nach Beherrschung weniger Befehle ist der Einsteiger in der Lage, Dateien zu erstellen, mit Informationen zu laden und auszuwerten.
dBASE II für den Schneider CPC
Best.-Nr. MT 837, ISBN 3-89090-188-3

Dank seiner Menütechnik ist **MULTIPLAN** sehr schnell erlernbar. Mit diesem Buch von Dr. Peter Albrecht werden Sie Ihre Tabellenkalkulation ohne Probleme in den Griff bekommen. Als Nachschlagewerk leistet es auch dem Profi nützliche Dienste.
MULTIPLAN für den Schneider CPC
Best.-Nr. MT 835, ISBN 3-89090-186-7

Jedes Buch kostet DM 49,– (sFr. 45,10).
Erhältlich bei Ihrem Buchhändler.

Diese Markt&Technik-Softwareprodukte erhalten Sie in den Computer-Abteilungen der Kaufhäuser Horten, Karstadt, Kaufhof, Quelle oder bei Ihrem Computerhändler.

Wenn Sie direkt beim Verlag bestellen wollen: per Nachnahme oder gegen Vorauskasse durch Verrechnungsscheck oder mit der eingehefteten Zahlkarte.
Bestellungen im Ausland bitte an nebenstehende Adressen.
Für Auskünfte steht Ihnen Herr Barsa, Tel. 089/46 13-133, gerne zur Verfügung.

Zum sensationellen Einführungspreis: DM 199,–* (sFr. 178,–)
nur bis 14. Oktober 1985!

Ab 15. Oktober kosten:

WordStar/ MailMerge	DM 299,—* (sFr. 260,—)
dBase II	DM 299,—* (sFr. 260,—)
Multiplan	DM 249,—* (sFr. 225,—)

* inkl. MwSt. Unverbindliche Preisempfehlung

Markt&Technik
Verlag Aktiengesellschaft
Hans-Pinsel-Straße 2, 8013 Haar bei München
Schweiz: Markt&Technik Vertriebs AG, Kollerstrasse 3, CH-6300 Zug, ☎ 042/22315 5
Österreich: Micromput-ique Schiller, Fasangasse 21, A-1030 Wien, ☎ 0222/785661

Grafik aus dem ROM gelockt

Der Basic-Interpreter des Schneiders arbeitet nicht direkt mit der Hardware zusammen, sondern benutzt allein für Grafik mehr als 100 Firmware-Routinen. Diese kann man natürlich als Programmierer auch selbst anwenden. Die interessantesten lernen Sie hier kennen.

Der Schneider-Computer hat eine vollkommen neuartige offene Gestaltung des Betriebssystems. Wurden frühere Heim- und Personal Computer noch weitgehend direkt programmiert, so hat man beim CPC viel stärker auf Sprungtabellen und Interruptsteuerung zurückgegriffen. Im Rhythmus des Systemtaktes werden verschiedene Unterprogramme, wie Tastaturabfrage und Bildschirmkontrolle, aufgerufen, die, wie in einem Katalog, alle im sogenannten Firmware-Jumpblock mit »Name und Adresse« aufgeführt sind. Das System benutzt sie reichlich, aber auch wir als Anwender können auf diese »Ansprungpunkte« zurückgreifen. Kennzeichen all dieser Einsprungadressen ist es, daß sie mit genau definierten Ein- und Aussprungbedingungen eine bestimmte exakt definierte Funktion ausführen. Man darf sie also meist wie Basic-Kommandos behandeln, nur daß man sich hier natürlich mit etwas anderen Übergabegrößen beschäftigen muß. Wir wollen untersuchen, welche Routinen uns das Betriebssystem zur Verfügung stellt und welche davon nützlich für Grafikausgabe sind. Es muß allerdings eine Warnung vorab gegeben werden. Die Grafik- und Textroutinen belegen den meisten Platz im sogenannten Firmware-Jumpblock. Es stehen allein über 100 Routinen zur Verfügung, mit denen sich sehr schöne Effekte auf dem Bildschirm zaubern lassen. Hier eine vollständige Einführung in alle Routinen zu geben, ist natürlich unmöglich. Statt dessen beschränken wir uns auf Spezialitäten und einfache, schnell nutzbare Routinen. Es sind solche, die Sie leicht in eigene Programme einbauen können.

Beginnen wir mit dem Cursor. Die Cursor-Darstellung können wir mit den Unterroutinen TXT PLACE CURSOR BB8A und TXT REMOVE CURSOR BB8D beeinflussen. Das TXT besagt dabei, daß es sich um eine Routine einer bestimmten »Unterabteilung« des Firmware-Jumpblocks handelt. Es folgt der Name der Funktion und danach die Einsprungadresse dieser Routine. Die erste Routine setzt das Cursorzeichen auf den Bildschirm, die zweite löscht es wieder. Dabei ist zu beachten, daß jede dieser Routinen nur einmal hintereinander aufgerufen werden darf; nach TXT PLACE CURSOR muß also erst TXT REMOVE CURSOR bearbeitet werden, bevor man TXT PLACE CURSOR wieder durchführen kann. Ändert man nämlich zwischenzeitlich die aktuelle Cursorposition, so bleibt das alte Cursorzeichen auf dem Bildschirm stehen. Außerdem muß man noch beachten, daß TXT REMOVE CURSOR bei falscher Anwendung ein Cursorzeichen auf dem Bildschirm darstellt, auch wenn an dieser Stelle eigentlich keines ist. Paßt man nicht auf, so erscheinen Cursorsymbole dort auf dem Bildschirm, wo man sie am wenigsten erwartet. Dies ist auch dann der Fall, wenn die Oberherrschaft über die Ausgabe an den Basic-Interpreter zurückgegeben wird, ohne vorher den Cursor rückgesetzt zu haben. Wenn der Interpreter nämlich eine neue Position besetzen will, so interessiert er sich nicht dafür, daß bereits ein Cursorblock gezeichnet ist. Er bildet sein eigenes Zeichen neu ab. Man hat in diesem Fall also zwei Cursorsymbole auf dem Bildschirm.

Den eigentlichen Aufruf führt man mit einem CALL-Befehl (CD 8A BB) durch. In Basic lautet das Kommando CALL. Probieren Sie einmal die folgenden Zeilen aus:

```
10  CLS:CALL&BB8A
20  IF INKEY$="" THEN 20
30  CALL&BB8D
40  IF INKEY$=" " THEN 40
```

Die Befehle simulieren, wie die beiden Routinen von Basic aus zur Darstellung eines Cursors benutzt werden können. Die erste Zeile setzt den Cursorblock. Durch Tastendruck (Zeile 20) wird er wieder gelöscht (Zeile 30) und nochmaliges Tippen bricht das Programm ab. Das Setzen der Cursorposition kann man in diesem Fall mit dem Basic-Kommando LOCATE erreichen.

Will man an einer bestimmten Stelle des Bildschirms Zeichen von Maschinencode aus setzen oder lesen, so muß zunächst dieser Bereich als Ein- und Ausgabeeinheit adressiert werden. Beim CPC wird dabei das STREAM-Konzept benutzt, das heißt dem Ein- und Ausgabegerät wird eine Nummer zugeordnet, mit der es angesprochen wird. Alle Änderungen, wie beispielsweise das Setzen von Schrift- oder Hintergrundfarbe, die Ausgabe von Fensterbegrenzungen (Windows), das Zulassen oder Sperren des Cursors oder auch des Zeichendarstellungsmodus, setzen voraus, daß der aktuelle Ein-/Ausgabekanal definiert wurde. Dies geschieht mit der Routine TXT STR SELECT BBB4. Diese wählt aus den acht möglichen Windows (0 bis 7) das aktuelle aus. Alle weiteren Ansprachen laufen dann auf dieses Gerät. Die Nummer des auszuwählenden Windows ist dabei der Routine in Register A mitzugeben. Beim Einsprung enthält A den vorher ausgewählten Stream, HL und die Flags sind zerstört. Man sollte diese Routine immer vor einer Ausgabe anspringen, um sicherzustellen, daß das gewählte Ein-/Ausgabegerät auch wirklich als aktuell definiert ist.

Abfrage der aktuellen Schriftfarbe

Wir wollen uns nun damit beschäftigen, wie wir die einzelnen Daten der Ein-/Ausgabegeräte ändern können. Das Setzen der Schriftfarbe geschieht mit der Routine TXT SET PEN BB90.

Register A enthält dabei die zu benutzende Farbe. Von Maschinensprache aus läßt sich dabei auch etwas erreichen, was von Basic aus nicht möglich ist: wir können die aktuelle Schriftfarbe abfragen. Dies geschieht mit der Routine TXT GET PEN BB93.

Register A enthält wieder die Farbe — und zwar die Schriftfarbe. In Analogie zu den vorne genannten Routinen existieren auch zwei Unterprogramme für die Hintergrundfarbe. Es sind dies TXT SET PAPER BB96 und TXT GET PAPER BB99.

Die Ein- und Aussprungbedingungen sind hier analog der Schriftroutinen.

Nachdem wir nun Schrift- und Hintergrundfarbe setzen können, stellt sich die Frage, wie es mit einer schnellen Vertauschung der beiden aussieht. Auch dies ist mit einer Systemroutine möglich. Sie heißt TXT INVERSE BB9C. Wir können dieses Unterprogramm — wie auch die Cursorsteuerungsroutinen — auch von Basic aus direkt ansprechen, da es keine Parameter benötigt. TXT INVERSE ist dabei eine reine Vertauschungsroutine. Sie ist unabhängig von der aktuellen Definition der Schrift- oder Hintergrundfarben. Um den Einsprung von Basic aus zu benutzen, geben wir einfach »CALL&BB9C« ein. Dadurch werden Vorder- und Hintergrundfarbe vertauscht. Nochmaliges Aufrufen der Routine stellt den Ausgangszustand wieder her.

Um verschiedene Werte zwischen zwei Ein-/Ausgabegeräten zu vertauschen, gibt es die Routine TXT SWAP STREAMS BBB7. Diese vertauscht die Werte für die Hinter- und Vordergrundfarbe, die Cursorposition und die Bildschirm-beziehungsweise Window-Grenzen, sowie den Zeichenmodus mit den Werten des zweiten Fensters. Register B enthält dabei die Nummer des einen Streams, Register C die des anderen. Beim Aussprung sind alle Registerpaare (bis auf die Indexregister) zerstört. Diese Operation ist nur zwischen Windows, nicht jedoch zwischen Textfenstern und wirklichen externen Geräten, wie Drucker oder Datacorder, möglich.

Damit haben Sie eine Reihe der wichtigsten Firmware-Routinen kennengelernt. All diese Routinen gelten auch für die neuen Schneider-Computer, den CPC 664 und den 6128, obwohl diese in einigen anderen Routinen des Sprungblocks ganz anders aufgebaut sind.

Kommen wir zu einigen Routinen speziell aus dem Grafik-Bereich. Alle Routinen für die Grafik sind abhängig von der aktuellen Stellung des Grafik-Cursors. Dieser läßt sich absolut (Angabe der neuen Position) oder relativ (Angabe der Verschiebung) bewegen. An seiner aktuellen Position können Punkte gezeichnet oder von dort Linien gezogen werden. Bei allen Operationen ist darüber hinaus zu berücksichtigen, daß eine Ausgabe auf dem Bildschirm nur dann erfolgt, wenn der Grafik-Cursor innerhalb eines vom Benutzer vorgegebenen Bereiches des Bildschirms liegt, dem Grafik-Fenster.

Zum Setzen des Grafik-Cursors gibt es im Firmware-Jumpblock den Einsprung GRA MOVE ABSOLUTE BBC0. Dieser setzt den Grafik-Cursor auf den durch die Register DE (X-Koordinate) und HL (Y-Koordinate) definierten Bildschirmpunkt. Die Angabe erfolgt dabei immer relativ zum Ursprung (unten links). Diese Routine stellt also das Gegenstück für den Basic-Befehl MOVE dar. Eine relative Bewegung kann man mit GRA MOVE RELATIVE BBC3 erreichen. Die Register sind dieselben. Allerdings wird hier die Position relativ zum Ursprung (wie bei dem Basic-Befehl MOVER) angegeben. Wollen wir die aktuelle Position des Cursors abfragen (Basic-Befehle XPOS, YPOS), so können wir die Routine GRA ASK CURSOR BBC6 benutzen. Diese liefert uns in den Registern DE und HL die entsprechenden Werte. Das Setzen des Ursprungs können wir mit GRA SET ORIGIN BBC9 erreichen. Hier enthalten die Register DE und HL die Koordinaten. Mit den nächsten Befehlen ist es möglich, Punkte zu setzen und Linien zu ziehen. Es sind PLOT ABSOLUTE BBEA, GRA PLOT RELATIVE BBED, GRA LINE ABSOLUTE BBF6 und GRA LINE RELATIVE BBF9.

Routinen selbst genutzt

Die beiden ersten Routinen sind dabei für das Setzen eines Punktes, die unteren beiden für das Zeichnen von Linien zuständig. In allen Routinen erfolgt die Angabe von absoluter oder relativer Position im Register DE (X-Koordinate beziehungsweise X-Abstand) und im Register HL (Y-Koordinate beziehungsweise Y-Abstand). Die Funktionen und auch die Definitionen entsprechen dabei den Basic-Kommandos PLOT, PLOTR, DRAW und DRAWR. Wir wollen uns zum Schluß kurz anschauen, wie wir diese Grafikroutinen in eigene Programme benutzen können.

Als erstes ziehen wir eine Linie vom Punkt (0,100) zum Punkt (200,100), also einen waagrechten Strich. Dazu setzen wir den Grafik-Cursor auf (0,100) und rufen dann die Routine GRA LINE ABSOLUT auf. Die Register DE und HL sind dazu mit 200 und 100 geladen.

```
LD DE, 0000          11 00 00
LD HL, 0064          21 64 00
GRA MOVE ABSOLUTE    CD C0 BB
LD DE, 00C8          11 C8 00
CALL GRA LINE ABSOLUTE  CD F6 BB
RET                  C9
```

Dies ist das dazu notwendige Assemblerlisting. Rechts daneben finden Sie die zugehörigen hexadezimalen Codes. Nun müssen wir nur noch einen Platz finden, in den wir diese Werte laden können. Dazu bietet sich der Bereich knapp unterhalb der Obergrenze des Speichers an. Diese Obergrenze läßt sich ja mit MEMORY leicht nach unten verschieben. Zwischen der alten und der neuen Grenze ist dann Platz für die Maschinenprogramme. Und dort ist sie auch vor dem Überschreiben sicher. Ein kleines Basic-Programm, das diese Verschiebung durchführt und gleich auch noch die Werte in den reservierten Speicherplatz schreibt, könnte zum Beispiel so aussehen:

```
10 DATA 11,00,00,21,64,00,CD,C0,BB,
11,C8,00,CD,F6,BB,C9,00
20 MEMORY HIMEM-20:ca=HIMEM+1
30 FOR i=ca TO ca+16:READ a$:
POKE i,VAL("&"+a$):NEXT i
40 CALL ca
```

Nach der Ausführung dieses Programms erhalten Sie die gewünschte Linie. Da unser Maschinenprogramm jetzt gespeichert ist, können wir es jederzeit aufrufen. Natürlich ist es nicht besonders sinnvoll, immer dieselbe Linie zu zeichnen. Diese Routine sollte aber auch nur das Prinzip der Ansprache von Firmware klarstellen und dazu anregen, sich einmal näher mit diesen Systemroutinen zu befassen. Sie können zum Beispiel durch mehrmalige Ausgabe einer Linie ein Quadrat oder ein Rechteck erzeugen.

Wie würde das Ganze nun für eine relative Bewegung lauten? Der Ablauf der Routinen wäre derselbe. Einzige Änderung: wir müßten Register HL auf 0 setzen, weil ansonsten ein Strich um 200 Einheiten in Y-Richtung, also nach (200,300) gezogen würde. Danach könnten wir dann GRA LINE RELATIVE aufrufen.

(Carsten Straush)

Malen wie auf einer Leinwand

Eine tolle Grafik haben die Schneider-Computer ja schon. Doch mit Basic-Befehlen ist es schwer schöne Bilder zu zeichnen. Mit »CPC-Paint« ist das einfacher. Sie müssen sich nur noch um Ihr Kunstwerk kümmern.

Ist es nicht deprimierend, wenn man einen Computer besitzt, dessen grafische Fähigkeiten im Heimbereich unübertroffen sind und wenn das Basic dieses Computers nur zwei Befehle kennt, um Linien zu zeichnen oder Punkte zu setzen? Wer dies auch so sieht, dem kann mit »CPC-Paint« geholfen werden.

Das Programm arbeitet mit zwei Bildschirmspeichern. Der zweite liegt im Bereich von 6000 hex bis 9 FFF. Dieser zweite Bildschirm dient dazu, das jeweilige am Bildschirm befindliche Bild zwischenzuspeichern, damit es durch eventuelles Überschneiden mit den Windows für die Anweisungen nicht zerstört wird.

Die Steuerung des Programmes erfolgt ausschließlich mit dem Joystick. Als erstes wird gefragt ob ein Bild von Band geladen werden soll. Durch Drücken der »ENTER«-Taste wird die Laderoutine aufgerufen. Drückt man den Feuerknopf, so erscheint am oberen Bildschirmrand die Menüleiste, von der aus die verschiedenen Untermenüs aufgerufen werden.

Durch seitliches Bewegen des Joysticks kann der Cursor (ein Pfeil) zwischen den einzelnen Menüteilen bewegt werden. In die jeweiligen Untermenüs gelangt man, indem man den Joystick nach unten zieht. Daraufhin erscheint, je nach Stellung des Pfeiles in der Statuszeile, eine Liste der Unterfunktionen. Anders ist es im Menüteil »Colour«. Dieser dient zur Auswahl der Zeichenfarbe. Die Wahl wird durch Bewegung des Joysticks in senkrechter Richtung getroffen. Die Änderung der Farben erfolgt durch seitliches Bewegen des Joysticks. Nach Druck auf den Feuerknopf gelangt man wieder in die Statuszeile.

Bei den anderen Hilfsmenüs werden die verschiedenen Funktionen aufgelistet. Auch diese werden durch den Joystick ausgewählt, wobei die jeweilige Funktion invers dargestellt wird. Horizontales Bewegen des Joysticks ruft die gewählten Funktionen auf.

Bei dem Menüteil »Tools« kann man zwischen folgenden Funktionen wählen:

LINE: Verbindet zwei Punkte durch eine Linie; die Punkte müssen zuvor durch Drücken des Feuerknopfes markiert werden

LINES: Verbindet jeweils zwei Punkte durch eine Linie. Es muß ein Anfangspunkt gesetzt werden; jeder weitere Punkt wird mit dem letzten verbunden

STREAMS: Zeichnet Strahlen von der jeweiligen Position des Cursors zu einem zuvor markierten Mittelpunkt

FRAME: Durch Markierung zweier gegenüberliegender Eckpunkte wird ein Rechteck gezeichnet

BOX: Dasselbe wie FRAME, nur mit Farbe ausgefüllt

TRIANGLE: Durch Kennzeichnung dreier beliebig gewählter Eckpunkte wird ein in der aktuellen Zeichenfarbe ausgefülltes Dreieck gezeichnet

CIRCLE: Der Mittelpunkt wird markiert und nach Angabe des Radius wird ein Kreis gezeichnet

DISK: Dasselbe wie CIRCLE, nur mit Farbe ausgefüllt

ELLIPSE: Nach Setzen des Mittelpunkts und Kennzeichnung der Ausdehnung in x-Richtung sowie in der y-Richtung, wird eine ausgefüllte Ellipse gezeichnet

SPRAY: Füllt einen, durch zwei zuvor gewählte Eckpunkte begrenzten Bereich durch ein zufälliges Muster von Farben

TEXT: Fügt Text an beliebiger Stelle ein. Zuerst muß man die Stelle, an der der Text erscheinen soll, festlegen. Danach wird durch vertikales Bewegen des Joysticks der gesamte Zeichensatz des CPC an der festgelegten Stelle durchlaufen. Erscheint das gewünschte Zeichen, kann durch horizontales Bewegen des Joysticks der Cursor um ein Textzeichen nach links oder rechts verschoben werden. Durch Drücken des Feuerknopfes wird das Zeichen gesetzt und eine neue Textposition gewählt.

Der Menüteil »Help« verfügt über folgende Funktionen:

LOAD: Lädt Bild von der Kassette nach

SAVE: Speichert Bild auf die Kassette

FILL: Füllt abgegrenzten Bereich mit der jeweiligen Zeichenfarbe. Dazu muß man nur den Cursor in den auszufüllenden Bereich bringen, und danach den Feuerknopf drücken.

ZOOM: Vergrößert einen Bereich von 38 x 20 Punkten auf den Bildschirm, so daß mit dem Joystick eventuelle Korrekturen gemacht werden können. Durch Drücken des Feuerknopfes wird ein Punkt in der jeweiligen Zeichenfarbe gesetzt, durch längeres Drücken kommt es zur Änderrung der Zeichenfarbe. Ist man mit den Korrekturen fertig, so steuert man den Cursor nach oben oder unten aus dem Bildschirm.

MIRROR: Verschiebt, vergrößert oder spiegelt bereits gezeichnete Teile an beliebigen Stellen des Bildschirms. Der zu verschiebende Bereich wird durch zwei gegenüberliegende Punkte festgelegt. Dann markiert man mit dem Cursor die Stelle, wohin der Bereich verschoben werden soll.

HARDCOPY: Druckt den Bildschirminhalt zweifarbig auf dem Schneider-Drucker aus.

ERASE: Beendet Malvorgang oder es erfolgt ein Neustart.

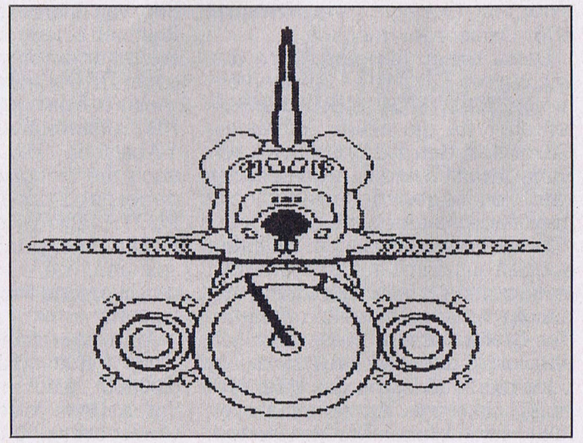
Noch einige wichtige Bemerkungen zur Arbeit mit »CPC PAINT«:

1.) Alle Routinen, die mit dem Cursor arbeiten, kann man verlassen, indem man diesen nach oben oder unten aus dem Bild steuert.

2.) Die Farbe des Bildschirmrandes ist gleich der Malfarbe.

3.) Die Bilder werden mit 3500 Baud gespeichert.

4.) Treten Komplikationen auf, zum Beispiel bei »FILL« oder »MIRROR«, so bricht man das Programm durch »ESC« ab und drückt die Taste »0« des Zehnerblocks. (Erhart Weis)



Sogar das Space-Shuttle kann man zeichnen


```
10  REM ****************************
20  REM *                          *
30  REM *    c p c - p a i n t     *
40  REM *                          *
50  REM *                          *
60  REM *   das irre malprogramm   *
70  REM *                          *
80  REM *         (c) 1985         *
90  REM *                          *
100 REM *    written & designed    *
110 REM *                          *
120 REM *           by             *
130 REM *                          *
140 REM *       erhart wais        *
150 REM *   gotelinedestrasze 3    *
160 REM *      3380 poechlarn      *
170 REM *                          *
180 REM *     tel.: 02757/347      *
190 REM *                          *
200 REM ****************************
210 GOTO 3350
220 REM *** menue
230 PEN 1:CALL 43000
240 LOCATE 7,1:PRINT"Tools      Help
Colour  "
250 LOCATE#1,pot+2,1:PRINT#1,CHR$(242)
260 jo%=JOY(0):IF jo%=0 THEN 260 ELSE FO
R i=1 TO 100:NEXT
270 LOCATE#1,pot+2,1:PRINT#1," "
280 pot=pot-(10 AND jo%=4)+(10 AND jo%=8
):IF pot<10 THEN pot=30 ELSE IF pot>30 T
HEN pot=10
290 LOCATE#1,pot+2,1:PRINT#1,CHR$(242)
300 IF jo%=2 THEN CALL 43020:ON pot/10 G
OTO 3070,3200,2330
310 IF jo%=16 THEN 3330
320 GOTO 260
330 REM  fill
340 GOSUB 450:IF back THEN 3200
350 pkt=TEST(xz,yz):a=0
360 IF yz<10 OR yz>390 THEN 430
370 IF TEST(xz-2,yz)=pkt AND xz>10 THEN
xz=xz-2:GOTO 360
380 m=4:n=4:xx=xz
390 o=m:m=TEST(xz,yz+2):IF m=pkt THEN IF
 o<>pkt THEN x(a)=xz:y(a)=yz+2:a=a+1
400 p=n:n=TEST(xz,yz-2):IF n=pkt THEN IF
 p<>pkt THEN x(a)=xz:y(a)=yz-2:a=a+1
410 IF TEST(xz+2,yz)=pkt AND xz<630 THEN
 xz=xz+2:GOTO 390
420 MOVE xx,yz:DRAW xz,yz,fa
430 IF a>0 THEN a=a-1:xz=x(a):yz=y(a):GO
TO 360
440 GOTO 340
450  REM  steuerung
460 back=0:halt=2:GOSUB 540
470 jo%=JOY(0):IF y>394 OR y<6 THEN back
=1:SOUND 1,100,100,13,,1:GOSUB 600:RETUR
N
480 IF jo%=0 THEN halt=2:GOTO 470
490 GOSUB 600:halt=halt+0.5
500 x=x-(halt AND jo%=4)+(halt AND jo%=8
)
510 y=y-(halt AND jo%=2)+(halt AND jo%=1
)
520 IF jo%=16 THEN xz=x:yz=y:GOSUB 600:F
OR i=0 TO 100:NEXT:RETURN
530 GOSUB 540:GOTO 470
540 REM fadenkreuz setzen
550 f1=TEST(x,y):f2=TESTR(4,-4):f3=TESTR
(2,-2):f4=TESTR(2,-2)
560 f5=TEST(x+2,y):f6=TEST(x+4,y):f7=TES
T(x,y-2):f8=TEST(x,y-4)
570 PLOT x,y,fa:PLOTR 4,-4,fa:PLOTR 2,-2
,fa:PLOTR 2,-2,fa
580 PLOT x+2,y,fa:PLOT x+4,y,fa:PLOT x,y
-2,fa:PLOT x,y-4,fa:IF f1=fa THEN PLOT x
,y,(fa+1) MOD 4
590 RETURN
600 REM loeschen
610 PLOT x,y,f1:PLOTR 4,-4,f2:PLOTR 2,-2
,f3:PLOTR 2,-2,f4
620 PLOT x+2,y,f5:PLOT x+4,y,f6:PLOT x,y
-2,f7:PLOT x,y-4,f8
630 RETURN
640 REM linie
650 GOSUB 2700:IF back THEN 3070
660 MOVE xa,ya:DRAW xz,yz,fa
670 GOTO 650
680 REM linien
690 GOSUB 450:IF back THEN 3070
700 xa=xz:ya=yz
710 GOSUB 2510:GOSUB 450:GOSUB 2600:IF b
ack THEN 3070
720 MOVE xa,ya:DRAW xz,yz,fa
730 GOTO 700
740 REM  strahlen
750 GOSUB 450:IF back THEN 3070
760 xa=xz:ya=yz
770 GOSUB 2510:GOSUB 450:GOSUB 2600:IF b
ack THEN 3070
780 MOVE xa,ya:DRAW xz,yz,fa
790 GOTO 770
800 REM viereck
810 GOSUB 2700:IF back THEN 3070
820 xx=MAX(xa,xz):yy=MAX(ya,yz):xy=MIN(x
a,xz):yx=MIN(ya,yz)
830 MOVE xy,yx:DRAW xx,yx,fa
840 DRAW xx,yy,fa:DRAW xy,yy,fa:DRAW xy,
yx,fa
850 GOTO 810
860 REM ausgefuelltes viereck
870 GOSUB 2700:IF back THEN 3070
880 xx=MAX(xa,xz):yy=MAX(ya,yz):xy=MIN(x
a,xz):yx=MIN(ya,yz)
890 FOR x=xy TO xx STEP 2
900 MOVE x,yx:DRAW x,yy,fa
910 NEXT
920 GOTO 870
930 REM  dreieck
940 GOSUB 450:IF back THEN 3070
950 xa=xz:ya=yz
960 GOSUB 2510:GOSUB 450:IF back THEN GO
SUB 2600:GOTO 3070
970 xb=xz:yb=yz
980 GOSUB 2620:GOSUB 450:GOSUB 2590:IF b
ack THEN 3070
990 drei=SQR((xa-xb)^2+(ya-yb)^2):IF dre
i=0 THEN MOVE xa,ya:DRAW xz,yz,fa:GOTO 9
40
1000 FOR a=0 TO drei STEP 0.5:MOVE xa+a*
(xb-xa)/drei,ya+a*(yb-ya)/drei
1010 DRAW xz,yz,fa:NEXT
1020 drei=SQR((xa-xz)^2+(ya-yz)^2):IF dr
ei=0 THEN 940
1030 FOR a=0 TO drei STEP 0.5:MOVE xa+a*
(xz-xa)/drei,ya+a*(yz-ya)/drei
1040 DRAW xb,yb,fa:NEXT
1050 GOTO 940
1060 REM circle
```

Listing. »CPC-Paint« — ein tolles Malprogramm

```
1070 GOSUB 2700
1080 IF back THEN 3070
1090 ra=SQR((xz-xa)^2+(yz-ya)^2):DEG
1100 FOR m=0 TO ra
1110 IF m>n THEN x=xa:y=ya:GOTO 1070
1120 n=SQR(ra*ra-m*m)
1130 PLOT xa-n,ya-m,fa:PLOT xa+n,ya-m,fa
1140 PLOT xa-n,ya+m,fa:PLOT xa+n,ya+m,fa
1150 PLOT xa-m,ya+n,fa:PLOT xa+m,ya+n,fa
1160 PLOT xa-m,ya-n,fa:PLOT xa+m,ya-n,fa
1170 NEXT
1180 GOTO 1070
1190 REM disk
1200 GOSUB 2700
1210 IF back THEN 3070
1220 ra=SQR((xz-xa)^2+(yz-ya)^2):DEG
1230 FOR m=0 TO ra
1240 IF m>n THEN x=xa:y=ya:GOTO 1200
1250 n=SQR(ra*ra-m*m)
1260 MOVE xa-n,ya-m:DRAW xa+n,ya-m,fa
1270 MOVE xa-n,ya+m:DRAW xa+n,ya+m,fa
1280 MOVE xa-m,ya+n:DRAW xa+m,ya+n,fa
1290 MOVE xa-m,ya-n:DRAW xa+m,ya-n,fa
1300 NEXT
1310 GOTO 1200
1320 GOSUB 2750:GOTO 3200
1330 REM load
1340 GOSUB 3030:PRINT#6," Press PLAY":PR
INT#6,"  then (ENTER)":PRINT#6:PRINT#6,"
  else press 'FIRE'"
1350 a$=INKEY$:IF a$=CHR$(13) THEN 1370
1360 IF JOY(0)=16 THEN CALL 43020:GOTO 2
20 ELSE GOTO 1350
1370 CALL 43020
1380 LOAD"!
1390 GOTO 220
1400 REM save
1410 GOSUB 3030:PRINT#6," Press REC & PL
AY":PRINT#6,"  then (ENTER)":PRINT#6:PRI
NT#6,"  else press 'FIRE'"
1420 a$=INKEY$:IF a$=CHR$(13) THEN 1440
1430 IF JOY(0)=16 THEN CALL 43020:GOTO 2
20 ELSE GOTO 1420
1440 CALL 43020
1450 SAVE"!...",b,&C000,&3FFF
1460 GOTO 220
1470 GOSUB 450:IF back THEN 3070: REM te
xt
1480 TAG:FOR i=0 TO 500:NEXT
1490 jo%=JOY(0):IF jo%=0 THEN 1490 ELSE
PLOT 640,400,fa
1500 FOR i=0 TO 50:NEXT:x=x+(16 AND jo%=
8)-(16 AND jo%=4)
1510 nom=nom-(1 AND jo%=2)+(1 AND jo%=1)
:IF nom<32 THEN nom=255 ELSE IF nom>255
THEN nom=32
1520 MOVE x,y:PRINT CHR$(nom);
1530 IF jo%=16 THEN TAGOFF:FOR i=0 TO 20
0:NEXT:GOTO 1470
1540 GOTO 1490
1550 REM erase
1560 GOSUB 3030:PRINT#6:PRINT#6," End of
 painting (y/n)?"
1570 a$=INKEY$:IF a$="" THEN 1570 ELSE a
$=LOWER$(a$)
1580 IF a$="y" THEN CLS:END
1590 IF a$<>"n" THEN 1570
1600 PRINT#6," Delete picture  (y/n)?"
1610 a$=INKEY$:IF a$="" THEN 1610 ELSE a
$=LOWER$(a$)
1620 IF a$="y" THEN RUN
1630 IF a$<>"n" THEN 1600
1640 CALL 43020
1650 GOTO 220
1660 REM zoom
1670 GOSUB 450:TAGOFF:IF back THEN 3200
1680 CALL 43000
1690 FOR yz=0 TO 40 STEP 2:FOR xz=0 TO 7
6 STEP 2
1700 far(xz/2,yz/2)=TEST(x+xz,y+40-yz)
1710 NEXT:NEXT
1720 PEN 1:LOCATE 1,2:PRINT CHR$(151)STR
ING$(38,CHR$(154))CHR$(157):LOCATE 1,23:
PRINT CHR$(151)STRING$(38,CHR$(154))CHR$
(157)
1730 xx=x:yy=y:xw=36:yw=16:x=312:y=166:G
OSUB 1940
1740 FOR yz=0 TO 19:FOR xz=0 TO 37
1750 LOCATE #4,xz+1,yz+1:PEN#4,far(xz,yz
):PRINT#4,CHR$(143);
1760 NEXT:NEXT
1770 GOSUB 540:TAG
1780 jo%=JOY(0):IF y<6 OR y>386 THEN SOU
ND 1,100,100,13,,1:GOTO 1860
1790 IF jo%=0 THEN 1780 ELSE FOR i=0 TO
30:NEXT
1800 GOSUB 600
1810 x=x-(16 AND jo%=4)+(16 AND jo%=8):x
w=xw-(2 AND jo%=4)+(2 AND jo%=8)
1820 y=y-(16 AND jo%=2)+(16 AND jo%=1):y
w=yw-(2 AND jo%=2)+(2 AND jo%=1)
1830 IF jo%=16 THEN PLOT 640,400,fa:BORD
ER fb(fa):MOVE x-8,y+8:PRINT CHR$(143);:
FOR i=0 TO 200:NEXT:PLOT 480+xw,2+yw,fa:
jo%=JOY(0):IF jo%=0 THEN GOSUB 540:GOTO
1780
1840 IF jo%=16 THEN PEN fa:fa=fa+1:IF fa
>3 THEN fa=0
1850 GOSUB 540:GOTO 1780
1860 FOR yz=3 TO 22:FOR xz=2 TO 39
1870 far(xz-2,yz-3)=TEST(xz*16-8,yz*16+8
)
1880 NEXT:NEXT
1890 TAGOFF:CALL 43020
1900 FOR yz=0 TO 38 STEP 2:FOR xz=0 TO 7
4 STEP 2
1910 PLOT xx+xz,yy+yz+2,far(xz/2,yz/2)
1920 NEXT:NEXT
1930 xz=320:yz=200:x=xz:y=yz:GOTO 1670
1940 LOCATE 30,23:PRINT CHR$(156)"
":LOCATE 36,23:PRINT CHR$(150):LOCATE 30
,24:PRINT CHR$(149):LOCATE 36,24:PRINT C
HR$(149):LOCATE 30,25:PRINT CHR$(153)"
    ":LOCATE 36,25:PRINT CHR$(147)
1950 m1=0:m2=0:FOR yc=2 TO 40 STEP 2:FOR
 xc=0 TO 76 STEP 2
1960 farbe=TEST(xc+xx,yc+yy):PLOT 480+xc
,yc+2,farbe
1970 NEXT:NEXT:RETURN
1980 REM ellipse
1990 RAD
2000 GOSUB 450:IF back THEN 3070
2010 xa=x:ya=y
2020 GOSUB 2510:GOSUB 450:IF back THEN G
OSUB 2600:GOTO 3070
2030 xb=x:yb=y:x=xa:y=ya
2040 GOSUB 2620:GOSUB 450:GOSUB 2590:IF
back THEN 3070
2050 xz=ABS(xa-xb):yz=ABS(ya-y):IF xz=0
THEN 2000
```

Listing. »CPC-Paint« — ein tolles Malprogramm (Fortsetzung)

Speichern Sie wohl...

für den Schneider CPC 464

5,25″ ..... 1,4 MB ..... CP/M 2.2 ..... VDOS 2.0

vortex Floppy-Disk-Station F1

jetzt mit VDOS 2.0: relative Dateien, Tracer, Disassembler, Assembler, BASIC-Befehlserweiterungen

Leistungen

— Ein (wahlweise zwei; von Anfang an, oder nachrüstbar) 5,25″-Slimline, 80 Track, DS/DD **6138 BASF**-Laufwerk der modernsten Technologie mit 708 KB (1,4 MB), formatierter Speicherkapaziät, 4 msec. Steprate, IBM 34-Formate.
— CP/M 2.2-Betriebssystem und Systemutilities
— Erweitertes BASIC-stand-alone-Diskettenbetriebssystem **VDOS 2.0**
— Ohne Soft- oder Hardwareänderungen kann ein Schneider 3″-Laufwerk über ein Adapterkabel angeschlossen werden. Softwarekonvertierung von 5,25″ auf 3″ und umgekehrt: kein Problem.

Preise

— F1/S Floppy-Disk-Station mit Controller und Laufwerk inkl. CP/M 2.2, VDOS und Handbuch **1198,— DM (unverbindliche Preisempfehlung)**
— F1/D Floppy-Disk-Station mit Controller und zwei Laufwerken inkl. CP/M 2.2, VDOS und Handbuch **1698,— DM (unverbindliche Preisempfehlung)**
— A1-S Aufrüstkit bestehend aus BASF-Laufwerk 6138 und Einbauanleitg. **500,— DM (unverbindliche Preisempfehlung)**
— 5,25″-Zweitlaufwerk F1-Z + Programm SPARA **698,— DM (unverbindliche Preisempfehlung)**
— Aufrüstkit A1-Z bestehend aus Controller, CP/M-Lizenz und Dienstprogrammen sowie Handbuch **548,— DM (unverbindliche Preisempfehlung)**

5,25″-Zweitlaufwerk F1-Z

Das Laufwerk F1-Z kann als Zweitlaufwerk an die Schneider 3″-Diskettenstation DDI-1 angeschlossen werden und hat dieselbe Speicherkapazität wie das 3″-Laufwerk. Es ist identisch mit der Station F1-S jedoch ohne Controller und ohne CP/M.

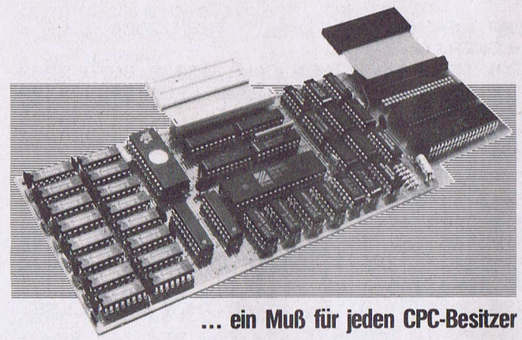
Das mitgelieferte Programm SPARA erlaubt Ihnen das Lesen und Beschreiben von Disketten gängiger CP/M-Systeme, welche Ihre Disketten einseitig mit 40 Spuren verwalten. (Diese Einschränkung bedingt der Controller der Schneider DDI-1.)

Erwacht jedoch Ihr Interesse an 708 KB oder 1,4 MB, so können Sie Ihre F1-Z problemlos zur F1-S oder F1-D aufrüsten.

64 KB bis 512 KB RAM-Erweiterung ... Druckerpuffer ... RAM-Floppy

vortex RAM-Erweiterung SP64 ...

— **jetzt endlich läuft jedes Standard-CP/M-Programm** (z.B. Wordstar, dBase, Multiplan)
— voll unter BASIC und CP/M einsetzbar
— das Betriebssystem der Karte (im ROM) ist nahtlos ins CPC-Betriebssystem eingebaut
— Die Benutzer des 3″-Schneider-Laufwerkes kommen durch den eingebauten ROM in den Genuß der VDOS 2.0 Features
— einfacher Einbau der Karte: kein Löten



Preise: SP 64/M, 64-KB-RAM-Erweiterung ohne ROM, ohne Bus-Puffer
nicht aufrüstbar **138,— DM (unverbindliche Preisempfehlung)**
SP 64, 64-KB-RAM-Erweiterung mit ROM, mit Bus-Puffer
aufrüstbar bis 512 KB **275,— DM (unverbindliche Preisempfehlung)**

... ein Muß für jeden CPC-Besitzer

Sie erhalten unsere Produkte:
— in allen Karstadt- und Horten-Computercentern
— in den technischen Kaufhäusern Phora und Brinkmann

Bei Bezugsproblemen rufen Sie uns bitte an.

Fordern Sie unser kostenloses Informationsmaterial an.

Die Programme **Wordstar, dBase und Multiplan** erhalten Sie zu sensationellen Preisen beim **M&T Software Verlag.** Alle Programme sind auf unsere RAM-Erweiterung abgestimmt und laufen mit ihr uneingeschränkt.

Mit jedem unserer Produkte erhalten Sie den **vortex Service-Paß.** Mit diesem Paß garantieren wir Ihnen einen kostenfreien Anspruch auf alle Neuerungen und eventuelle Verbesserungen unserer Betriebssystemsoftware. Für soft- und hardwaretechnische Fragen im Zusammenhang mit unseren Produkten haben wir eine **User-Sprechstunde** eingerichtet. Montags und Donnerstags von 18 Uhr - 21 Uhr stehen wir Ihnen telefonisch zur Verfügung.

CP/M 2.2 ist ein eingetragenes Warenzeichen der Firma Digital Research · VDOS und vortex sind eingetragene Warenzeichen der Firma vortex GmbH · Wordstar ist ein eingetragenes Warenzeichen der Firma Micro Pro. · dBase ist ein eingetragenes Warenzeichen der Firma Ashton Tate. · Multiplan ist eingetragenes Warenzeichen der Firma Microsoft.

vortex

7106 Neuenstadt, Klingenberg 13 — Tel.: Abt. Marketing 07139/2160, Telex 728915 — Tel.: Abt. Software 07117/775576

```
2060 IF yz-100>xz THEN GOSUB 2140:GOTO 2
080
2070 i1=PI/(yz*1.6):i2=2*PI:i3=0
2080 PLOT xa+xz*SIN(i2),ya+yz*COS(i2),fa
2090 DRAW xa+xz*SIN(i3),ya+yz*COS(i3),fa
2100 i2=i2-i1:i3=i3+i1
2110 IF xz>=yz-100 THEN IF i3>PI THEN xz
=xa:yz=ya:GOTO 2000
2120 IF yz-100>xz THEN IF i2<PI/2 THEN x
z=xa:yz=ya:GOTO 2000
2130 GOTO 2080
2140 i1=PI/(xz*3.2):i2=PI+PI/2:i3=PI+PI/
2
2150 RETURN
2160 REM spray
2170 GOSUB 2700:IF back THEN 3070 ELSE C
ALL 43000
2180 GOSUB 3030:PRINT#6:PRINT#6," Number
 of colours":PRINT#6,"    (min 2, max 4)"
;CHR$(143)
2190 a$=INKEY$:IF VAL(a$)>=2 AND VAL(a$)
<=4 THEN b=VAL(a$):GOTO 2210
2200 GOTO 2190
2210 CALL 43020
2220 xx=MAX(xa,xz):yy=MAX(ya,yz):xy=MIN(
xa,xz):yx=MIN(ya,yz)
2230 FOR y=yx TO yy STEP 2:FOR x=xy TO x
x STEP 2
2240 a=RND*(b-1):PLOT x,y,a
2250 NEXT:NEXT
2260 GOTO 2170
2270 REM hard
2280 CALL &A400
2290 GOSUB 3030:PRINT#6:PRINT#6,"  Do yo
u want":PRINT#6,"     another copy (y/n)
?"
2300 a$=INKEY$:IF a$="" THEN 2300 ELSE a
$=LOWER$(a$)
2310 IF a$="y" THEN CALL 43020:GOTO 2280
2320 IF a$<>"n" THEN 2300 ELSE CALL 4302
0:GOTO 220
2330 REM  colour
2340 LOCATE#5,1,1:CALL 43000
2350 FOR i=0 TO 3:PAPER #5,i:PRINT#5,SPA
CE$(5);:NEXT
2360 PAPER#5,0:PEN#5,1
2370 FOR i=5 TO 20 STEP 5:LOCATE #5,1,i:
PRINT#5," ":NEXT:LOCATE #5,1,5*fa+5:PRIN
T#5,CHR$(244)
2380 jo%=JOY(0):IF jo%=0 THEN 2380 ELSE
FOR i=0 TO 50:NEXT
2390 FOR i=1 TO 50:NEXT
2400 IF jo%=16 THEN CALL 43020:BORDER fb
(fa):GOTO 220
2410 IF jo%=1 THEN fa=fa-1:IF fa<0 THEN
fa=0:GOTO 2370 ELSE GOTO 2370
2420 IF jo%=2 THEN fa=fa+1:IF fa>3 THEN
fa=3:GOTO 2370 ELSE GOTO 2370
2430 IF jo%=4 THEN fb(fa)=fb(fa)-1:IF fb
(fa)<0 THEN fb(fa)=0
2440 IF jo%=8 THEN fb(fa)=fb(fa)+1:IF fb
(fa)>26 THEN fb(fa)=26
2450 INK fa,fb(fa):GOTO 2380
2460 INK fa,fb(fa):GOTO 2380
2470 fa=fa-1:IF fa<0 THEN fa=0
2480 GOTO 2370
2490 fa=fa+1:IF fa>3 THEN fa=3
2500 GOTO 2370
2510 REM  punkt 1*******
2520 x1=TEST(xa,ya):x2=TESTR(2,2):x3=TES
TR(0,-4):x4=TESTR(-4,0):x5=TESTR(0,4)
2530 PLOT xa,ya,fa:IF x1=fa THEN PLOT xa
,ya,(fa+1) MOD 4
2540 PLOT xa+2,ya+2,fa:IF x2=fa THEN PLO
T xa+2,ya+2,(fa+1) MOD 4
2550 PLOT xa+2,ya-2,fa:IF x3=fa THEN PLO
T xa+2,ya-2,(fa+1) MOD 4
2560 PLOT xa-2,ya-2,fa:IF x4=fa THEN PLO
T xa-2,ya-2,(fa+1) MOD 4
2570 PLOT xa-2,ya+2,fa:IF x5=fa THEN PLO
T xa-2,ya+2,(fa+1) MOD 4
2580 RETURN
2590 PLOT xb,yb,x6:PLOT xb+2,yb+2,x7:PLO
T xb+2,yb-2,x8:PLOT xb-2,yb-2,x9:PLOT xb
-2,yb+2,x10
2600 PLOT xa,ya,x1:PLOT xa+2,ya+2,x2:PLO
T xa+2,ya-2,x3:PLOT xa-2,ya-2,x4:PLOT xa
-2,ya+2,x5
2610 RETURN
2620 REM  punkt 2*******
2630 x6=TEST(xb,yb):x7=TESTR(2,2):x8=TES
TR(0,-4):x9=TESTR(-4,0):x10=TESTR(0,4)
2640 PLOT xb,yb,fa:IF x6=fa THEN PLOT xb
,yb,(fa+1) MOD 4
2650 PLOT xb+2,yb+2,fa:IF x7=fa THEN PLO
T xb+2,yb+2,(fa+1) MOD 4
2660 PLOT xb+2,yb-2,fa:IF x8=fa THEN PLO
T xb+2,yb-2,(fa+1) MOD 4
2670 PLOT xb-2,yb-2,fa:IF x9=fa THEN PLO
T xb-2,yb-2,(fa+1) MOD 4
2680 PLOT xb-2,yb+2,fa:IF x10=fa THEN PL
OT xb-2,yb+2,(fa+1) MOD 4
2690 RETURN
2700 REM *****
2710 GOSUB 450:IF back THEN RETURN
2720 xa=xz:ya=yz
2730 GOSUB 2510:GOSUB 450:GOSUB 2600
2740 RETURN
2750 REM  mirror
2760 REM
2770 GOSUB 450:IF back THEN RETURN
2780 xa=x:ya=y
2790 GOSUB 2510:GOSUB 450:IF back THEN G
OSUB 2600:RETURN
2800 xb=x:yb=y
2810 GOSUB 2620:GOSUB 450:IF back THEN G
OSUB 2590:RETURN
2820 xc=MAX(xa,xb):yc=MAX(ya,yb):xc1=MIN
(xa,xb):yc1=MIN(ya,yb)
2830 xc2=x:yc2=y:GOSUB 2590
2840 GOSUB 3030:PRINT#6,"Delete Backgrou
nd (y/n) ?"
2850 a$=INKEY$:a$=LOWER$(a$):IF a$="n" O
R a$="y" THEN GOTO 2860 ELSE GOTO 2850
2860 IF a$="n" THEN flg=1 ELSE flg=0
2870 PRINT#6,"Magnify Part       (y/n) ?"
2880 a$=INKEY$:a$=LOWER$(a$):IF a$="y" O
R a$="n" THEN GOTO 2890 ELSE GOTO 2880
2890 IF a$="y" THEN flg1=1 ELSE flg1=0
2900 PRINT#6,"Reflect Part       (y/n) ?"
2910 a$=INKEY$:a$=LOWER$(a$):IF a$="y" O
R a$="n" THEN GOTO 2920 ELSE GOTO 2910
2920 IF a$="y" THEN xc=MIN(xa,xb):xc1=MA
X(xa,xb):st=-2 ELSE st=2
2930 CALL 43020:IF flg1<>0 THEN GOTO 298
0
2940 m1=0:m2=0:FOR m=xc1 TO xc STEP st:F
OR m3=yc1 TO yc STEP 2
2950 farbe=TEST(m,m3):IF flg=0 THEN PLOT
 xc2+m1,yc2+m2,farbe ELSE IF TEST(xc2+m1
,yc2+m2)=0 THEN PLOT xc2+m1,yc2+m2,farbe
2960 m2=m2+2:NEXT:m1=m1+2:m2=0:NEXT
```



```
2970 GOTO 2760
2980 m1=0:m2=0:FOR m=xc1 TO xc STEP st:F
OR m3=yc1 TO yc STEP 2
2990 farbe1=TEST(m,m3):IF flg=0 THEN PLO
T xc2+m1,yc2+m2,farbe1:PLOT xc2+m1,yc2+m
2+2,farbe1:PLOT xc2+m1+2,yc2+m2,farbe1:P
LOT xc2+m1+2,yc2+m2+2,farbe1
3000 IF flg=1 AND TEST(xc2+m1,yc2+m2)=0
THEN PLOT xc2+m1,yc2+m2,farbe1:PLOT xc2+
m1,yc2+m2+2,farbe1:PLOT xc2+m1+2,yc2+m2,
farbe1:PLOT xc2+m1+2,yc2+m2+2,farbe1
3010 m2=m2+4:NEXT:m1=m1+4:m2=0:NEXT
3020 GOTO 2760
3030 CALL 43000:CLS#6
3040 LOCATE 11,17:PRINT CHR$(150)STRING$
(26,CHR$(154))CHR$(156)
3050 FOR i=18 TO 21:LOCATE 11,i:PRINT CH
R$(149):LOCATE 38,i:PRINT CHR$(149)::NEX
T:LOCATE 11,22:PRINT CHR$(147)STRING$(26
,CHR$(154))CHR$(153)
3060 RETURN
3070 CALL 43000:WINDOW#2,30,38,12,22
3080 LOCATE 29,11:PEN 1:PRINT CHR$(150)S
TRING$(9,CHR$(154))CHR$(156)
3090 FOR i=12 TO 22:LOCATE 29,i:PRINT CH
R$(149):LOCATE 39,i:PRINT CHR$(149)::NEX
T
3100 LOCATE 29,23:PRINT CHR$(147)STRING$
(9,CHR$(154))CHR$(153)
3110 CLS#2:RESTORE 3110:FOR i=1 TO 11:RE
AD w$
3120 LOCATE 1,i:IF i=me1 THEN PRINT#2,CH
R$(24);w$;CHR$(24) ELSE PRINT#2,w$
3130 NEXT
3140 DATA Line,Lines,Streams,Frame,Box,T
riangle,Circle,Disk,Ellipse,Spray,Text
3150 jo%=JOY(0):IF jo%=0 THEN 3150
3160 IF jo%=16 THEN CALL 43020:GOTO 220
3170 me1=me1-(1 AND jo%=1)+(1 AND jo%=2)
:IF me1<=1 THEN me1=1 ELSE IF me1>=11 TH
EN me1=11
3180 IF jo%=4 OR jo%=8 THEN 3330
3190 GOTO 3110
3200 CALL 43000:WINDOW#2,30,38,5,11
3210 LOCATE 29,4:PEN 1:PRINT CHR$(150)ST
RING$(9,CHR$(154))CHR$(156)
3220 FOR i=5 TO 11:LOCATE 29,i:PRINT CHR
$(149):LOCATE 39,i:PRINT CHR$(149)::NEXT
3230 LOCATE 29,12:PRINT CHR$(147)STRING$
(9,CHR$(154))CHR$(153)
3240 CLS#2:RESTORE 3240:FOR i=1 TO 7:REA
D w$
3250 LOCATE 1,i:IF i=me2 THEN PRINT#2,CH
R$(24);w$;CHR$(24) ELSE PRINT#2,w$
3260 NEXT
3270 DATA Load,Save,Fill,Zoom,Mirror,Har
dcopy,Erase
3280 jo%=JOY(0):IF jo%=0 THEN 3280
3290 IF jo%=16 THEN CALL 43020:GOTO 220
3300 me2=me2-(1 AND jo%=1)+(1 AND jo%=2)
:IF me2<=1 THEN me2=1 ELSE IF me2>=7 THE
N me2=7
3310 IF jo%=4 OR jo%=8 THEN 3340
3320 GOTO 3240
3330 CALL 43020:x=xz:y=yz:ON me1 GOTO 64
0,680,740,800,860,930,1060,1190,1980,216
0,1470
3340 CALL 43020:x=xz:y=yz:ON me2 GOTO 13
30,1400,330,1660,1320,2270,1550
3350 REM ****************************
3360 MEMORY &5FFF:RESTORE 3360
3370 FOR i=43000 TO 43011
3380 READ wert
3390 POKE i,wert:NEXT
3400 FOR i=43020 TO 43031
3410 READ wert
3420 POKE i,wert:NEXT
3430 DATA &01,&ff,&3f,&11,&00,&60,&21,&0
0,&c0,&ed,&b0,&c9
3440 DATA &01,&ff,&3f,&11,&00,&c0,&21,&0
0,&60,&ed,&b0,&c9
3450 CLEAR:DEFINT a-h,j-o,q-z:pot=10:ver
t=1:x=320:y=200:DIM fb(3),x(40),y(40),fa
r(40,20):xz=x:yz=y:me1=1:me2=1:nom=32
3460 fb(0)=0:fb(1)=2:fb(2)=18:fb(3)=6:fa
=1
3470 MODE 1:INK 0,0:INK 1,2:INK 2,18:INK
 3,6:PEN 1:BORDER fb(fa)
3480 WINDOW #1,3,40,1,1
3490 WINDOW#4,2,39,3,22
3500 WINDOW#5,40,40,3,23
3510 WINDOW#6,12,37,18,21
3520 PRINT CHR$(150)STRING$(38,CHR$(154)
)CHR$(156)
3530 FOR i=2 TO 24:LOCATE 1,i:PRINT CHR$
(149):LOCATE 40,i:PRINT CHR$(149);:NEXT:
PRINT CHR$(147)STRING$(38,CHR$(154))CHR$
(153);
3540 KEY 0,"call 43020:goto 40"+CHR$(13)
3550 ENT 1,75,-1,2
3560 ENV 1,100,-1,2
3570 RESTORE 3570:REM turbo
3580 FOR i=1 TO 9:READ we:POKE 42500+i,w
e:NEXT
3590 DATA 33,93,0,62,10,205,104,188,201
3600 POKE 42502,95:CALL 42501
3610 ORIGIN 0,0,10,630,390,10
3620 REM hardcopy von data becker um &40
0 verschoben
3630 FOR i=&A400 TO &A4BF
3640 READ byte: POKE i,byte:s=s+byte:NEX
T
3650 DATA &cd,&ba,&bb,&cd,&e7,&bb,&32,&b
d,&a4,&cd,&6c,&a4,&21,&8f,&01,&22
3660 DATA &be,&a4,&11,&00,&00,&3e,&07,&3
2,&c0,&a4,&cd,&7c,&a4,&0e,&00,&3a
3670 DATA &c0,&a4,&47,&e5,&d5,&c5,&cd,&f
0,&bb,&c1,&d1,&21,&bd,&a4,&be,&e1
3680 DATA &37,&20,&01,&a7,&cb,&11,&2b,&2
b,&10,&e9,&cd,&af,&a4,&79,&cd,&a6
3690 DATA &a4,&13,&e5,&21,&7f,&02,&37,&e
d,&52,&e1,&38,&05,&2a,&be,&a4,&18
3700 DATA &cc,&23,&7c,&b5,&c8,&2b,&11,&0
0,&00,&22,&be,&a4,&3e,&07,&bd,&20
3710 DATA &b9,&7c,&b4,&20,&b5,&3e,&04,&3
2,&c0,&a4,&18,&ae,&3e,&1b,&cd,&a6
3720 DATA &a4,&3e,&41,&cd,&a6,&a4,&3e,&0
7,&cd,&a6,&a4,&c9,&e5,&3e,&42,&cd
3730 DATA &1e,&bb,&e1,&28,&02,&e1,&c9,&3
e,&0d,&cd,&a6,&a4,&3e,&0a,&cd,&a6
3740 DATA &a4,&3e,&1b,&cd,&a6,&a4,&3e,&4
c,&cd,&a6,&a4,&3e,&7f,&cd,&a6,&a4
3750 DATA &3e,&02,&cd,&a6,&a4,&c9,&cd,&2
e,&bd,&38,&fb,&cd,&2b,&bd,&c9,&3a
3760 DATA &c0,&a4,&fe,&07,&c8,&af,&cb,&1
1,&cb,&11,&cb,&11,&c9,&00,&00,&00
3770 PRINT#8,CHR$(27);"3";CHR$(10):REM z
eilenvorschub
3780 GOTO 1330
```

Listing. »CPC-Paint« — ein tolles Malprogramm (Schluß)

Bildschirm-spielereien

Nicht immer muß man »nützliche« Programme abtippen. Auch Spielereien am Computer üben einen faszinierenden Reiz aus. Dieses Listing lädt Sie zum Träumen ein.

Grafik ist schon immer ein Reizwort für fast jeden Programmierer gewesen. Doch es ist gar nicht so leicht gutaussehende Bilder auf den Monitor zu zaubern. Das Programm »Bildschirmspielereien« zeigt, wie man ständige Bewegung interessant gestalten kann. Durch Veränderung der entsprechenden Register kann man alle Farben durchlaufen lassen. Und das auch in Basic mit hoher Geschwindigkeit.

Das Programm ist in zwei Teile gegliedert, die in verschiedenen Modi laufen. Teil 1 nützt im Modus 2 die hochauflösende Grafik, Teil 2 die Farbenvielzahl des Modus 0. Durch verschiedene Helligkeiten werden erstaunliche 3-D-Effekte erzielt. Allerdings kommt dies nur mit einem grünen Monitor richtig zur Geltung.

(Günter Thurn)

```
10 REM Graphik-Demonstrationsprogramm
20 REM (c) 1985 Guenter Thurn
30 REM     Linggstr. 10
40 REM     8500 Nuernberg 70
50 REM Programmteil 1
60 MODE 2
70 DEG
80 FOR b=1 TO 6
90 CLS
100 INK 0,27:INK 1,0:BORDER 27
110 FOR a=1 TO 360
120 ORIGIN 320,200
130 ON b GOSUB 270,280,290,300,310,320
140 NEXT
150 FOR c=1 TO 500:NEXT
160 CLS
170 INK 1,27:INK 0,0:BORDER 0
180 ORIGIN 0,0
190 e=390:f=630
200 FOR d=10 TO 630 STEP 5
210 ON b GOSUB 330,340,350,360,370,380
220 e=e-3:f=f-12
230 NEXT
240 NEXT
250 GOTO 400
260 REM Formeln Programmteil 1
270 DRAW 310*SIN(a),190*COS(a),1:RETURN
280 DRAW 310*SIN(a),190*COS(a)*SQR(COS(a
)^2):RETURN
290 DRAW 310*SIN(a)*SQR(SIN(a)^2),190*CO
S(a)*SQR(COS(a)^2):RETURN
300 DRAW 310*SIN(a)*SQR(SIN(a+45)^2),190
*COS(a)*SQR(COS(a+112.5)^2):RETURN
310 DRAW 190*COS(a)*SQR(COS(a+135)^2),19
0*SIN(a)*SQR(SIN(a+45)^2):RETURN
320 DRAW 310*(SIN(a)*SQR(SIN(a+45)^2))^2
,190*COS(a)*SQR(COS(a+135)^2):RETURN
330 PLOT d,d:DRAW d,e:DRAW f,e:DRAW f,d:
DRAW d,d:RETURN
340 PLOT d,d:DRAW d,e:DRAW f,d:DRAW d,d:
RETURN
350 PLOT d,d:DRAW d,e:DRAW e,e:DRAW f,d:
DRAW f,e:DRAW d,d:RETURN
360 PLOT d,d:DRAW d,e:DRAW f,d:DRAW f,e:
DRAW d,d:RETURN
370 PLOT d,d:DRAW d,e:DRAW e-150,e:DRAW
f,d:DRAW e,f:DRAW d,d:RETURN
380 PLOT d,d:DRAW d,e:DRAW f,e:DRAW f,d:
DRAW d,f:RETURN
390 REM Programmteil 2
400 MODE 0
410 FOR b=1 TO 6
420 FOR a=0 TO 15
430 INK a,a
440 PLOT 120*(b-1)+4*a-200,400,a:DRAW 20
0+120*(b-1)+4*a,0,a
450 NEXT
460 FOR a=15 TO 1 STEP -1
470 PLOT 200+120*b-4*a,400,a:DRAW 120*b-
4*a-200,0,a
480 NEXT
490 NEXT
500 GOSUB 780
510 FOR b=0 TO 6
520 FOR a=0 TO 15
530 PLOT b*124+60-4*a,1,a:DRAW b*124+60-
4*a,400,a
540 PLOT 1,b*62+30-2*a,a:DRAW 640,b*62+3
0-2*a,a
550 PLOT b*124+60+4*a,1,a:DRAW b*124+60+
4*a,400,a
560 PLOT 1,b*62+30+2*a,a:DRAW 640,b*62+3
0+2*a,a
570 NEXT
580 NEXT
590 GOSUB 780
600 FOR a=0 TO 100
610 PLOT 320-4*a,1,g:DRAW 320-4*a,400,g

620 PLOT 1,200-2*a,g:DRAW 640,200-2*a,g
630 PLOT 320+4*a,1,g:DRAW 320+4*a,400,g
640 PLOT 1,200+2*a,g:DRAW 640,200+2*a,g
650 g=g+1:IF g=16 THEN g=0
660 NEXT
670 GOSUB 780
680 MOVE 320,282
690 FOR a=0 TO 15
700 FOR b=1 TO 360 STEP 50
710 DEG
720 DRAW 320+(80+4*a)*SIN(b),202+(80+2*a
)*COS(b),15-a
730 NEXT
740 NEXT
750 GOSUB 780
760 RUN
770 REM Farbeffekte
780 FOR c=1 TO 6
790 FOR a=0 TO 15:INK a,16-a:FOR d=1 TO
25:NEXT:BORDER 16-a:NEXT
800 FOR a=0 TO 15:INK a,a:FOR d=1 TO 25:
NEXT:BORDER a:NEXT
810 NEXT
820 RETURN
```

Listing. »Bildschirmspielereien« mit dem Computer

Musik und Sound selbst programmiert

Auf dem Schneider-Computer Musik zu programmieren ist nicht schwer. Man muß nur wissen wie. Dieser Artikel gibt Ihnen Hinweise und viele Tips, wie Sie schöne Klänge erzeugen können.

Wer sich mit dem Schneider beschäftigt, redet über vieles, angefangen von Ausbaubarkeit über das Betriebssystem bis hin zu ROM-Erweiterungen und Zeichensatz. Ein Thema bleibt dabei jedoch meist auf der Strecke: der Sound. Dabei sind die musikalischen Fähigkeiten mit einem schnöden »PRINT CHR$(7)« noch lange nicht erschöpft. Schon unter Basic läßt sich mit Hilfe von Warteschlangen und einer ausgeklügelten Unterbrechungstechnik durch Geräusche viel am Ablauf von Spielen und Simulationsprogrammen verbessern. Aber auch das Arbeiten mit Datenverwaltungsprogrammen und Textroutinen gestaltet sich erheblich angenehmer, wenn zum Beispiel nach dem Einlesen oder Speichern einer Datei eine akustische Endkennung ausgegeben wird. Daneben kann man die Musik beim Schneider natürlich auch noch als Selbstzweck betreiben.

Wie das geht, welche Grundlagen bestehen und mit welchen Tips, Tricks und Einblicken in die Hardware man noch etwas mehr aus dem Computer herausholen kann, darum geht es in diesem Artikel. Im ersten Teil kommen wir auf die Grundlagen zu sprechen und stellen Wirkungsweise und Problematik der verschiedenen Basic-Kommandos vor. Der zweite Teil gehört dann denen, die mehr wissen wollen. Hier geht es um den internen Ablauf von Soundsteuerung und Klangausgabe und um die Nutzung von Betriebssystemroutinen, mit denen eine ganze Reihe von Verbesserungen beim Klang möglich sind.

Betrachten wir zunächst einmal die Hardware. Der Schneider gibt mit Hilfe des Soundchips (AY 3-8912 von General Instruments) maximal drei Töne auf drei verschiedenen Ausgabeleitungen, den Kanälen, zur gleichen Zeit aus. Diese drei Kanäle sind beim CPC nun so geschaltet, daß sie die Geräusche in Stereo abgeben. Kanal A liegt dabei auf dem linken Lautsprecher, C auf dem rechten und B in der Mitte. Wenn Sie also den Kanal B ansprechen, so wird ein Klang auf beiden Lautsprechern ausgegeben. Daneben verfügt der Soundchip noch über einen Rauschgenerator, der jedem der drei Kanäle separat zugemischt werden kann. Dazu müssen an den Baustein die erforderlichen Daten für Tonhöhe, Lautstärke und natürlich auch die des Tonregisters (Kanals), für welches diese Angaben gelten sollen, weitergegeben werden. Als eine Besonderheit kann der AY3-8912 noch den Lautstärkeverlauf selbständig im Lauf der Zeit ändern, das heißt er kann nach bestimmten Mustern schwankende Lautstärken erzeugen.

Das dominierende Kommando für die Erzeugung von Musik ist der SOUND-Befehl. Er stellt das komplexeste Kommando im gesamten Basic-Befehlssatz des Schneider dar. SOUND verfügt über insgesamt sieben Parameter. Der erste ist ein allgemeiner Steuerwert, auf den wir gleich zu sprechen kommen. Es folgen die Angaben zur Tonhöhe, zur Länge des Tones und zur Grundlautstärke. Mit zwei zusätzlichen Parametern ist es möglich, die Lautstärke und Tonhöhe im Zeitablauf zu variieren. Abschließend wird noch darüber entschieden, ob der Rauschgenerator zugeschaltet werden soll oder nicht. Bevor wir uns mit dem Befehl und der Wirkung der einzelnen Parameter etwas näher beschäftigen, wollen wir ein paar Worte über das Konzept der Sound-Steuerung verlieren.

Softwaregesteuerte Töne

Der Ablauf der Klangwiedergabe beim Schneider ist nicht nur Software-gesteuert, was ja selbstverständlich ist, sondern auch Software-orientiert. Die Befehle werden also benutzt, um einzelne Kenngrößen für die Hardware-Bausteine zu setzen. Die Hardware führt aus, was die Software befiehlt.

Die Basis des Sound-Befehls bilden drei Warteschlangen. Sie fungieren als Pufferspeicher für die drei Kanäle. Jede Warteschlange kann dabei maximal vier Kommandos enthalten: Eins, das gerade abgearbeitet wird und drei weitere, die auf der Warteliste stehen. Grundsätzlich werden die Anweisungen für den Sound nacheinander behandelt. Dazu existiert ein spezieller Unterbrechungstakt von 100 Hz. In diesem Rhythmus — hundertmal pro Sekunde — überprüft der CPC, ob ein Sound-Befehl abgeschlossen ist oder eine Änderung vorgenommen werden muß. Im ersten Fall wird der alte Sound »vorne« aus der Warteschlange »ausgestoßen« und der nächste Klang bearbeitet. Ferner wird die Tonhöhe oder -lautstärke geändert und der Computer geht bis zum nächsten Takt wieder in Wartestellung — oder nimmt andere Systemfunktionen wahr.

Im Speicherbereich oberhalb des Basic ist ein bestimmter Platz für die Ablage der verschiedenen Werte reserviert. Kommt ein neues Sound-Kommando hinzu, wird dieses hinten angefügt, vorausgesetzt, es ist noch ausreichend Platz vorhanden. Die Maximalzahl von vier abspeicherbaren Sound-Kommandos darf aber nicht überschritten sein. Gleichzeitig arbeitet der Prozessor mit dem Unterbrechungstakt das erste Kommando, wie oben beschrieben, ab. Ist schließlich ein Sound-Kommando bearbeitet, so werden die Parameter aller nachfolgenden Kommandos um eine Position nach »vorn« gerückt. Dadurch lagern die Angaben der alten Nummer 2 jetzt auf der Position der alten Nummer 1 und die zweite Sound-Anweisung wird ausgeführt. Da in jeder Schlange Töne mit unterschiedlicher Länge gespeichert werden können, ist es dabei auch möglich, daß Kanal A noch den ersten Sound-Befehl bearbeitet, während zum Beispiel Kanal C schon das dritte Kommando behandelt hat.

Wir haben weiter oben festgestellt, daß die Sound-Befehle normalerweise in Folge ausgeführt werden. Dies muß jedoch nicht immer so sein. Der CPC verfügt nämlich noch über zwei weitere Modi im Bereich der Tonerzeugung. Zum ersten ist es möglich, eine Anweisung für den Sound im »Hold«-Mode einzugeben. Der Klang bleibt dann gespeichert, wird aber nicht ausgeführt. Man kann dann später, wenn

der Klang »erscheinen« soll, diesen mit RELEASE aufrufen. So ist es zum Beispiel möglich einen komplexen Klangeffekt, der in einem Spiel nur beim Eintritt einer speziellen Bedingung, dann aber möglichst schnell, auftreten soll, bereits in die Schlange zu speichern, um ihn bei Bedarf sofort zur Verfügung zu haben.

Mit der zweiten Betriebsart (Flush) ist es möglich, die normale Reihenfolge zu ändern. Man kann nämlich ein neues Sound-Kommando mit Priorität ausführen. Dieses unterbricht dann den normalen Musikablauf und wird sofort ausgeführt. Dazu muß es sofort an der ersten Stelle der Warteschlange stehen. In diesem Fall werden die möglicherweise noch folgenden weiteren Kommandos nicht mehr ausgeführt. Die Warteschlange wird also faktisch gelöscht. Ein Beispiel: Bei einem Action-Spiel soll bei einem Fehler die fröhliche Dauermelodie durch den obligatorischen Sterbe-Song abgelöst werden. Flush bricht dann ab und gibt die Todesmelodie aus.

Eine weitere Spezialität des Schneider betrifft die Synchronisation von Klängen. So kann man die Ausführung einer Tonfolge in der Warteschlange so lange anhalten, bis in einer weiteren Warteschlange (oder auch den beiden anderen) ebenfalls die benötigten Daten für den nächsten Klang zur Verfügung stehen. Es wird also eine Art bedingtes Hold ausgeführt. Diese Betriebsart wird als Rendezvous bezeichnet. Ein Sound synchronisiert dabei den anderen, das heißt zwei Klangfolgen werden nun gleichzeitig abgespielt. Diese Verkettung hat den Vorteil, daß man speziell bei mehrstimmigen Musikstücken nicht immer noch darauf zu achten braucht, ob Töne exakt zum selben Zeitpunkt ausgegeben werden. Führt man zum Beispiel zwischen zwei Sound-Befehlen, die verschiedene Kanäle betreffen, noch weitere Anweisungen aus, so würden sich durch die Bearbeitungszeit für diese Befehle die Anfangszeiten der beiden Soundkommandos verschieben.

Nach soviel Theorie sind ein paar Experimente dringend notwendig. Wenn Sie dabei diesen natürlich sehr stark schematisierten Ablauf mit seinen Spezialitäten im Hinterkopf behalten, werden Ihnen die Besonderheiten des Sound-Kommandos relativ schnell klar werden.

Wir wollen unsere Untersuchungen mit ein paar ganz einfachen Befehlen beginnen. Wenn Sie ein Kabel zum Anschluß des Computers an die Stereoanlage haben, so erhalten Sie die Tonausgabe auf verschiedenen Kanälen Ihrer Anlage im Wohnzimmer; ansonsten eben nur über den eingebauten Lautsprecher des Computers.

»SOUND 1,100,100« läßt einen Piepser in mittlerer Tonhöhe auf dem linken Kanal erklingen. Wenn Sie nun die 1 durch eine 4 ersetzen, wechselt der Klang den Lautsprecher. Jetzt piepst es von rechts. Mit diesen ersten beiden Experimenten haben Sie schon die wichtigsten drei Parameter von SOUND etwas näher unter die Lupe genommen. Diese müssen unbedingt angegeben werden, um den Ablauf des Sound-Kommandos zu ermöglichen. Beginnen wir von hinten. Parameter Nummer 3 erhält die Tonlänge. Sie ist in hundertstel Sekunden angegeben. Die minimale Periode entspricht also genau einem Unterbrechungstaktzyklus. Wenn Sie eine Viertelnote spielen wollen, wäre hier 25 einzusetzen; bei einer halben 50. Parameter 2 definiert die Tonhöhe. Die Angabe erfolgt hier in Einheiten von 125000 Hertz. Diese Frequenz stellt einen Grundtakt dar, aus dem alle anderen Töne abgeleitet werden. Gibt man zum Beispiel 284 ein, so erhält man eine Frequenz, deren Periode 284mal so lang ist wie die Grundeinheit. Dies entspricht ziemlich genau dem Wert des international genormten A mit 440 Hertz. Die Frequenz eines Tones läßt sich errechnen, indem man 125000 durch den Wert von oben teilt. Umgekehrt erhält man den einzusetzenden Wert, indem man 125000 durch die gegebene Frequenz dividiert. Geeignete Werte für die verschiedenen Noten und Oktaven geben die Tabellen im Anhang IV des Handbuches an. Dabei ist zu sagen, daß sich jede Oktave aus der Grundoktave durch Verdoppelung oder Halbierung der einzusetzenden Werte ableiten läßt. Gibt man also statt 284, 586 ein, so erhält man ein A, welches eine Oktave tiefer liegt. Andererseits liefert ein Wert von 142 ein A in der nächsthöheren Oktave.

Als dritte Angabe bleibt nun noch Parameter 1 übrig. Er trägt die Bezeichnung Kanalstatus und enthält in verschiedenen Bits codiert alle Informationen über den angesprochenen Kanal und das Verhältnis des Klangs zu den anderen Sounds (Flush, Rendezvous). Die Bedeutung der einzelnen Bits entnehmen Sie dabei bitte der Tabelle 1.

Die unteren, rechten, niederwertigen drei Bits stehen dabei für den angesprochenen Kanal. »SOUND

Dezimal	Bit	Kommando
1	0	Sende Sound nach Kanal A (links)
2	1	Sende Sound nach Kanal B (mitte)
4	2	Sende Sound nach Kanal C (rechts)
8	3	Rendezvous mit Kanal A
16	4	Rendezvous mit Kanal B
32	5	Rendezvous mit Kanal C
64	6	Hold
128	7	Flush

Tabelle 1. Die Bits des ersten Parameters

```
10 ' ************
20 ' ** hrs v1 **
30 ' ************
40 DATA 1.1136.23
50 DATA 1.0.4
60 DATA 1.1136.48
70 DATA 1.1012.25
80 DATA 1.956.50
90 DATA 1.758.25
100 DATA 1.851.50
110 DATA 1.1136.25
120 DATA 1.1012.50
130 DATA 1.568.23
140 DATA 1.0.4
150 DATA 1.568.46
160 DATA 1.0.4
170 DATA 1.568.23
180 DATA 1.638.37
190 DATA 1.758.13
200 DATA 1.851.25
210 DATA 1.758.75
220 DATA 1.0.2
230 DATA 1.758.48
240 DATA 1.568.23
250 DATA 1.0.4
260 DATA 1.568.46
270 DATA 1.0.4
280 DATA 1.568.23
290 DATA 1.638.23
300 DATA 1.0.4
310 DATA 1.638.23
320 DATA 1.758.25
330 DATA 1.851.23
340 DATA 1.0.3
350 DATA 1.851.12
360 DATA 1.0.3
370 DATA 1.851.23
380 DATA 1.1136.25
390 DATA 1.956.50
400 DATA 1.1136.23
410 DATA 1.0.3
420 DATA 1.1136.48
430 DATA 1.0.3
440 DATA 1.1136.23
450 DATA 1.1204.48
460 DATA 1.0.4
470 DATA 1.1204.23
480 DATA 1.1136.100
490 DATA 0.0.0
500 ON SQ(1) GOSUB 520
510 GOTO 500
520 READ k.t.1:IF k=0
THEN END ELSE SOUND
k.t/4.1*1.5:RETURN
```

Listing 1. »House of the Rising Sun«

1,284,100« gibt also ein ganzes A auf Kanal A aus, »SOUND 1,284,100« auf Kanal B und so weiter. Es ist jedoch auch möglich mit einem Befehl zwei Kanäle gleichzeitig anzusprechen. Dazu müssen beide Bits gesetzt sein. In Basic erreichen wir dies, indem wir die in der ersten Spalte stehenden Dezimalwerte der einzelnen Bits addieren. Soll ein Sound parallel auf Kanal A und C auftreten, so geben wir »SOUND 5,284,100« ein. Welche Funktion haben nun die anderen Bits? Sie decodieren die verschiedenen Betriebsarten und Synchronisationen. Addieren wir 128 zum Kanalstatus, so wird der ausgegebene Klang im Flush-Modus ausgeführt. Vergleichen Sie einmal die beiden nachfolgenden Eingaben: »SOUND 1,100,200«, »SOUND 1,200,100« und »SOUND 1,100,200«, »SOUND 129,200,100«.

Im ersten Fall hören Sie zuerst zwei Sekunden lang einen Ton von 1250 Hertz. Dann wird auf 625 Hertz heruntergeschaltet. Bei Eingabe Nummer 2 beginnt es ebenso, allerdings wird nach dem zweiten Befehl sofort der neue Ton angesteuert. Wenn Sie nach dem ersten SOUND noch einen weiteren Befehl eingeben, werden Sie auch, wie oben schon gesagt, feststellen, daß dieser im zweiten Fall nicht mehr ausgeführt wird. In dem Moment, sobald der CPC den dritten Klang in die Warteschlange einreiht, wird die Flush-Markierung erkannt und damit der zugehörige Ton sofort ausgegeben. Für den zweiten Ton bleibt keine Zeit mehr übrig.

Probieren Sie nun einmal »SOUND 68,284,200«. Hier passiert nichts, jedenfalls vorläufig. Erst ein »RELEASE 4« (hier gelten ebenfalls die Dezimalwerte) löst den Klang aus. Ein Ton mit gesetztem Bit für RELEASE wird quasi unter Vorbehalt in die Warteschlange eingereiht. Es bedarf erst noch eines Aktivierungskommandos, um ihn dann wirklich zum Klingen zu bringen. Diese Art der Sound-Ansprache eignet sich besonders, wenn man komplexe Klänge, gegebenenfalls unter zusätzlicher Definition von Ton- oder Lautstärkehüllkurven (siehe unten), schnell auf eine bestimmte Bedingung hin ausgeben will. Dazu lädt man den benötigten Klang in eine freie Warteschlange. Ist die Bedingung erfüllt, so steht der Ton schnell zur Verfügung. Natürlich sind auch Kombinationen zwischen den einzelnen Effekten möglich. So kann zum Beispiel mit »SOUND 128+64+2,284,50« eine Unterbrechung der momentanen Sound-Schlange für Kanal B bei gleichzeitiger Definition eines »A« im Hold-Modus erreicht werden. Dieser Klang wird wiederum erst nach »RELEASE 2« erzeugt.

Nun aber zur Synchronisation von Klängen. Als Beispiel sollten Sie einmal das Listing 1 abtippen. Es spielt den Evergreen »House of the Rising Sun« und demonstriert eine Möglichkeit, wie man eine Melodie abspeichern kann: in DATA-Zeilen. Jede Zeile enthält dabei genau einen Ton. Die Ausgabe erfolgt auf dem linken Kanal (Kanal A). Das Listing zeigt Ihnen dabei, wie die Definition einer einfachen Grundmelodie aussehen kann. Jede DATA-Zeile beinhaltet genau drei Parameter (für Kanalstatus, Tonhöhe und -länge). Diese werden nacheinander in die Warteschlange für Kanal A eingereiht und dann ausgeführt. Im Listing 2 finden Sie die Erweiterungen, die notwendig sind, wenn Sie mit Begleitung spielen wollen. In der Melodiestimme ändern sich nur die Angaben für den Kanalstatus. Kanal A soll ja mit den anderen zusammenspielen. Um diesen Zweck zu erreichen, müssen wir dafür Sorge tragen, daß die Töne der Melodiestimme parallel zu denen der Begleitung abgearbeitet werden. So ist es zum Beispiel nicht möglich, auf den zweiten Ton einfach mit SOUND 4,758,75 einen Oberton zu addieren. Bei dieser Eingabe würden nämlich die Warteschlangen von A und C nacheinander geladen. Da dieses Laden auch unter Basic schnell abläuft, hätten wir am Ende der zweiten DATA-Zeile zwei Sounds im Kanal A zwischengespeichert und den zugehörigen Begleitklang in Kanal C. Mit dem nächsten Unterbrechungstakt würde der Prozessor also den ersten Sound im Kanal A und den ersten Sound im Kanal C abarbeiten. Sie können diesen Ablauf mit den folgenden drei Befehlen leicht nachvollziehen:

10 SOUND 1,200,25
20 SOUND 1,400,25:SOUND 4,800,25

Der tiefe Ton auf Kanal C erklingt bereits mit dem ersten (hohen) Ton in Kanal A. Bei dieser Eingabe würde also die Ausgabe des ersten Begleittones schon am Anfang des Stücks erfolgen, was ja nicht vorgesehen ist. Auch weitere Begleittöne würden in dem Moment ausgegeben, in dem sie definiert werden.

```
10 ' ************
20 ' ** hrs v1 **
30 ' ************
40 DATA 1.1136.23
50 DATA 33.1136.48.12.758.75
60 DATA 1.1012.25
70 DATA 33.956.50.12.638.75
80 DATA 1.758.25
90 DATA 33.851.50.12.676.75
100 DATA 1.1136.25
110 DATA 33.956.50.12.716.75
120 DATA 1.568.23
130 DATA 1.0.4
140 DATA 33.568.46.12.758.75
150 DATA 1.0.4
160 DATA 1.568.23
170 DATA 33.638.37.12.956.75
180 DATA 1.758.13
190 DATA 1.851.25
200 DATA 33.758.75.12.1012.75
210 DATA 1.0.2
220 DATA 33.758.48.12.1012.50
230 DATA 1.568.23
240 DATA 1.0.4
250 DATA 33.568.46.12.758.75
260 DATA 1.0.4
270 DATA 1.568.23
280 DATA 33.638.23.12.956.75
290 DATA 1.0.4
300 DATA 1.638.23
310 DATA 1.758.25
320 DATA 33.851.24.12.676.75
330 DATA 1.0.2
340 DATA 1.851.12
350 DATA 1.0.2
360 DATA 1.851.24
370 DATA 1.1136.12
380 DATA 33.956.50.12.716.75
390 DATA 1.1136.23
400 DATA 1.0.3
410 DATA 33.1136.50.12.758.75
420 DATA 1.956.12
430 DATA 1.851.13
440 DATA 33.758.48.12.1012.75
450 DATA 1.0.4
460 DATA 1.851.13
470 DATA 1.956.12
480 DATA 33.1136.100.12.758.100
490 DATA 0.0.0
500 ON SQ(1) GOSUB 520
510 GOTO 500
520 READ k.t.1:IF 1=0 THEN
END ELSE SOUND
k.t/4.1*1.5:RETURN
```

Listing 2. Mehrstimmig klingt der Evergreen besser

Sound mit Rendezvous

Die Folge wäre die schönste Disharmonie. Wir müssen die Ausgabe der Begleitung daher so lange unterdrücken, bis der zugehörige Klang (in unserem Beispiel der zweite Sound im Register A) an der Reihe ist.

Dazu muß in beiden Kanälen die Rendezvous-Bedingung hergestellt sein. Kanal A muß also nun einen Verweis erhalten, daß der zweite Ton erst gespielt werden soll, wenn in Kanal C sein Widerpart vorhanden ist, umgekehrt muß Kanal C gestoppt werden, solange der zweite Ton in Kanal A noch nicht bereitsteht. Die Rendezvous-Bedingung für den Kanal C hat den Wert 32 (Bit 5 gesetzt). Daher ergibt sich für den

Kanalstatuswert der Töne, die mit der Begleitung verknüpft werden sollen, der neue Wert von 32+1=33. Auf der anderen Seite erhalten die Begleittöne alle den Kanalstatuswert 12. Dieser setzt sich aus der 8 für die Rendezvousbedingung und der 4 für den angesprochenen Kanal zusammen.

Bei der Rendezvous-Technik sind im wesentlichen zwei Punkte bemerkenswert. Zum ersten gilt, daß die Rendezvous-Bedingung immer in beiden Kanälen gleichzeitig, natürlich mit unterschiedlichen Kanalangaben, ausgeführt werden muß. Erst wenn beide Sound-Kommandos definiert worden sind, kommt es zur Parallelausgabe der beiden Töne. Wird dagegen nur ein Kanal mit der Rendezvous-Klausel versehen, so bleibt die Ausgabe auf diesem Kanal gegebenenfalls ewig in Wartestellung. Sie kann allerdings durch eine Flush-Ausgabe oder ein »PRINT CHR$(7)« — dieser Code setzt alle Warteschlangen zurück — jederzeit reaktiviert werden.

Man kann übrigens nicht nur zwei Kanäle, sondern auch alle drei durch Rendezvous-Bedingungen verknüpfen. Hat man es mit mehrfachen Rendezvous-Bedingungen zu tun, so gilt die »UND«-Verknüpfung. Es müssen also sämtliche Bedingungen erfüllt sein, bevor die drei Klänge zusammen ausgegeben werden. Auch sind hier bei jedem Sound-Befehl alle beide Referenz-Kanäle anzugeben.

Hierbei kann man übrigens noch einen kleinen Trick benutzen. Da sich ein Kanal immer im Rendezvous mit sich selbst befindet, können Sie problemlos das Bit für die Rendezvous-Bedingung mit dem eigenen Kanal setzen. Dies hat den Vorteil, daß Sie aus den neun möglichen Kombinationen bei dreistimmigen Stücken nicht immer die zwei passenden herauszusuchen brauchen, um diese dann zu addieren. Durch die Zusammenfassung ergibt sich nämlich immer die dreifache Rendezvous-Bedingung (Rendezvous mit A, Rendezvous mit B und Rendezvous mit C) mit dem Wert 56. Sie erhalten dann als neue Kanalstatusziffern also 57, 58 und 60 für die Ausgabe auf den Kanälen A, B und C. Ein a-Moll-Dreiklang sähe nach diesem Verfahren wie folgt aus: »SOUND 57,284,500:SOUND 58,239,500:SOUND 60,190,500«

Mit dieser etwas komplexeren Form können wir jetzt unsere Begleitung noch um jeweils einen Ton erweitern und damit normale Dreiklänge ausgeben. Sie sollten diese Erweiterung einmal selbst programmieren. Dazu müssen Sie die Kanalstatuswerte, die bis jetzt mit 33 und 12 definiert wurden, auf die neuen Angaben 57 und 60 ändern. Der dritte Ton im Kanal B enthält dann jeweils den Kanalstatus 58. Die benötigten Angaben für die Tonhöhe finden Sie wieder im Anhang VII des Handbuchs. Als Tonlänge sollten Sie jeweils 75 eintragen. Bei der Auswahl der einzusetzenden Töne sind Sie weitgehend ungebunden. Alles was schön klingt, ist erlaubt. Daß dabei auch relativ komplizierte Melodien mit etwas Fantasie produzierbar sind, demonstriert das Listing »Jailhouse Rock« (Listing 3).

```
10 ' ********************
20 ' ** Jailhouse Rock **
30 ' **    adapted by    **
40 ' **      CPC 464     **
50 ' **        and       **
60 ' **        C.S       **
70 ' ********************
80 DATA 1,301,12
90 DATA 1,0,1
100 DATA 57,301,12,58,478,12,60,568,12
110 DATA 1,284,12.5
120 DATA 1,358,12.5
130 DATA 1,426,12.5
140 DATA 1,358,12.5
150 DATA 1,319,12.5
160 DATA 1,358,12.5
170 DATA 1,426,12.5
180 DATA 1,358,23
190 DATA 1,0,2
200 DATA 1,358,12.5
210 DATA 1,301,25
220 '2.wdh
230 DATA 57,0.25,58,506,25,60,379,25
240 DATA 1,301,12
250 DATA 1,0,1
260 DATA 57,301,12,58,478,12,60,568,12
270 DATA 1,284,12.5
280 DATA 1,358,12.5
290 DATA 1,426,12.5
300 DATA 1,358,12.5
310 DATA 1,319,12.5
320 DATA 1,358,12.5
330 DATA 1,426,12.5
340 DATA 1,358,12
350 DATA 1,0,1
360 DATA 1,358,12
370 DATA 1,319,12.5
380 DATA 1,358,12
390 DATA 1,0,1
400 DATA 1,358,12
410 'dritte wdh
420 DATA 57,0,25,58,506,25,60,379,25
430 DATA 1,301,12
440 DATA 1,0,1
450 DATA 57,301,12, 58,478,12, 60,568,12

460 DATA 1,358,12.5
470 DATA 1,426,12.5
480 DATA 1,358,12.5
490 DATA 1,319,12.5
500 DATA 1,358,12.5
510 DATA 1,426,12.5
520 DATA 1,358,24
530 DATA 1,0,1
540 DATA 1,358,12.5
550 DATA 1,301,12
560 DATA 1,0,1
570 DATA 1,301,12
580 ' 4.teil
590 DATA 57,0,25,58,506,25,60,379,25
600 DATA 1,301,12
610 DATA 1,0,1
620 DATA 57,301,12, 58,478,12, 60,568,12
630 DATA 1,284,12.5
640 DATA 1,358,12.5
650 DATA 1,426,12.5
660 DATA 1,358,12
670 DATA 1,0,1
680 DATA 1,358,12
690 DATA 1,0,1
700 DATA 1,358,12
710 DATA 1,0,1
720 DATA 1,358,12
730 DATA 1,0,1
740 DATA 1,301,24
750 DATA 1,0,1
760 DATA 1,301,12.5
770 DATA 1,358,25
```


Bei der Begleitung sind wir jedoch nicht auf Melodie und zwei Begleittöne, beziehungsweise einen Begleitakkord, beschränkt. Eine weitere interessante Möglichkeit besteht darin, unserer Musik noch eine Schlagzeugbegleitung mit auf den Weg zu geben. Eine Vorbemerkung, speziell für empfindliche Ohren, ist hier jedoch gleich anzubringen. Die Benutzung des Rauschgenerators, den wir für die Schlagzeugimitation brauchen, benötigt immer einen vernünftigen Lautsprecher, das heißt die Ausgabe muß über die Stereoanlage mit dem Soundkabel erfolgen. Will man hier mit dem eingebauten Lautsprecher arbeiten, so wird man schon nach den ersten Programmierversuchen enttäuscht das Handtuch werfen. Der ausgegebene Klang ist nämlich wirklich kläglich. Spielt man das Ganze dagegen über die Stereoanlage ab, so kann man einen sehr realitätsnahen Sound erzielen.

Schlagzeugimitation

Zunächst ein paar Grundbemerkungen zu Schlaginstrumenten. Ein Schlagzeugklang läßt sich in zwei Teile zerlegen: Einen Grundton und ein überlagertes Rauschen in einem bestimmten Frequenzbereich. Beide Größen können sich dabei im Zeitablauf sowohl hinsichtlich der Lautstärke als auch der Tonhöhe ändern. Bei einigen Schlaginstrumenten fehlt der Grundklang völlig. Hier wird nur Rauschen in einem bestimmten Frequenzband erzeugt.

Wenn wir nun ein Schlaginstrument auf dem CPC imitieren wollen, so müssen wir umgekehrt vorgehen. Wir benötigen einen Rauschklang, zu dem wir dann gegebenenfalls noch einen Grundton addieren müssen. Das Rauschen erhalten wir, indem wir im entsprechenden Sound-Kommando den Einsatz des Rauschgenerators zulassen. Dazu ist der 7. Parameter im Sound-Befehl bestimmt. Dieser definiert in groben Zügen die Rauschfrequenz, die benutzt werden soll. Wird er nicht angegeben, oder auf Null gesetzt, was dasselbe ist, so wird kein Rauschen ausgegeben. Ist er eins, so erhalten wir eine relativ hohe Rauschfrequenz. Eine Erhöhung des Rauschgeneratorwertes führt zu tieferen Durchschnittsfrequenzen. Hierbei sind zwei Punkte zu beachten: Zum einen verfügt der Soundchip nur über einen einzigen Rauschgenerator; es ist daher nicht möglich auf verschiedenen Kanälen verschiedene Rauschfrequenzen auszugeben. Es wird immer auf dasselbe Rauschgeneratorsignal zurückgegriffen. Maßgeblich ist dabei die letzte Eingabe. Wenn Sie also zum Beispiel auf Kanal A eine Rauschperiode von fünf definieren und auf Kanal C danach eine Periode von zehn, so wird am Schluß die tiefere Rauschfrequenz (der Periode zehn entsprechend) ausgegeben. Punkt zwei betrifft das Lautstärkeverhältnis von Grundton zu Rauschfrequenz. Für beide Größen gilt dabei derselbe Wert, das heißt die mittlere Lautstärke der Rauschfrequenz und die Lautstärke des Grundtons entsprechen sich. Dies wirft für uns einige Probleme auf. Der charakte-

```
780 DATA 1,0,12.5
790 DATA 1,358,25
800 DATA 1,301,25
810 DATA 57,0,25,58,358,25,60,1073,25
820 DATA 57,0,12,58,358,12,60,716,12
830 DATA 57,0,1,58,358,1,60,0,1
840 DATA 57,0,12,58,358,12,60,716,12
850 DATA 57,0,25,58,358,25,60,638,25
860 DATA 57,0,12,58,179,12,60,536,12
870 DATA 57,0,1,58,179,1,60,536,1
880 DATA 57,0,11,58,179,11,60,536,11
890 DATA 57,0,12,58,179,12,60,716,12
900 DATA 57,0,1,58,179,1,60,716,1
910 DATA 57,0,12,58,179,12,60,716,12
920 DATA 1,0,25
930 DATA 1,301,25
940 DATA 1,358,25
950 DATA 57,0,25,58,478,25,60,568,25
960 DATA 57,0,12,58,568,12,60,478,12
970 DATA 57,0,1,58,568,1,60,0,1
980 DATA 57,0,12,58,568,12,60,478,12
990 DATA 57,0,25,58,568,25,60,426,25
1000 DATA 57,0,12,58,358,12,60,402,12
1010 DATA 57,0,1,58,358,1,60,402,1
1020 DATA 57,0,11,58,358,11,60,402,11
1030 DATA 57,0,12,58,358,12,60,426,12
1040 DATA 57,0,1,58,358,1,60,426,1
1050 DATA 57,0,12,58,358,12,60,426,12
1060 DATA 57,0,25,58,358,25,60,568,25
1070 DATA 57,301,12.5,58,758,12.5,60,716
,12.5
1080 DATA 57,358,12.5,58,956,12.5,60,716
,12.5
1090 DATA 57,301,12.5,58,506,12.5,60,956
,12.5
1100 DATA 57,284,12.5,58,478,12.5,60,956
,12.5
1110 DATA 57,358,12.5,58,536,12.5,60,956
,12.5
1120 DATA 57,426,12.5,58,536,12.5,60,956
,12.5
1130 DATA 57,358,24,58,536,24,60,956,24

1140 DATA 57,0,1,58,0,1,60,0,1
1150 DATA 57,358,12,58,536,12,60,478,12
1160 DATA 57,301,12.5,58,426,12.5,60,478
,12.5
1170 DATA 57,301,25,58,426,25,60,536,25
1180 DATA 57,0,25,58,426,25,60,716,25
1190 DATA 57,0,25,58,426,25,60,851,25
1200 DATA 57,358,12,58,426,12,60,1073,12

1210 DATA 57,0,1,58,0,1,60,0,1
1220 DATA 57,358,12,58,426,12,60,1073,12
1230 DATA 57,301,12.5,58,506,12.5,60,716
,12.5
1240 DATA 57,284,12.5,58,478,12.5,60,716
,12.5
1250 DATA 57,358,12.5,58,568,12.5,60,716
,12.5
1260 DATA 57,426,12.5,58,568,12.5,60,716
,12.5
1270 DATA 57,358,12,58,602,12,60,536,12
1280 DATA 57,0,1,58,0,1,60,0,1
1290 DATA 57,358,24,58,602,24,60,536,24
1300 DATA 57,0,1,58,0,1,60,536,1
1310 DATA 57,358,37.5,58,568,37.5,60,716
,37.5
1320 DATA 57,0,12.5,58,0,12.5,60,716,12.
5
1330 'DATA 57,0,37.5,58,379,37.5,60,758,
37.5
1340 DATA 0,0,0
1350 READ s,t,l:IF s=0 THEN FOR i=1 TO 2
000:NEXT:RUN ELSE SOUND s,t,l*1.3:GOTO 1
350
```

Listing 3. »Jailhouse Rock«

ristische Klang eines Schlaginstruments ergibt sich nämlich aus einem bestimmten Lautstärkeverhältnis von Grundton und Rauschfrequenz. Dieser Effekt ist zwar zu erzielen, der Preis für den Zugewinn an Realität ist allerdings erheblich. Wir müssen nämlich einen weiteren Kanal für die Ausgabe des Schlagzeugklanges opfern. Auf einem Kanal geben wir dann nur das Rauschen mit vorgewählter Lautstärke, jedoch ohne Grundfrequenz (Parameter zwei = 0) aus. Mit Hilfe einer anderen Warteschlange mischen wir dann (durch Rendezvous-Bedingungen verknüpft) den zugehörigen Grundton mit einer neuen Lautstärke zu. Dieses Vorgehen ist insbesondere dann notwendig, wenn wir für das Rauschen und den Grundton verschiedene Lautstärkeverläufe realisieren wollen.

Zunächst einmal wollen wir uns jedoch durch einige Experimente mit dem Klang von Rauschen mit oder ohne Grundton vertraut machen. Geben Sie dazu einmal ein: »SOUND 1,0,5,5,0,0,1«. Das Ergebnis ist ein hohes Rauschen ohne Grundton. Wenn Sie nun mit dem letzten Parameter spielen, können Sie die Rauschfrequenz erniedrigen. Zusätzlich zu den ersten drei Angaben, die uns ja schon geläufig sind, haben wir jetzt auch noch den Wert für die Grundlautstärke (4. Parameter) angegeben. Normalerweise, das heißt wenn dieser Parameter nicht spezifiziert wurde, ist hier der Wert 4 eingestellt. Um die Ausgabe etwas lauter zu machen, wurde hier 5 angegeben. Wir wollen nun einmal einen Grundton zu unserem Rauschen addieren. Das neue Kommando lautet dann zum Beispiel »SOUND 1,380,5,5,0,0,5«.

Hier wird nun mit einer etwas tieferen Rauschfrequenz gearbeitet. Die Tonlänge ist mit acht auch etwas länger und zusätzlich haben wir einen Ton von ungefähr 330 Hertz addiert. Sie sollten nun unbedingt einmal die Tonlänge variieren. Bei Werten von über zehn wird Ihnen die oben angesprochene Problematik des Lautstärkeverhältnisses sicherlich klar werden. Der Grundton »scheint zu stark durch«, was ein etwas seltsames Gesamtklangbild ergibt. Bei kurzen Tonlängen, wie sie zum Beispiel für die Imitation eines Beckens benutzt werden, reicht dieses einfache Vorgehen jedoch völlig aus.

Wenn Sie nun den Schlagzeug-Sound in ein Musikstück integrieren wollen, so gibt es zwei grundsätzliche Möglichkeiten. Zum einen können Sie den Schlagzeugklang wie eine normale Note in das Musikstück einbinden. Dazu gehen Sie wie beim zwei-, beziehungsweise dreistimmigen Spiel vor. Das Schlagzeug muß dann an der Stelle in der DATA-Zeile stehen, wo es eingeführt werden soll und daneben müssen natürlich auch noch die entsprechenden Rendezvous-Bedingungen erfüllt sein. Hierbei ist dann allerdings zu beachten, daß Sie für ein Rauschen alle sieben Parameter einlesen müssen. Auch die Definitionen für die Töne in den anderen Schlangen müssen dann mit sieben Parametern erfolgen, was Eingabezeit und Speicherplatz kostet. Daneben ist diese Art der Definition natürlich auch wenig komfortabel.

```
10 ' ************************
20 ' ** hrs v2+schlagzeug **
30 ' ************************
40 DATA 1,1136,23
50 DATA 33,1136,48,12,758,75
60 DATA 1,1012,25
70 DATA 33,956,50,12,638,75
80 DATA 1,758,25
90 DATA 33,851,50,12,676,75
100 DATA 1,1136,25
110 DATA 33,956,50,12,716,75
120 DATA 1,568,23
130 DATA 1,0,4
140 DATA 33,568,46,12,758,75
150 DATA 1,0,4
160 DATA 1,568,23
170 DATA 33,638,37,12,956,75
180 DATA 1,758,13
190 DATA 1,851,25
200 DATA 33,758,73,12,1012,75
210 DATA 1,0,2
220 DATA 33,758,50,12,1012,50
230 DATA 1,568,23
240 DATA 1,0,4
250 DATA 33,568,46,12,758,75
260 DATA 1,0,4
270 DATA 1,568,23
280 DATA 33,638,23,12,956,75
290 DATA 1,0,4
300 DATA 1,638,23
310 DATA 1,758,25
320 DATA 33,851,24,12,676,75
330 DATA 1,0,2
340 DATA 1,851,12
350 DATA 1,0,2
360 DATA 1,851,24
370 DATA 1,1136,12
380 DATA 33,956,50,12,716,75
390 DATA 1,1136,23
400 DATA 1,0,3
410 DATA 33,1136,50,12,758,75
420 DATA 1,956,12
430 DATA 1,851,13
440 DATA 33,758,48,12,1012,75
450 DATA 1,0,4
460 DATA 1,851,13
470 DATA 1,956,12
480 DATA 33,1136,100,12,758,100
490 DATA 0,0,0
500 p=1.4:SOUND 2,0,25*p
510 ON SQ(1) GOSUB 530
520 GOTO 510
530 READ k,t,l:IF 1=0 THEN GOSUB 550:FOR
 j=1 TO 1650:NEXT:RUN ELSE SOUND k,t/4,1
*p
540 IF (SQ(2) AND 7)<>4 THEN RETURN
550 SOUND 2,384,5*p,6,0,0,7:SOUND 2,0,20
*p:SOUND 2,0,5*p,5,0,0,3:SOUND 2,0,20*p:
SOUND 2,0,5*p,5,0,0,3:SOUND 2,0,20*p
560 RETURN
```

Listing 4.
Auch Schlagzeug kann zu den Songs gemischt werden

```
10 '
20 ' Sound-Register
30 '
40 MEMORY 41999
50 DATA 3e,00,0e,00,cd,34,bd,c9
60 FOR i=0 TO 7:READ a$:POKE 42000+i,VAL
("&"+a$):NEXT
70 INPUT"Register";reg
80 INPUT "Wert";wert
90 POKE 42001,reg:POKE 42003,wert
100 CALL 42000
110 GOTO 70
```

Listing 5.
Mit dieser Routine beeinflussen Sie die Register des Soundchips

Ein einfacher ¾-Takt

Bei Schlaginstrumenten bietet sich jedoch ein anderes Vorgehen an. Logischerweise ändert sich der Rhythmus eines Stücks nicht in jedem Takt. Der Rhythmus des zweiten Takts ist im Normalfall eine Wiederholung der Schlagzeugklänge des ersten Taktes. Wenn man sich mit einer solch einfachen Rhythmuskonstruktion zufrieden gibt, so kann man das komplette Schlagzeug als Schleife programmieren. Wir wollen das einmal anhand von »House of the Rising Sun« (Listing 4) demonstrieren. Die einzelnen Töne werden mit der READ-Bedingung in Zeile 520 in die Warteschlange gegeben. Aufgerufen wird dieses Unterprogramm von Zeile 510 beziehungsweise 520. Dort wird geprüft, ob in der Warteschlange für den Kanal A bereits wieder Platz ist. Falls diese Bedingung erfüllt ist, wird das Unterprogramm in Zeile 530 aufgerufen. Auf die Verwendung der Funktion SQ (Kanal-Nummer) kommen wir später noch zurück. Hier soll nur die Einbindung des Schlagzeugrhythmus in dem Unterprogramm beschrieben werden. Dies geht relativ einfach. Während in der vorherigen Version des Stücks nach der Ausgabe des Soundbefehls direkt wieder zurückgesprungen wurde, wird nun erst in Zeile 540 abgeprüft, ob die Warteschlange für Kanal B vielleicht leer ist. Ist dies der Fall, so wird mittels Zeile 550 der neue Schlagzeugsound ausgegeben, ansonsten direkt zurückgesprungen.

Der Schlagzeugrhythmus als solcher ist recht einfach aufgebaut (solider, einfacher ¾-Takt). Am Anfang steht ein tiefer Sound mit beigemischtem Grundton. Es folgt eine Pause, die dann zu zwei weiteren, durch Pausen getrennten, höheren Rauschsignalen überleitet. Mit wenigen Eingaben hat man so die »perfekte« Begleitung für den Dreivierteltakt zusammen.

Beim Abspielen nach dieser Methode ist allerdings noch eines zu beachten. Wir brauchen, um einen ganzen Takt mit Sound-Kommandos abspielen zu können, sechs nacheinanderstehende Befehle (vergleiche Zeile 550). Da in einer Warteschlange nur maximal vier Befehle Platz haben, hält das Basic-Programm nach der vierten Sound-Anweisung an und wartet bis die ersten beiden Schlagzeug-Klänge abgearbeitet sind, bevor es dann, nachdem die Befehle 5 und 6 eingeladen wurden, zurückkehrt. Dies bedeutet, daß die Abarbeitung der Zeile 550 in Basic mehr als eine viertel Sekunde dauert, die der Computer im wesentlichen mit Warten verbringt. Aufgrund der Warteschlangenkonzeption hat dies normalerweise keine Folgen, wenn man davon ausgeht, daß die anderen Warteschlangen beim Einsprung in das Programm vollständig geladen sind. Diese haben dann Töne für eine halbe bis dreiviertel Sekunde vorrätig, so daß der normale Tonablauf nicht durch diese Wartezeit beeinträchtigt wird. Problematisch wird es jedoch, wenn man im Melodiekanal relativ schnelle Noten bei dieser Technik ausgeben will, zum Beispiel Sechzehntelnoten. Noch unangenehmer wird es, wenn dabei auch einige identische Noten nacheinander gespielt werden müssen.

Im ersten Fall müßten während der Wartezeit vier Sechzehntelnoten abgearbeitet werden. Selbst wenn die Schlange des Melodienkanals bis auf den letzten Sound vollgestopft ist, dürfte es aufgrund der Geschwindigkeit des Basic-Interpreters zu einer kleinen Pause im Melodiekanal kommen. Gänzlich unlösbar wird dieses Problem natürlich, wenn vier gleiche Sechzehntelnoten nacheinander gespielt werden sollen.

Grundsätzlich verbindet der CPC nämlich gleiche Tonwerte. Geben wir also ein »SOUND 1,284,25: SOUND 1,284,25«, so werden nicht zwei Viertelnoten gespielt, sondern die beiden Tonlängen werden zu einer halben Note zusammengezogen. Bei gleichen Noten ist es daher notwendig, zwischen den Noten Ton- oder Lautstärke kurz für mindestens ein bis zwei Hundertstelsekunden auf Null zu setzen und die Tonausgaben der beiden Noten längenmäßig entsprechend zu verkürzen. Wir haben dieses Konzept bei der Abspeicherung des Stücks in DATA-Zeilen schon des öfteren benutzt. Diese Vorgehensweise würde aber nun bedeuten, daß für die Ausgabe unserer vier Sechzehntelnoten insgesamt auch Sound-Befehle nacheinander in die Warteschlange eingereiht werden müßten. Mit dem gerade beschriebenen Zeitablauf ist dies natürlich nicht mehr möglich. Es würde also eine unangenehme Pause in der Melodie entstehen und, was noch viel schlimmer ist, Rhythmus und Melodie wären danach nicht mehr synchronisiert. Die Melodie würde dem Takt hinterherlaufen.

Die einzige Abhilfe — wenn wir davon absehen, die Schlagzeugklänge nacheinander wie Töne abzuspeichern — besteht darin, auch Kanal B Stückchen für Stückchen zu laden und so eine Überfütterung zu vermeiden. Dazu kann zum Beispiel beim ersten Ansprung der Routine die erste Hälfte des Schlagzeugklangs ausgegeben werden. Gleichzeitig setzt man ein Flag (beispielsweise die Basic-Variable flag/) auf 1. Baut man nun eine geeignete IF-Abfrage ein (IF flag/=1), dann wird beim zweiten Anlauf der zweite Teil des Klangs ausgegeben. Danach wäre dann flag/ wieder zurückzusetzen. Als nächstes folgt dann wieder der erste Teil.

Der CPC als Schlagzeugcomputer

Natürlich funktioniert die gerade beschriebene Art der Schlagzeugdefinition auch für andere Takte und Rhythmen. Wer sich zu Hause kein eigenes Schlagzeug hinstellen kann, aber trotzdem nicht auf rhythmische Begleitung verzichten will, kann den Schneider sogar als Rhythmuscomputer einsetzen. Wenn Sie nämlich zum Beispiel

```
545 INPUT "Geschwindigkeit";P
555 GOTO 550
```

eingeben und dieses geänderte Programm nun mit RUN 545 starten, so wird der CPC zum Rhythmuscomputer. Dieser Dreizeiler (Zeile 545 bis 555) demonstriert natürlich nur das Prinzip. Wenn Sie nun einige weitere Rhythmen in mehreren Soundkommandos abgelegt haben, diese wie in Zeile 550 schreiben und das Ganze mit einem vernünftigen Menü verbinden, so haben Sie einen Schlagzeugcomputer.

Wir können den Rauschgenerator jedoch nicht nur zur Erzeugung von Schlagzeugimitationen benutzen. Das gesamte Gebiet der Soundeffekte bietet ein breites Anwendungsfeld auch speziell für Rauschklänge. Sie reicht von Meeresbrandung bis zum Dröhnen oder Rauschen der Raketentriebwerke in einem Actionspiel. Auch so manch anderer Klang, bei dem man den Einsatz vom Rauschgenerator zuerst gar nicht vermutet, greift doch darauf zurück. Meist wird aber nicht mit konstanten Rauschfrequenzen gearbeitet, sondern das Rauschen wird periodisch im Zeitablauf variiert. Beim Schneider erreichten wir dies, indem wir nacheinander dasselbe Sound-Kommando jeweils mit geänderter Rauschfrequenz ausgeben. Nachfolgend ein kleines Beispiel.

```
10 FOR i=1 TO 7: SOUND 1,0,25,5,
0,0,i:NEXT i
```

Diese kleine Schleife gibt nacheinander sieben Befehle mit geänderten Rauschfrequenzen aus, beginnend bei den höchsten Frequenzen. Der Effekt ist jetzt schon recht eindrucksvoll. Wollen wir diesen Vorgang nur periodisch als Schwingung ablaufen lassen, das heißt von hohen Frequenzen zu tiefen und wieder zurück zu hohen, und so weiter, so haben wir einen Klang, der dem Brandungsrauschen relativ ähnlich ist. Dazu müssen wir die eben angegebene Zeile nur noch ein wenig erweitern. Sie lautet jetzt:

```
10 FOR i=1 TO 7: SOUND 1,0,5+2*
i,5,0,0,8-i:NEXT i: GOTO 10
```

Nun haben wir eine periodische Rauschschwingung erhalten. Töne aus seinem Schneider zu locken heißt ausprobieren und experimentieren. So können Sie zum Beispiel aus den Wellen leicht eine Dampflok machen.

```
20 FOR i=1 TO 7:SOUND 1,0,12,i,0,
0,8-i:NEXT i:FOR i=1 TO 7:SOUND
1,0,12,i,0,0,i:NEXT i:GOTO 20
```

Oder variieren Sie zum Beispiel einmal die Tonlänge und — was unseren Wellenklang noch realer macht — definieren Sie auch die Lautstärke in Abhängigkeit von der Schleifenvariablen. Als vierten Parameter im ersten Sound-Kommando können Sie zum Beispiel i einsetzen, im zweiten Kommando muß es dann 8-i heißen. In diesem Fall nimmt nicht nur die Rauschfrequenz ab, sondern gleichzeitig auch noch die Lautstärke zu. Die maximale Lautstärke wird in diesem Fall bei der tiefsten Frequenz erreicht. Vertauschen Sie nun einmal die Lautstärkeangaben im ersten gegen die des zweiten Sound-Befehls. Es entsteht wieder ein neuer Klang. Sollte Ihnen dabei die Abfolge der Wellenbewegung zu eintönig ablaufen, so können Sie auch vor dem GOTO noch eine zufallsabhängige Warteschleife einfügen, zum Beispiel »FOR t=1 TO 3000*RND (0):NEXT«. Dazu müssen Sie allerdings dann die beiden Sound-Anweisungen inklusive der dazugehörigen Schleifen miteinander vertauschen. Schon geringste Änderungen in den Kommandos können dabei zu völlig anders klingenden Tönen führen. Wenn Sie zum Beispiel statt der reinen Rauschausgabe in unseren Befehlen noch einen kleinen Grundton, zum Beispiel mit der Periode 2000, mitklingen lassen, so wird aus der Welle ein Propellerflugzeug. Ersetzen Sie die 2000 durch eine 1000, so entsteht ein Sound, den man aus Actionfilmen nur zu gut gewohnt ist. Das ungesunde Brummen eines Elektrizitätswerkes kurz bevor dieses in die Luft gejagt wird. Obschon die zunehmende, beziehungsweise abnehmende Lautstärke vom menschlichen Ohr bereits relativ schnell als der Klang eines sich bewegenden Gegenstandes interpretiert wird, können wir diesen Effekt noch verstärken, wenn wir zusätzlich auch noch mit den Stereokanälen arbeiten. Geben Sie dabei auf den Kanälen A und C zweimal denselben Sound-Befehl aus, nur daß in einem Fall die Schleife von i=1 bis 7 läuft, während sie auf dem anderen Kanal von 7 bis 1 fällt. Da wir in unserem letzten Beispiel die Lautstärke bereits von dem Parameter i abhängig gemacht hatten, ändert sich die Lautstärke gleich mit. Unser Flugzeug wandert nun auch akustisch von links nach rechts und wieder zurück. Welche Effekte sich hieraus noch für Actionspiele gewinnen lassen, braucht wohl nicht näher ausgeführt zu werden. Lassen Sie Ihrer Fantasie und Ihrem Spieltrieb einfach freien Lauf und probieren Sie.

Wir haben bis jetzt hauptsächlich das Rauschen mit Hilfe von Zählschleifen geändert. Dasselbe Verfahren können Sie natürlich auch zur Imitation einer Sirene anwenden. Nur sollten Sie in diesem Fall i über ein paar mehr Stufen laufen lassen. Zum Beispiel:

```
10 FOR i=1000 TO 1 STEP —1:
SOUND 1,i+100,3:NEXT i:FOR i=1
TO 1000: SOUND 1,100+i,3 NEXT
i:GOTO 10.
```

Neben Schleifen gibt es allerdings auch noch eine andere Möglichkeit, beim CPC im Sound-Befehl Lautstärken, beziehungsweise Tonhöhen-Änderungen gleich mit zu programmieren. Dafür sind die Parameter 5 und 6 in der Sound-Anweisung zuständig. Teilweise mußten wir sie bereits bei der Definition des zugeschalteten Rauschens angeben, wobei wir sie aber immer auf Null gesetzt hatten.

Veränderbare Töne durch Hüllkurven

Mit diesen beiden Parametern ist es möglich, komplexe Lautstärke- und Tonverläufe zu programmieren. Die Definition erfolgt dabei mittels sogenannter Ton- beziehungsweise Lautstärke-Veränderungsfolgen.

Das Prinzip ist dabei sehr ähnlich. Im Sound-Kommando muß an der entsprechenden Stelle (Parameter 5 oder Parameter 6) eine Zahl zwischen 1 und 15 stehen. Diese steuert die Lautstärke beziehungsweise die Veränderung der Töne. Die Veränderungen selbst werden dabei mit den Befehlen ENT und ENV definiert.

ENT legt einen komplexen Tonverlauf fest, bei ENV ändert sich die Lautstärke. Komplex bedeutet hierbei, daß sehr verschiedenartige Änderungen, die nacheinander ausgeführt werden sollen, in einer Änderungsfolge definiert werden können. Ein Beispiel für eine komplexe Lautstärkehüllkurve könnte ein kurzes Anschwellen der Lautstärke sein, beginnend bei 0 bis zu einem Maximalwert. Dann folgt ein langsames Abgleiten auf ein geringeres Niveau, das dann über eine gewisse Zeit erhalten bleibt, bevor die Lautstärke bis auf den Nullpegel sinkt. In diesem Beispiel wären vier verschiedene Änderungen nacheinander auszuführen. Diese Veränderungen können nun in einer Lautstärke-Veränderungsfolge definiert werden. Um Lautstärke- und Tonverläufe abzuspeichern, stehen beim Schneider 15 Lautstärke- und 15 Tonverläufe zur Verfügung. Sie tragen die Nummern 1 bis 15.

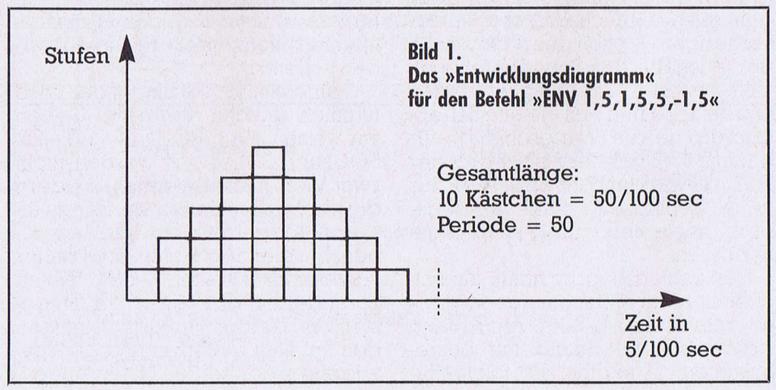


Bild 1.
Das »Entwicklungsdiagramm« für den Befehl »ENV 1,5,1,5,5,-1,5«

Programmierte Lautstärkewechsel mit ENV

Gibt man im Soundkommando statt der Null an der fünften Stelle eine Zahl zwischen 1 und 15 ein, so wird der Sound-Befehl unter der dazugehörigen (Lautstärke-)Hüllkurve ausgeführt. Die im Sound-Befehl definierte Lautstärke stellt dann nur noch die Grundlautstärke dar, die die Ausgangsbasis für die Hüllkurve bildet. Die Ausführung eines Sound-Kommandos unter einer Veränderungsfolge geschieht also in zwei Stufen. Zuerst einmal muß die gewünschte Veränderung definiert werden. Dazu dient der Befehl ENV. Danach kann die so definierte Folge in einer beliebigen Anzahl von Sound-Befehlen benutzt werden.

ENV hat als ersten Parameter die Nummer der Hüllkurve und dann maximal fünf Veränderungsfolgen, wobei jede Folge durch drei Parameter definiert ist. Die erste Angabe steht dabei für die Anzahl der auszuführenden Änderungsschritte. Angabe Nummer 2 bezieht sich auf die Schrittweite und der dritte Parameter gibt die Pausenzeit zwischen den einzelnen Schritten in Hundertstelsekunden an. Soll also die Lautstärke in einer ganz einfachen Anwendung von dem Grundniveau 4 in drei Schritten zu je zwei Lautstärkestufen steigen, wobei die Umschaltung von einer Lautstärkestufe auf die nächsthöhere nach jeweils einer Zehntel Sekunde erfolgen soll, so müssen wir wie folgt vorgehen. Wir nehmen einmal an, die Hüllkurve für die Lautstärke soll die Nummer 1 tragen. Danach wären nun die drei benötigten Änderungsparameter anzugeben: Drei Schritte mit je zwei Lautstärkestufen im Abstand von zehn Hundertstelsekunden, also »ENT 1,3,210«. Das zugehörige Sound-Kommando würde dann »SOUND 1,100,1000,4,1« lauten.

Nach demselben Prinzip sind auch schwierige Änderungen zu definieren. Die benötigten Angaben für eine zweite — und alle weiteren Änderungen — werden dabei ganz einfach durch Kommas getrennt und hinten an den Befehl ENT angehängt. Ist die erste Änderung abgeschlossen und die Gesamtlänge im Sound-Kommando noch nicht erreicht, so wird die zweite durchgeführt und so weiter. Im Zusammenhang mit der Definition der Lautstärke sind jedoch noch einige Besonderheiten und Korrekturen zum Schneider-Handbuch hervorzuheben.

Zunächst einmal zum Umfang der Lautstärke: Wenn keine Hüllkurve definiert ist, kann die Lautstärke zwischen 0 und 7 schwanken (0 bedeutet dabei, daß der Ton nicht ausgegeben wird, 7 kennzeichnet den Maximalwert). Gibt man für die Lautstärke den Parameter 8 ein, so hört man wiederum nichts. 9 entspricht der Lautstärke, die man auch bei 1 erhalten würde und so weiter. Die Lautstärke wird also bei der Abarbeitung des Sound-Kommandos Modulo 8 genommen. Die Modulo-Funktion führt eine Division eines Wertes durch einen anderen durch und der dabei verbleibende Rest wird als Ergebnis zurückgegeben. Obwohl im Handbuch nicht aufgeführt, verfügt auch der CPC über diese Funktion und zwar auch unter Basic. Das Befehlsformat ist dabei x MOD y, wobei x den Wert darstellt, auf den die Modulofunktion angewendet werden soll, und y den einzusetzenden Faktor. Wenn Sie also zum Beispiel eingeben »PRINT 9 MOD 8« so erhalten Sie als Ergebnis eine »1« (9:8=1 Rest 1). Die »1« vom Rest wird zurückgegeben. Dieses Teilen durch 8 wird aber nur bei fehlender Lautstärke-Veränderungsfolge benutzt. Normalerweise werden die Werte der Lautstärke Modulo 16 genommen. Die Schrittanzahl bei einer Lautstärkefolge kann zwischen 0 und 127 schwanken, die Schrittweite zwischen —128 und 127. Mit Hilfe von Schrittweite und -anzahl ist also problemlos ein Überschreiten von 15 möglich. Eine solche Lautstärke-Veränderungsfolge beinhaltet dann eine periodische Bewegung. Das heißt nachdem der Maximalwert überschritten wurde, beginnt der Ton wieder ganz leise. Probieren Sie den folgenden Zweizeiler aus

```
10 ENV 1,100,2,20
20 SOUND 1,100,1000,4,1.
```

Die Tonausgabe beginnt nun mit mittlerer Lautstärke, definiert durch die 4 im Sound-Befehl. Danach wird es so lange lauter (in Zweierschritten), bis die 15 überschritten wird. Jede Änderung der Lautstärke erfolgt dabei nach einer Fünftel Sekunde (20/100 Sekunden). Beim sechsten Schritt wird dann der Wert von 15 überschritten. Die daraus resultierende 16 setzt die Lautstärke auf 0 zurück. Mit Anfangswert 0 erhöht sich dann die Lautstärke wieder in Zweierschritten. Die nächste Periode hat begonnen. Es ist natürlich nicht nur möglich die Lautstärke zu erhöhen, sondern auch zu vermindern. Dazu muß der Wert mit negativem Vorzeichen eingegeben werden. Ändern Sie doch dazu einmal in Zeile 10 die 2 in —2. Sie erhalten jetzt einen linear abnehmenden Ton.

Damit wären wir auch schon bei der Anwendung verschiedener Hüllkurven. Neben dem breiten Bereich der Soundeffekte stellt nämlich insbesondere die Ausgabe eines Tones mit relativ kurzfristigen Änderungen der Lautstärke das Hauptanwendungsgebiet dar. Wenn wir einen Ton mit einem »normalen« Instrument erklingen lassen, so ist dieser nicht, wie beim CPC gewohnt, sofort mit maximaler Lautstärke vorhanden, sondern klingt erst leise an. Er bleibt normalerweise auch nicht auf dem gewählten Lautstärkepegel, ebensowenig reißt der Ton nach Ablauf der Tonlänge sofort ab, sondern wird langsam leiser.

Am einfachsten kann man sich dies klar machen, wenn man den Schwingungsvorgang eines Saiteninstrumentes betrachtet. Beim Anschlagen der Seite nimmt die Lautstärke zunächst zum Maximalwert hin relativ schnell zu. Der Ton bleibt danach für eine gewisse Zeit erhalten bevor er langsam ausschwingt. Dieses An- und Ausschwingverhalten ist eines der Hauptkennzeichen, mit Hilfe derer wir Instrumente unterscheiden können. Soll also mit Hilfe des Computers irgendein Instrument imitiert werden, so müssen wir auch die dazugehörige Lautstärkehüllkurve imitieren. Wir wollen dies einmal anhand eines Beispiels klarmachen. Dazu schauen wir uns einmal an, wie wir mit Hilfe der Lautstärkekurve den Grundton A mit Länge einer halben Note, aber unterschiedlicher Hüllkurve, mit ganz verschiedenen Klängen ausgeben können. Zunächst einmal die Standardansprache für ein tiefes A.

```
20 SOUND 1,284,50,5
```

Wir ändern nun die Lautstärkeangabe auf den Wert 0 und fügen, durch nachfolgendes Komma getrennt, eine Lautstärkehüllkurve mit Nummer 1 an. Ohne die Definition der Lautstärkehüllkurve würde die Tonausgabe nun wegen Lautstärke 0 unterdrückt werden. Definieren wir jetzt

```
10 ENV 1,5,3,1,10,0,3,6,—2,2
```

so wird der Ton wieder ausgegeben, jetzt aber unter der Bedingung der Hüllkurve. Wenn Sie die Klangwirkung dieser Ausgabe mit der vorherigen vergleichen, werden Sie den Unterschied relativ schnell feststellen. Zum Vergleich einmal die beiden Töne nacheinander.

```
10 ENV 1,5,3,1,10,0,3,6,—2,2
20 SOUND 1,284,50,0,1
30 FOR i=1 TO 1000:NEXT
40 SOUND 1,284,0,5,0
```

Die hier gewählte Hüllkurve führte in fünf Schritten zunächst zu größerer Lautstärke bis zum Maximalwert, beginnend bei 0. Danach bleibt der Ton 30 Hundertstelsekunden lang konstant bevor die Kurve dann langsam abflacht. Sie sollten nun unbedingt einmal mit den Parametern in der Lautstärkehüllkurve spielen, um sich mit den verschiedenen Wirkungen vertraut zu machen. Während Sie mit den oben definierten Parametern einen oboenähnlichen Klang erhalten, können Sie durch Variation (Änderung des Anbeziehungsweise Ausschwingverhaltens) viele andere Instrumentenklänge erhalten.

Allerdings ist zu beachten, daß Sie dieselbe Lautstärkehüllkurve natürlich nur für eine bestimmte Tonlänge verwenden können. Je nach Tonlänge und Instrument muß also mit unterschiedlichen Hüllkurven gearbeitet werden. Dies ergibt sich zum einen daraus, daß die Haltezeit auf Maximalpegel, je nach Länge des Tones, unterschiedlich lang definiert werden muß. Zum anderen ist es natürlich auch nötig, das Ausschwingverhalten um einiges auszudehnen, wenn die Erhöhung der Tonlänge nicht nur auf einem längeren Aushalten auf Restniveau (dem 4. Teil der Kurve) beruhen soll. Je mehr Arbeit Sie gerade hier investieren, desto realistischer wird der Klang. Glücklicherweise müssen die Tonhüllkurven nur einmal, zum Beispiel am Programmanfang in einem Initialisierungsteil, definiert werden. Später können sie dann bei jedem neuen Sound-Befehl ohne Neudefinition aufgerufen werden. Da wir 15 Hüllkurven zur Verfügung haben, treten normalerweise keine Probleme wegen der Anzahl verschiedener Töne auf. Etwas eng wird er allerdings, wenn man sehr kurze Töne unter einer Hüllkurve spielen will. Denn schließlich ist der zeitliche Abstand zwischen zwei Änderungen mit einer Hundertstelsekunde relativ groß, wenn eine Sechzehntelnote nur 6,25 Hundertstelsekunden dauert. Somit sind für die Hüllkurvendefinition solcher Noten maximal sechs Änderungen machbar. Danach ist die Note zu Ende und kann logischerweise nicht mehr verändert werden. Besonders bei etwas komplizierteren Lautstärkeverläufen stößt man relativ schnell an die Grenzen des Machbaren. In der Regel reichen die vorhandenen Möglichkeiten vollkommen aus.

Wie lassen sich nun Töne mit Hilfe von Tonveränderungsfolgen variieren? Das Ganze läuft als Prozeß ähnlich dem der Lautstärkedefinition ab. Mit ENT wird eine Tonhüllkurve definiert. Diese hat wiederum zuerst eine Zahl zwischen 1 und 15, die Nummer der Hüllkurve. Als nächstes folgen die für die Änderung benötigten Parameter. Hier sind wieder maximal fünf Blöcke zu je drei Parameter in einem Befehl realisierbar. Der erste der drei Parameter definiert wieder die Anzahl der Schritte (durch das Vorzeichen spezifiziert). Der mittlere bezieht sich auf die Pausen zwischen den einzelnen Änderungen. Die Definition, ob ein Ton unter einer Tonhüllkurve ausgegeben werden soll, erfolgt durch die Angabe des sechsten Parameters im Sound-Kommando. Ton- und Lautstärkehüllkurven schließen sich dabei nicht aus, das heißt Klänge können gleichzeitig sowohl unter einer Lautstärke- wie unter einer Tonhüllkurve ausgegeben werden.

Der Ton ändert sich ständig

Bei der Definition von ENT muß allerdings noch auf eine Besonderheit verwiesen werden. Während bei ENV nur positive Zahlen zwischen 1 und 15 erlaubt sind, kann man mit ENT auch eine negative Tonhüllkurve definieren. Der Unterschied zwischen positiver und negativer Folge ist aber recht einfach. In jedem Fall wird die Tonveränderungsfolge ausgeführt. Ist der Sound danach noch nicht beendet (Tonlänge größer als die Anzahl der Schritte durch die Tonveränderungsfolgen), so dauert der Restton bei positiven Hüllkurven auf dem letzten Niveau weiter an. Im anderen Fall wird die Tonveränderung so lange wiederholt, bis der Sound zu Ende ist. Zyklisch ablaufende Veränderungen werden so definiert.

Das einfachste Beispiel dafür ist eine Sirene. Wie man eine Sirene mit Hilfe von Schleifen programmiert, haben wir schon weiter vorn betrachtet. Zunächst nimmt der Ton von tiefen zu hohen Frequenzen zu,

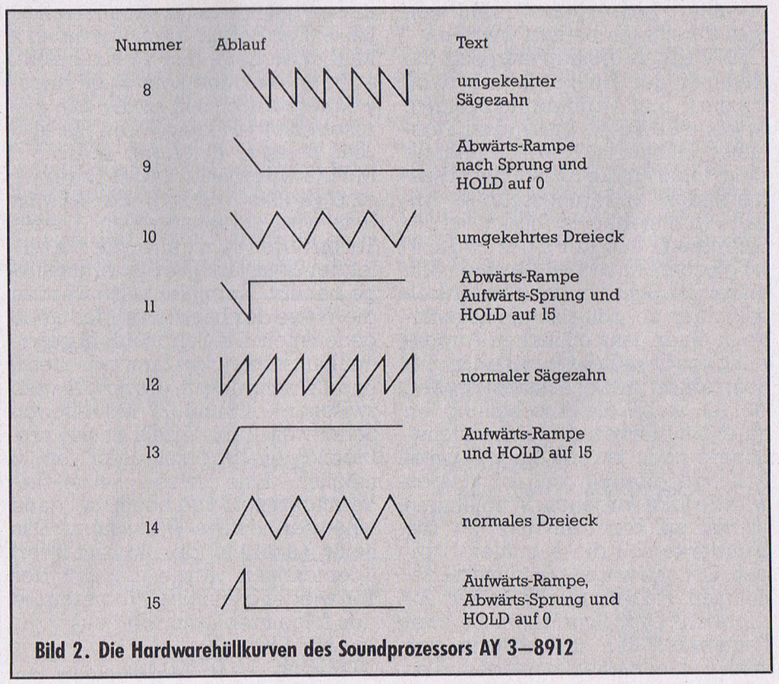


Bild 2. Die Hardwarehüllkurven des Soundprozessors AY 3—8912

bevor er dann wieder abfällt. Das Ganze wiederholt sich als zyklischer Prozeß. Mit ENT können wir dies nun ohne umständliche Schleifen leicht programmieren. Vergleichen Sie einmal die Ergebnisse mit positiven oder negativen Vorzeichen.

```
10 ENT 1,35,—10,1,35,10,1
20 SOUND 1,400,500,6,0,1
```

Jetzt läuft nur eine Schwingung ab. Ändern Sie nun die Tonhüllkurvennummer in —1, so gibt es eine richtige Sirene. Ein immer wiederkehrender Prozeß ist also schon mit zwei Veränderungsfolgen zu realisieren.

Ein so definierter Klang hört sich jedoch noch etwas ungewohnt an. Der Grund dafür ist in der Art und Weise zu suchen, in der der CPC Tonhöhen festlegt. Während nämlich unsere Sirene mit relativ gleichmäßigen Tonhöhenänderungen zum Maximum wandert, also immer dieselbe relative Änderung durchführt, ändern wir die Tonhöhe absolut. Schlagen Sie einmal in Anhang VII des Handbuchs nach und Ihnen wird das Problem sofort klar. Sie finden dort in der Oktave 0 den Wert für die Periodenänderung, der notwendig ist, um den Ton C in ein D zu verwandeln (52 oder richtiger: —52). Diese Zahl errechnet sich, indem man den Periodenwert des C von dem des D abzieht (426—478 = —52). Wenn wir also in einer Stufe den Ton vom C in ein D verändern wollen, so muß die Tonhöhenänderung —52 betragen. Wenn wir dieselbe Änderung (—52) in der Oktave 2 machen, so erhalten wir gleich die nächste Oktave. Dies führt dazu, daß wir keine lineare Erhöhung des Tones erhalten, sondern ein »Überschwappen« in den hohen Frequenzbereichen haben. Aber auch hierzu gibt es Abhilfe. Zum einen können wir natürlich ganz auf ENT verzichten und die Sirene wieder mit einer FOR...TO...-Schleife imitieren. Der andere Weg läuft über mehrere Tonveränderungsfolgen. Wir müssen im unteren Bereich mit größeren Sprüngen operieren, um dann bei höheren Frequenzen die Schrittweite kleiner zu machen. Dies sieht beispielsweise so aus:

```
10 ENT —1,50,—10,1,40,—5,1,40,5,1,50,
10,1
20 SOUND 1,1000,2000,5,0,1
```

Neben Sirenen und ähnlichen »Spielklängen« kann man die Tonvariation mit ENT auch noch für die einfache Tonausgabe benutzen. Zum Beispiel um spezielle Klangverfremdungen zu erhalten.

Zwei beliebte Sound-Effekte sind Chorus und Vibrato. Der erste entsteht dadurch, daß man zwei nebeneinanderliegende Töne sehr eng benutzt — also Klänge mit fast gleicher Periode. Beim Schneider ist dies durch zwei gleichartige Sound-Befehle möglich, die zur gleichen Zeit bearbeitet werden (zum Beispiel »SOUND 1,284,100:SOUND 4,285,100«). Auf den Kanälen A und C wird jetzt der Ton mit geringer gegenseitiger Verstimmung ausgegeben. Man erhält einen sphärenartigen Klang.

Den zweiten Effekt, das Vibrato, erzeugt man, indem ein Ton fortwährend in der Tonhöhe variiert wird. Allerdings nur in einem sehr engen Rahmen. Dies erreichen Sie zum Beispiel mit »ENT —1,1,1,5,1,—1,5: SOUND 1,284,100,5,0,1«. Das Ergebnis ist ein relativ schwaches, aber schnelles Vibrato. Sie können nun sowohl die Pausenzeit, wie auch die Anzahl der Schritte — oder die Schrittweite — im ENT-Kommando verändern. Natürlich läßt sich auch mit Hilfe von ENV ein netter und allgemein bekannter Effekt erzeugen — das sogenannte Tremolo. Beim Tremolo wird durch einen speziellen Verstärker die Lautstärke zyklisch durch eine Sinusschwingung variiert, das heißt sie nimmt zunächst zu, dann wieder ab und so weiter. Ein Lautstärke-Änderungszyklus hat dabei meist eine Länge zwischen einer Zehntel und einer ganzen Sekunde. Diesen Effekt kann man natürlich relativ einfach mit ENV nachbilden. Wir sind dabei jedoch mit einem kleinen Problem konfrontiert. Das ENV-Kommando ermöglicht uns ja nicht eine zyklische Lautstärkeänderung zu programmieren. Wir müssen daher auf eine Schleife zurückgreifen. Dazu programmieren wir den einen Zyklus mit Hilfe von ENV (zum Beispiel 10 ENV 1,5,1,5,5,—1,5).

Die Lautstärke soll also in Einerschritten fünfmal erhöht werden, wobei zwischen den einzelnen Schritten eine Pause von fünf Hundertstelsekunden eingelegt wird. Danach liegt die Lautstärke wieder auf Ausgangsniveau. Geben wir nun

```
20 SOUND 1,284,100,5,1
```

ein, so wird der Lautstärkezyklus einmal durchlaufen und dann auf dem letzten Lautstärkeniveau der Ton mit seiner Restlänge ausgegeben. Der Grund ist, daß die Tonlänge länger definiert war als die Änderungsangaben in der Hüllkurve. In der Hüllkurve werden insgesamt fünf Änderungen für die Erhöhung der Lautstärke und danach fünf Änderungen bis zum Verstummen ausgeführt. Dies geschieht jeweils mit einer Zwanzigstelsekunde Pause, womit der gesamte Zyklus eine halbe Sekunde dauert. Da unser Ton aber eine Länge von einer Sekunde hat, müssen wir im Sound-Kommando die Tonlänge auf 50 herabsetzen und mit »30 GOTO 20« diesen Befehl mehrmals nacheinander aufrufen. Damit haben wir dann den gewünschten zyklischen Prozeß. Dasselbe Verfahren können wir natürlich auch bei kürzeren Lautstärkeschwingungen benutzen. Hier muß dann wiederum die Tonlängenangabe im Sound-Kommando vermindert werden, um die notwendige Übereinstimmung zwischen ENV-Kurve und Soundlänge zu erreichen. Wenn Sie nun in Zeile 30 noch eine Zählvariable, zum Beispiel z, bei jedem Durchlauf erhöhen, können Sie auch einfach eine Gesamtlänge für den Sound festlegen. Beim Überschreiten einer bestimmten Höchstgrenze kann dann ein Sprung aus dieser Endlosschleife heraus erfolgen (beispielsweise bei z>10).

Die Programmierung von Lautstärke und Tonveränderung mittels Hüllkurven sieht zwar auf den ersten Blick etwas schwierig aus, aber wenn Sie alle Beispiele ausprobiert haben, dann besitzen Sie jetzt bereits einige Übung. Wenn Sie die Ton- und Lautstärkeänderung in einzelnen Schritten auf einem karierten Blatt auftragen, werden Sie auch keinerlei Probleme haben, die richtige Länge im Sound-Kommando zu bestimmen. Ein Beispiel für ein solches Sound-Entwicklungs-Diagramm finden Sie in Bild 1. Als Bezeichnungen auf der X-Achse wählen Sie dazu am besten die minimale Zeit für einen Veränderungsschritt. Auf der Y-Achse werden dann die Lautstärkepegel aufgetragen.

Auch im Bereich der Ton- und Lautstärkehüllkurven konnten wir natürlich nur anhand einiger Beispiele aufzeigen, was möglich ist. Wenn Sie die beiden Hüllkurvenarten miteinander kombinieren und das Ganze dann auf verschiedenen Kanälen mit unterschiedlichen Synchronisationsbedingungen oder Rauschen ausgeben, wird Ihnen erst richtig klar, welche Klänge der CPC alles machen kann.

Doch damit sind unsere Möglichkeiten im Soundbereich noch längst nicht erschöpft. Was die Klangerzeugung von Basic aus betrifft, haben wir zwar alle wesentlichen Punkte beschrieben, doch schließlich kann der CPC nicht nur Basic. Wenn wir noch mehr erreichen wollen, müssen wir uns mit der

Maschinen- oder besser gesagt Systemebene beschäftigen. Einen ersten Einstieg dazu soll uns die Beschreibung der ENV- und ENT-Kommandos liefern. Dabei findet man eine Reihe von Angaben, die scheinbar nicht so recht zusammenpassen und im übrigen von Basic aus auch nicht ausführbar sind. Betrachtet man jedoch die Bearbeitung dieser Routinen von der Maschinenseite, so wird alles sehr schnell verständlich.

Jede Hüllkurve, ENV oder ENT, benötigt genau 16 Byte. Das erste Byte repräsentiert dabei die Hüllkurvennummer. Es folgen maximal 15 belegte Bytes, wobei jeweils drei eine Veränderungsfolge aufnehmen (wie schon von Basic gewohnt, mit Schrittanzahl-Änderungsrichtung und Änderungshöhe). Als dritte Angabe steht die Pausenzeit. Das Handbuch weist nun einige auf den ersten Blick etwas seltsame Einschränkungen auf. So ist zum Beispiel für die Schrittanzahl nur ein Wert zwischen 0 und 239 erlaubt, was bei 8 Bit scheinbar keinen Sinn ergibt. Ähnlich sieht es bei dem Befehl ENV aus. Hier sind nur Schrittzahlen zwischen 0 und 127 erlaubt, obwohl auch hier scheinbar 255 Schritte möglich sind. Warum gibt es diese Einschränkungen?

Auf der Maschinenebene ist es möglich, neben der relativen Angabe einer Änderung einen absolut definierten Endwert anzugeben. Auf diese Art und Weise legt der CPC bei einer Tonveränderungsfolge, die vor dem Ende des Sounds beendet sein muß, fest, daß der zuletzt definierte Ton für den Rest des Sounds erhalten bleiben soll.

Bei ENV ist zusätzlich noch etwas anderes möglich: Der Direktzugriff auf die Hardwarehüllkurven des Soundprozessors. Um diese zu benutzen, müssen wir jedoch zunächst einmal den beim CPC verwendeten Soundchip und dessen interne Struktur etwas näher betrachten.

Der Schneider benutzt den Soundchip AY-8912 von General Instruments. Dieser IC besitzt drei getrennt programmierbare Ton- und einen Rauschgenerator. Dieser kann wahlweise einen oder mehreren Registern zugeschaltet werden. Dabei sind die Frequenzen des Ton- und Rauschgenerators über die Software programmierbar. Die Lautstärke des gemischten Signals, das heißt des Tonsignals mit Rauschen, kann ebenfalls softwaremäßig eingestellt werden. Neben diesen Möglichkeiten besitzt der Soundchip noch eine veränderbare Hardware-Hüllkurve. Bei Verwendung dieser Hüllkurve wird die Änderung der Lautstärke im Zeitablauf durch den Soundchip selbst gemacht, während bei ENT und ENV jede Veränderung durch die Software gesteuert wird. Hier verändert also das Betriebssystem fixe Werte und schafft so die neuen Klänge.

Zur Definition der gewünschten Eigenschaften verfügt der — auch als PSG (programmierbare Sound-Generator) bezeichnete — Chip über 16 Register, von denen wir 15 (mit den Nummern 0 bis 14) benutzen dürfen.

Die Register 0 bis 5 bestimmen die Periodendauer und damit die Frequenz des Tonsignals. Diese Register sind als 12-Bit-Wert gespeichert. Dabei enthalten die Register 0, 2 und 4 die niederwertigsten 8 Bit für die Generatoren A, B und C. Die unteren vier Bit der Register 1, 3 und 5 stellen die Frequenz grob ein. Je kleiner der 12-Bit-Wert eines Registers wird, desto höher ist der Ton. Um einen Ton über einen Kanal ausgeben zu können, muß jedoch das entsprechende Bit im Freischaltregister (Register 7) gelöscht sein.

Register 6 gibt die mittlere Frequenz des Rauschgenerators an. Sie ist in den unteren 5 Bit gespeichert. Die oberen drei werden nicht benutzt. Ob das Rauschen zugeschaltet wird, oder nicht, wird wiederum von Register 7 gesteuert.

Das Freischaltregister (7) sagt also aus, ob das Signal des Ton- oder Rauschgenerators auf die Ausgänge weitergegeben werden soll. Daneben ist es noch für die Richtung eines zusätzlich integrierten Ports zuständig. Die notwendigen Daten sind dabei in den verschiedenen Bits abgelegt. Die Bedeutung:
Bit 0: Freischaltung Kanal A
(0 = freigeschaltet, 1 = gesperrt)
Bit 1: Freischaltung Kanal B
(0 = freigeschaltet, 1 = gesperrt)
Bit 2: Freischaltung Kanal C
(0 = freigeschaltet, 1 = gesperrt)
Bit 3: Rauschaddition Kanal A
(0 = zugeschaltet, 1 = abgeschaltet)
Bit 4: Rauschaddition Kanal B
(0 = zugeschaltet, 1 = abgeschaltet)
Bit 5: Rauschaddition Kanal C
(0 = zugeschaltet, 1 = abgeschaltet)
Bit 6: Port-Modus
(0 = Eingabe, 1 = Ausgabe)
Bit 7: nicht verwendet (Ausgabe-Modus des nicht herausgeführten zweiten Ports)

Die Register 8 bis 10 bestimmen die Lautstärke der verschiedenen Kanäle. Dabei werden die Bits 0 bis 3 für die Definition der Lautstärke benutzt, so daß Werte von 0 bis 15 möglich sind. Das fünfte Bit hat dabei eine Sonderfunktion. Ist es gesetzt, so kann nicht auf die unteren 4 Bit zugegriffen werden, sondern der Lautstärkeverlauf wird durch die Hardwarehüllkurve bestimmt (Register 11 bis 13). Register 8 legt die Lautstärke für den Kanal A fest, 9 für B und 10 für C.

Wenn Sie diese Registerstruktur im Kopf behalten, verstehen Sie natürlich auch die Einschränkungen bei ENT und ENV. Um die 12-Bit-Tonregister setzen zu können, benötigen wir zwei Bytes, wobei das eine voll benutzt wird, von dem anderen jedoch nur die unteren vier Bits. Beim CPC hat man die oberen Bits für die Entscheidung benutzt, ob die eingebaute Hardwarehüllkurve benutzt wird oder nicht. Falls alle vier Bit gesetzt sind, so wird der Rest des Bytes und das nachfolgende Byte als Definition eines Tons verstanden, der mit der Hardwarehüllkurve angegeben wird. Ähnlich sieht das Ganze bei ENV aus. Ist in der Anzahl der Schritte das höchste Bit gesetzt, so wird nun nicht mehr auf die programmierte Softwarehüllkurve zurückgegriffen, sondern mit den Registern 11 bis 13 eine Hardwarehüllkurve von definierter Länge angesprochen.

Die Funktion der Hardwarehüllkurve entspricht im Prinzip der Definition von Lautstärkeveränderungen mit ENV. Mit den Registern 11 und 12 (Lo-Hi abgespeichert), wird die Periodendauer, das heißt die Länge der Kurve, festgelegt. Register 13, genauer gesagt die Bits 0 bis 3 von Register 13, bestimmen die Kurvenform des Hüllkurvengenerators. Dabei stehen eine ganze Reihe von verschiedenen Abläufen zur Verfügung. Es sind hier auch relativ schwierige, und was besonders wichtig ist, periodische Abläufe vorgesehen. Da die verbale Beschreibung von Lautstärkeänderungen etwas schwierig ist, haben wir die verschiedenen Lautstärkeverläufe in Bild 2 dargestellt. Dabei werden nur die Hüllkurven für Werte der vier Bits zwischen 8 und 15 gezeigt. Die Lautstärkenverläufe mit Werten zwischen 0 und 7 stellen eine Kopie derselben dar. Um eine Hardwarehüllkurve zu verwenden, muß diese also zunächst durch Setzen der Register 11 bis 13 definiert sein. Dann kann sie durch Setzen von Bit 4 in den Lautstärkeregistern aufgerufen werden. Allerdings sieht man sich hier vor das erste Problem gestellt. Da nur eine Hüllkurve festgelegt werden kann, ist man bei mehrstimmigen Musikstücken klanglich sehr eingeschränkt. Deshalb wurde bei der Entwicklung des Schneider-Basics auf die Softwarelösung zurückgegriffen. Dies soll uns jedoch nicht daran hindern, auch einmal die Wirkungen des direkten Zugriffs auszuprobieren.

Zunächst müssen wir aber klären, wie wir überhaupt auf die Register des Soundchips zugreifen können. Dazu existiert im Betriebssystem des Computers die sehr vielseitige Maschinenroutine MC SOUND REGISTER, die für unsere Zwecke geeignet ist. Sie erlaubt einen problemlosen Zugriff auf die verschiedenen Register des Chips. Dabei muß der Akkumulator, also das Register A des Z80, die Nummer des gewünschten Tonregisters enthalten. Das Register C die zu sendenden Daten. Wir benötigen also ein kleines Maschinencode-Programm, das diese zwei Register lädt, die Routine im Betriebssystem aufruft und anschließend wieder ins Basic-Programm zurückkehrt.

Wenn Sie aus diesem Grund noch ein wenig Basic für den Komfort addieren, erhalten Sie das kleine Programm »Soundregister« (Listing 5). Das Programm ist schnell erklärt. Zur Ablage eines Maschinenprogramms benutzen wir den Speicherbereich von 42000 nach oben. Das Maschinenprogramm selbst stellt uns vor keine großen Probleme. Es besteht aus den vier Maschinencode-Befehlen:

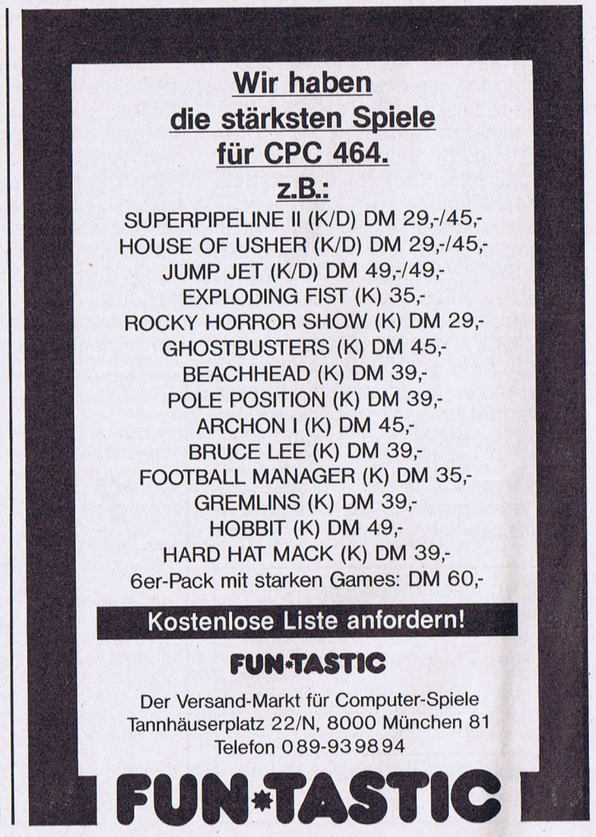
LD A, 0	3E 00
LD C, 0	0E 00
CALL MC SOUND REGISTER	CD 34 BD
RET	C9

Anstelle der beiden Nullen müssen wir nun nur noch das gewünschte Register, beziehungsweise den zu sendenden Wert, POKEn. Dies geschieht nach der INPUT-Abfrage in Zeile 90. Es folgt der Aufruf unserer Routine und ein Rücksprung nach Zeile 70 für weitere Experimente.

Wenn Sie beim Setzen der Periode eines Tongenerators trotz aufgedrehten Lautstärkereglers nicht gleich einen Ton aus dem Lautsprecher vernehmen, so haben Sie wahrscheinlich das entsprechende Bit im Freischaltregister noch nicht gelöscht. Besonders bei der Eingabe dieses Registers sollten Sie sich an eine schöne Eigenschaft des CPC erinnern: Die Eingabe von Binärzahlen mittels des vorangestellten Kürzels &X. Damit können Werte Bit für Bit bestimmt werden. Besonders beim Beschreiben des Freischaltregisters ist das von großem Nutzen. Um Ihnen gleich ein anfängliches Erfolgserlebnis zu ermöglichen, hier einmal eine sehr interessante Kombination:

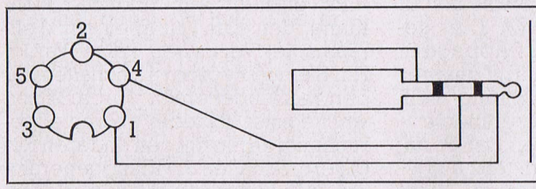
Register	Wert
0	65
7	&X00110110
8	&00010000
13	14
11	255

Es entsteht ein Science-fiction-Klang, den man gut in einem Weltraumspiel verwerten kann. Variieren Sie nun Register 11, schalten Sie den Rauschgenerator durch Setzen von Register 7 wieder ab und experimentieren Sie mit den anderen Registern. Schalten Sie Klanggeneratoren zu und wieder ab und erleben Sie völlig ungeahnte Soundkombinationen. Spätestens wenn Sie dies durchgespielt haben, sollte Ihnen klar sein, warum es sich auch beim CPC lohnt, in die Maschinenebene hinabzutauchen, denn diese Klänge sind mit Basic-Befehlen zum großen Teil nicht darzustellen, denn da benötigen wir den Direktzugriff auf den Soundchip. (Carsten Straush)



Krach in Stereo: Ein Soundkabel für den Schneider

Musik kann der Schneider in Stereo ausgeben. Aber wie? Über den eingebauten Lautsprecher sicher nicht. Man braucht ein Verbindungskabel zur Stereoanlage.



So müssen der Klinken- und der DIN-Stecker verbunden werden

In jeder Werbung von Schneider findet sich der dezente Hinweis, daß der Computer Musik in Stereo ausgeben kann. Doch leider sagt einem niemand wie. Auch das Handbuch enthält hier nur einen untergeordneten Hinweis der Form: Benutzen Sie ein Stereoverbindungskabel. Damit kann man als Laie natürlich recht wenig anfangen. Dabei ist es schon mit einem relativ geringen Kostenaufwand (zirka 6 Mark) möglich, sich ganz neue klangliche Dimensionen zu erschließen. Töne von verschiedenen Seiten, die sich sogar scheinbar von einer Seite zur anderen bewegen können, erzielen nämlich speziell bei Aktion- und Sportspielen (Ballwechsel) ganz tolle Effekte, die erheblich zur Realitätsnähe oder zum (gewollten)»Schockeffekt« beitragen. Mit einem vollen Sound gewinnt auch so mancher Flugsimulator immens an Realität. Alles was man dazu braucht, ist ein Kabel mit den richtigen Steckverbindungen und natürlich die heimische Stereoanlage.

Stereo, das heißt zunächst einmal rechts und links. Man benötigt also zwei Lautsprecher. Da man beim CPC allenfalls über ein »halbes«Exemplar dieser Gattung verfügt, ist man auf eine externe Klangwiedergabe angewiesen. Dies stellt uns grundsätzlich vor keine größeren Schwierigkeiten, denn mit einer Stereoanlage haben wir ja bereits ein brauchbares Instrument im Haus. Auch auf der anderen Seite gibt es keine Probleme. Der Schneider verfügt nämlich bereits über eine Buchse mit der Bezeichnung I/O, mit der die Tonsignale, die wir für die Ansteuerung unserer Stereoanlage benötigen, zur Verfügung stehen. Es geht jetzt also nur noch darum, die Anlage mit dem Computer zu verbinden.

Da der Computer als Ausgabegerät fungiert, müssen bei der Stereoanlage die Eingänge beschaltet werden. Sie können dabei sowohl den Auxiliary-Eingang (AUX), als auch den Tonband-Eingang (TB) benutzen. Sollte der Verstärkerteil Ihrer Stereoanlage relativ unempfindlich sein (leiser Ton), können Sie auch den Mikrofon-Eingang verwenden.

Als nächstes müssen wir uns mit der Auswahl der benötigten Stecker beschäftigen. Auf der Seite des Schneiders ist dies einfach. Sie brauchen in jedem Fall nur einen 3,5 mm-Stereo-Klinkenstecker. Das Schwergewicht des Wortes liegt dabei auf Stereo. Es gibt auch Mono-Klinkenstecker, aber mit denen funktioniert das Ganze nicht. Betrachten wir nun die andere Seite. Welche Stecker Sie für ihre Stereoanlage brauchen, hängt davon ab, welches Gerät Sie benutzen.

Grundsätzlich gibt es zwei Normen, die in Deutschland verbreitet sind. Es ist dies die japanische Norm, die mit Cinch-Steckern arbeitet, und die deutsche DIN-Norm (fünfpoliger DIN-Stecker). Daneben gibt es natürlich auch noch Geräte, die über Buchsen für beide Steckverbindungen verfügen. Hier ist es dann natürlich egal, welches Kabel Sie basteln.

Je nach Gerätetyp brauchen Sie also entweder zwei Cinch-Stecker oder einen fünfpoligen DIN-Stecker. Zur Verbindung reicht ein zweiadriges abgeschirmtes Kabel. Zuerst entfernen Sie an den Kabelenden die Isolation, den Plastikmantel. Nun stoßen Sie auf ein silbriges Drahtgeflecht, die Abschirmung. Von dieser durch zwei weitere Plastikschläuche getrennt, bleiben die beiden Übertragungsleitungen übrig.

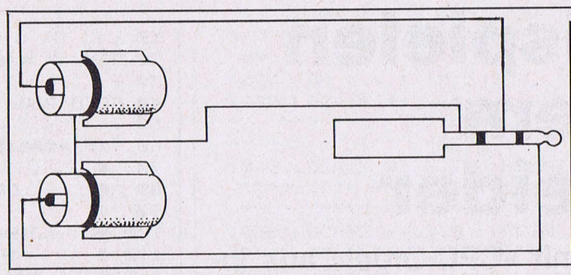
Die Beschaltung ist nun relativ einfach. Die Stereoklinke verfügt über drei Anschlußpole: zwei Blechlaschen mit Loch, an die die beiden Übertragungsleitungen angeschlossen werden, und ein etwas größeres Metallstück, das auch zum Einklemmen des Kabels dient und mit der Abschirmung verbunden wird. Damit sind wir auf der Seite des Computers mit allen Lötarbeiten fertig. Als nächstes sollten Sie nun die Schutzkappe auf das Kabel aufziehen und auf den Stecker aufschrauben. Am Rande noch eine Anmerkung für Techniker: Obwohl der CPC mit einer Klinkenbuchse ausgestattet ist, von der man normalerweise erwartet, daß ihr Ausgang relativ niederohmig ist, liegt der Ausgangswiderstand doch mit 22 kΩ ziemlich hoch. Ein Direktanschluß eines Lautsprechers bringt daher kein Ergebnis.

Wenden wir uns der anderen Seite zu. Wenn Sie mit Cinch-Steckern arbeiten, ist die Sache einfach. Das zweiadrige abgeschirmte Kabel wird in der Mitte getrennt und je eine Leitung und die dazugehörige Abschirmung werden in einen Cinchstecker geführt. Dieser hat zwei Anschlüsse, einen für die Abschirmung (mit dem äußeren Metallzylinder verbunden), der andere in der Mitte geht auf den Stift. Um die Auswahl, welcher Draht dabei zu welchem Kanal (rechts oder links) gehört, brauchen wir uns bei den Cinchsteckern nicht besonders zu kümmern. Wir können jederzeit noch nachträglich an der Stereoanlage die beiden Stecker vertauschen.

Etwas komplizierter sieht das Ganze dagegen bei Verwendung eines DIN-Steckers aus. Hier werden alle Informationen, das heißt, das Tonsignal für den rechten Kanal, das Signal für den linken Kanal und auch Masse, in einem Stecker übergeben. Das Ganze vereinfacht sich jedoch dadurch erheblich, daß jeder Stift des Steckers mit einer Nummer (von 1 bis 5) versehen ist. Sie können diese Nummern erkennen, wenn sie von vorne auf die Stifte des Steckers schauen. Sie erhalten dann im Gegenuhrzeigersinn die Nummern 1, 4, 2, 5, 3. Uns interessieren hier nur die beiden Eingänge (Stift 1 und Stift 4) sowie Masse (Stift 2). Die Abschirmung wird auf Masse gelegt, das heißt mit dem mittleren Stift verbunden. Die Beschaltung der Pole 1 und 4 hängt nun davon ab, welche Leitung Sie bei der Stereoklinke für die Spitze benutzt haben. Diese muß mit Stift 1 verbunden werden; für die andere bleibt dann nur noch Pin 4 übrig. Die Schaltpläne geben die Verdrahtung für beide Varianten wieder. Nachdem Sie das Kabel zusammengelötet haben, können Sie nun zum Experimentieren übergehen. Um die korrekte Beschaltung zu überprüfen, reichen drei einfache Soundkomandos. Geben Sie zunächst »SOUND 4,1000,100« ein.

Sie müßten jetzt einen Ton in mittlerer Tonhöhe auf dem rechten Kanal hören. Nun das Ganze mit »SOUND 1,100,100«.

Derselbe Klang taucht jetzt genauso im linken Kanal auf. Geben Sie nun SOUND 2,100,100, so tönt es Ihnen aus beiden Lautsprechern gleichzeitig mit gleicher Intensität entgegen. Einige weitere Programmbeispiele und Anwendungen finden Sie in dem Artikel über Sound. (Carsten Straush)



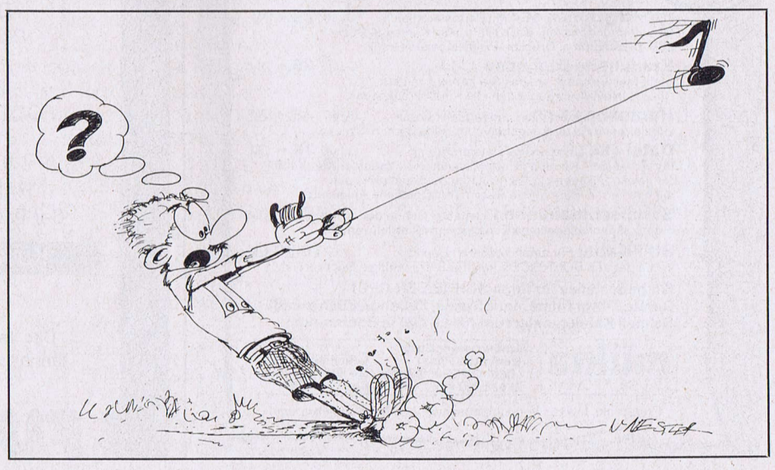
Klinken- und Cinch-Stecker sind einfach zu verbinden

3 m abgeschirmtes zweiadriges Kabel

1 Stereo-Klinkenstecker 3,5 mm

1 DIN-Stecker fünfpolig oder 2 Cinch-Stecker

Die Bauteile für das Verbindungskabel zur Stereoanlage



CPC 464 FORTH

Erleben auch Sie auf Ihrem CPC 464 die Geschwindigkeit eines schnellen FORTH-Compilers mit Turtlegrafik, Editor, Assembler, Tracer und De-Compiler. Dieses System ist im neuesten FORTH83-Standard geschrieben und erzeugt kompakte Programme. Die Grafik ist um Kreis und Füllbefehle erweitert, das System setzt ebenfalls Windows ein.

Das Programm wird mit einem 180seitigen deutschem Handbuch geliefert.

Preis auf Kassette DM **148,—**
auf Disk 3″, 5,25″ DM **178,—**

Ausführliches Prospektblatt bei:

FORTH-SYSTEME Angelika Flesch
Postfach 12 26, 7820 Titisee-Neustadt, Tel. 0 76 51/16 65

CPC 464 und 664
3-D-Schach mit Sprachausgabe

Dieses SUPER-Schachprogramm mit brillanter räumlicher Darstellung des Schachbrettes und Sprachausgabe des Zuges setzt neue Maßstäbe. Das Schachbrett kann um 360 Grad gedreht werden. Extrem spielstarkes Programm mit 7 Spielstufen. Das Programm kann auf Tastendruck auf dem Diskettenlaufwerk des CPC 664 gespeichert werden. Wechsel der Stufe und der Seite während des Spiels. Änderung der Positionen zu jeder Zeit. Änderung der Farben des Brettes und der Figuren. Große Eröffnungsbibliothek. Zugempfehlung. Durch Spielstufe ∅ auch für den Anfänger geeignet. **Preis: 59.—** (inkl. MwSt.) zuzüglich Versandkosten.

ERC-SOFT, E. Reitemann, Füllenbachstr. 11, 4 Düsseldorf 30, Telefon: 02 11/67 59 83

ES IST SOWEIT!

AMSTRAD

COMMODORE 64

SPECTRUM

Der neue Katalog 5 ist da!
Vollgestopft auf 72 Seiten mit Software, Hardware und Büchern für nebenstehende Computer. Einfach 1.80 DM in Briefmarken schicken und der Katalog kommt.

THOMAS WAGNER SOFTWAREVERSAND
Postfach 112 243, 8900 Augsburg

Vortex-Produkte für den SCHNEIDER CPC 464

- 5¼-Zoll-Diskettenstation mit Netzteil sep. Controller CPM 2.2* und einem Laufwerk **1198 DM**
- 5¼-Zoll-Diskettenstation mit Netzteil sep. Controller CPM 2.2* und zwei Laufwerken **1598 DM**
- Controller mit Kabel und CP/M 2.2* **548 DM**
- Programm **PARA**. Mit diesem Programm können Sie alle Diskettenparameter menügesteuert installieren und Disketten von z.B. Alphatronic, Osborn, Kaypro usw. direkt lesen und auf das gewünschte Format übertragen. **58 DM**
- Programm **Graphik Master** ist ein leistungsfähiges Programm zum Erstellen von Bildern und Grafiken auf dem Schneider CPC. Menügesteuert können Grafikfunktionen verwendet werden, die das Erstellen und Ausdrucken von Bildern erheblich vereinfachen. Alle Bilder können auf Diskette gespeichert werden. **79 DM**
- Drucker **Okimate 21**. PC-Farbdrucker in Verbindung mit dem Programm »Graphik Master« die ideale Ergänzung zu Ihrem Schneider CPC 464. (Lieferbar auch mit seriel. und IBM-Schnittstelle.) **899 DM**
- **Speichererweiterung** aufrüstbar auf insgesamt 512 KByte (64 KB stehen dem Benutzer für CP/M 2.2* und der Rest für z.B. eine RAM-Disk zur Verfügung). ab **170 DM**
- **Adapterkabel**. Zum Anschluß einer 3-Zoll-Schneider-Floppystation an die Vortex 5¼-Zoll-Version. **58 DM**
- **In Kürze erhältlich:** Expansion Unit mit 8 Steckplätzen und RS232-Karte.

BASF Floppy-Disk und Winchester-Laufwerke

BASF 6128 48 tpi, 5¼ Zoll, 2x40 Spuren	**435 DM**
BASF 6138 96 tpi, 5¼ Zoll, 2x80 Spuren	**439 DM**
BASF 6238 96 tpi, 5¼ Zoll, 4x80 Spuren	**850 DM**
BASF 6185 mit 15 MByte unformatiert	**1448 DM**
BASF 6185 mit 27 MByte unformatiert	**2198 DM**
2x BASF 6138 im Gehäuse mit Stromversorgung	**998 DM**

Alle angegebenen Preise inkl. Mehrwertsteuer

G+R Siemens Micro-Computer-Service GmbH
Leobener Str. 28, 7000 Stuttgart 30
Telefon 07 11/85 90 88, nach 17 Uhr 07 11/8 56 71 37

Orgelspielen mit dem Schneider

Machen Sie mit »CPC-Orgel« aus Ihrem CPC ein elektronisches Musikinstrument. Dreistimmige Melodien können komponiert und gespielt werden.

Mit »CPC-Orgel« können Sie auf der Tastatur des Schneider dreistimmige Melodien spielen. Das Programm ist sehr benutzerfreundlich gebaut und erklärt sich von selbst. Ein bißchen Theorie ist aber schon notwendig, wenn man interessante Tonfolgen erzeugen will. Aber dafür finden Sie in diesem Heft eine Einführung in die Soundprogrammierung des Schneider-Computers.

Nach dem Start des Programms müssen Sie die aktuelle Uhrzeit eingeben. Diese ist ständig auf dem Bildschirm eingeblendet — wichtige Termine werden also nicht vergessen. Nach dem Titelbild zeigt der Bildschirm die Tastenbelegung, sowie Daten über Hüllkurve, Auswahlpunkte und Parametereinstellung. Sie können Ihr erstes Musikstück auf dem Computer spielen. (Oliver Heggelbacher)

st,mi,se	: aktuelle Uhrzeit
n (x,y)	: Feld für die Tonperiode der Note x in der Oktave y
k (x,y)	: Feld für die Tastaturabfrage und Zuweisung der entsprechenden Note und Oktave
w	: Ton-Hüllenkurvenwiederholung (0=AUS/4=EIN)
o	: Oktave
ra	: Rauschen
hk (x,y)	: Parameter für die Lautstärken-Hüllenkurve
tk (x,y)	: Parameter für die Ton-Hüllenkurve
i	: Tonkanal (1, 2 oder 4)
sa	: Schrittanzahl
sg	: Schrittgröße
pl	: Pausenlänge

Variablenliste

50- 220	: Autorennachweis und Erläuterung
260- 290	: Uhrzeitangabe
350- 540	: Einlesen der Daten und Dimensionierung der Felder
560- 690	: Definieren der Sonderzeichen
730- 860	: Titelbild
950-1110	: Definieren der Windows und Zeichnen der »Klaviatur«
1150-1170	: Anfangswerte setzen
1210-1220	: Definieren der Hüllkurven
1280-1380	: Zeichnen der Hüllkurven und des Menüs
1420-1430	: Tonkanal festlegen und Repeat-Funktion heruntersetzen
1440-1480	: Hauptschleife: Tastaturabfrage und Spielen des Tons
1540-1960	: Parameter verändern
2020-2120	: Eingabe der Hüllkurvenparameter
2160-2190	: Ein ungültiger Wert wurde eingegeben
2230	: Löschen der Windows 2 bis 4 und Rücksprung ins Hauptprogramm
2270-2290	: Neue Uhrzeit errechnen und anzeigen
2330-2380	: INPUT-Routine

Was wird wo gemacht — die wichtigsten Programmzeilen

```
10 :
20 REM  ~ = CTRL + "
30 :
40 GOTO 260
50 :
60  ############################
70  #                          #
80  #       CPC-Orgel          #
90  #                          #
100 #      (C) 1985 by         #
110 #                          #
120 #     Andromeda Soft       #
130 #                          #
140 #     geschrieben von:     #
150 #                          #
160 #   Oliver Heggelbacher    #
170 #                          #
180 #      Steibensteg 8       #
190 #                          #
200 #      7778 Markdorf       #
210 #                          #
220 ############################
230 :
240 :       Eingabe der Uhrzeit
250 :
260 INK 0,19 : INK 1,0 : PAPER 0 : PEN 1
 : MODE 2 : INPUT "Uhrzeit (Stunden,Minu
ten,Sekunden) ";st,mi,se
270 IF st>23 OR MI>59 OR se>59 OR st<0 O
R mi<0 OR se<0 THEN 260
280 EVERY 50 GOSUB 2270
290 GOTO 350
300 :
310 :      einlesen der Datas
320 :
330 : und definieren der Sonderzeichen
340 :
350 DIM n(12,7) : FOR t = 0 TO 11 : pf=0
 : FOR i = 0 TO 7 : READ a : pf=pf+a : n
(t,i)=a
360 NEXT i : READ s : IF pf=s THEN NEXT
t  ELSE BORDER 0 : MODE 1 : PRINT "Zeile
 ";310+t*10;" wurde falsch eingegeben" :
 PRINT : END
370 DATA 3822,1911,956,478,239,119,60,30
,7615
380 DATA 3608,1804,902,451,225,113,56,28
,7187
390 DATA 3405,1703,851,426,213,106,53,27
,6784
400 DATA 3214,1607,804,402,201,100,50,25
,6403
410 DATA 3034,1517,758,379,190,95,47,24,
6044
420 DATA 2863,1432,716,358,179,89,45,22,
5704
430 DATA 2703,1351,676,338,169,84,42,21,
5384
440 DATA 2551,1276,638,319,159,80,40,20,
5083
450 DATA 2408,1204,602,301,150,75,38,19,
4797
460 DATA 2273,1136,568,284,142,71,36,18,
4528
470 DATA 2145,1073,536,268,134,67,34,17,
4274
480 DATA 2025,1012,506,253,127,63,32,16,
4034
490 DIM k(50,1) : FOR t = 0 TO 50 : k(t,
0)=12 : NEXT t
```

Listing. »CPC-Orgel«


```
500  FOR t= 0 TO 2 : FOR i = 0 TO 11 : R
EAD b$ : b=ASC(UPPER$(b$))-44 : k(b,0)=i
 : k(b,1) = t : NEXT i : NEXT t
510 k(3,0)=0 : k(3,1)=3
520 DATA q,2,w,3,e,r,5,t,6,y,7,u,i
530 DATA 9,o,0,p,@,^,[,a,z,s,x,c
540 DATA f,v,g,b,n,j,m,k,",",l,"."
550 :
560 SYMBOL AFTER 48
570 SYMBOL 48,126,66,66,66,66,66,126,0
580 SYMBOL 49,2,2,2,2,2,2,2,0
590 SYMBOL 50,126,2,2,126,64,64,126,0
600 SYMBOL 51,126,2,2,126,2,2,126,0
610 SYMBOL 52,66,66,66,126,2,2,2,0
620 SYMBOL 53,126,64,64,126,2,2,126,0
630 SYMBOL 54,126,64,64,126,66,66,126,0
640 SYMBOL 55,126,2,2,2,2,2,2,0
650 SYMBOL 56,126,66,66,126,66,66,126,0
660 SYMBOL 57,126,66,66,126,2,2,126,0
670 SYMBOL 123,4,10,10,10,12,8,24,30
680 SYMBOL 124,41,41,41,30,8,8,56,56
690 SYMBOL 126,255,255,255,255,255,255,2
55,255
700 REM
710 REM       Titelbild
720 REM
730 ENV 1,15,1,1,3,-1,1,1,0,10,12,-1,8 :
 ENV 3,6,2,1,6,-1,1
740 MODE 0 : INK 0,0 : INK 1,6 : INK 2,1
9 : INK 3,11 : BORDER 0 : PAPER 0 : CLS

750 PEN 1 : LOCATE 6,3 : PRINT "CPC-Orge
l"
760 PEN 2 : LOCATE 6,6 : PRINT CHR$(164)
;" 1985 by"
770 PEN 1 : LOCATE 4,8 : PRINT "Andromed
a Soft"
780  PEN 2 : LOCATE 1,8 : PRINT "{" : LO
CATE 1,9 : PRINT "!"
790 PEN 3 : LOCATE 2,12 : PRINT"geschrie
ben von"
800 LOCATE 1,14 : PRINT "Oliver Heggelba
cher"
810 PEN 1 : LOCATE 4,20 : PRINT "Taste d
ruecken"
820 x=6 : i=1
830 IF x<6 THEN i=1 : x=6 ELSE IF x>25 T
HEN i=-1 : SOUND 1,119,0,0,1
840 x=x+i : INK 1,x
850 FOR t = 1 TO 40 : NEXT t
860 IF INKEY$="" THEN 830 ELSE INK 1,22
: GOTO 950
870 :
880 :
890 :     Hauptprogramm
900 :
910 :
920 ###
930 ###    Zeichnen des Keyboards
940 ###
950 CLS : MODE 2
960 WINDOW #1,1,80,1,16 : WINDOW #2,1,20
,17,25 : WINDOW #3,21,40,17,25 : WINDOW
 #4,41,80,17,25
970 WINDOW #5,71,78,2,2 : uhr=1
980 INK 0,0 : INK 1,13 : PAPER #1,0 : PE
N #1,1 : CLS #1 : PAPER #2,1 : PEN #2,0
: CLS #2 : PAPER #3,0 : PEN #3,1 : CLS #
3 : PAPER #4,1 : PEN #4,0 : CLS #4
990 PAPER #5,1 : PEN #5,0 : CLS #5
1000 LOCATE #1,18,1 : PRINT #1,"C P C -
O R G E L    -     KEYBOARD"
1010 LOCATE #1,2,3 : PRINT #1,"~~~ 2 ~~
3 ~~~ ~~~ 5 ~~ 6 ~~ 7 ~~~ ~~~ 9 ~~ 0 ~~~
 ~~~ ^ ~~~"
1020 FOR t = 1 TO 3
1030 LOCATE #1,2,3+t : PRINT #1,"~~~   ~
~   ~~~ ~~~   ~~   ~~   ~~~ ~~~   ~~   ~
~ ~~~   ~~~" : NEXT t
1040 LOCATE #1,2,7 : PRINT #1,"~~~~ ~~~~
 ~~~~ ~~~~ ~~~~ ~~~~ ~~~~ ~~~~ ~~~~ ~~~~
 ~~~~ ~~~~"
1050 LOCATE #1,2,8 : PRINT #1,"~~q~ ~~w~
 ~~e~ ~~r~ ~~t~ ~~y~ ~~u~ ~~i~ ~~o~ ~~p~
 ~~@~ ~~[~"
1060 LOCATE #1,10,11 : PRINT #1, "{" : L
OCATE #1,10,12 : PRINT #1,"!"
1070 LOCATE #1,20,10 : PRINT #1," a ~~ s
 ~~~ ~~~ f ~~ g ~~~ ~~~ h ~~ j ~~ k ~~~
 ~~~"
1080 FOR t = 1 TO 3
1090 LOCATE #1,20,10+t : PRINT #1,"   ~~
   ~~~ ~~~   ~~   ~~~ ~~~   ~~   ~~   ~~
~ ~~~" : NEXT t
1100 LOCATE #1,20,14 : PRINT #1,"  ~~~~
 ~~~~ ~~~~ ~~~~ ~~~~ ~~~~ ~~~~ ~~~~ ~~~~
 ~~~~"
1110 LOCATE #1,20,15 : PRINT #1,"  ~~z~
 ~~x~ ~~c~ ~~v~ ~~b~ ~~n~ ~~m~ ~~,~ ~~.~
 ~~/~"
1120 REM
1130 REM  Anfangswerte der Huellkurven
1140 REM
1150 w=0 : o=3 : ra=0
1160 FOR t = 1 TO 4 : FOR i = 1 TO 3 : R
EAD a : hk(t,i)=a : NEXT i : NEXT t
1170 DATA 15,1,1,3,-1,1,1,0,13,12,-1,15
1180 REM
1190 REM   Einstellen der Huellkurven
1200 REM
1210 ENV 2,hk(1,1),hk(1,2),hk(1,3),hk(2,
1),hk(2,2),hk(2,3),hk(3,1),hk(3,2),hk(3,
3),hk(4,1),hk(4,2),hk(4,3)
1220 ENT 2-w,tk(1,1),tk(1,2),tk(1,3),tk(
2,1),tk(2,2),tk(2,3),tk(3,1),tk(3,2),tk(
3,3),tk(4,1),tk(4,2),tk(4,3)
1230 REM
1240 REM  Zeichnen der Huellkurven
1250 REM
1260 REM  und des Menues
1270 REM
1280 PRINT #2," Lautst.-Huellk." : PRINT
 #3," Ton-Huellkurve" : LOCATE #4,1,2 :
PRINT #4," ; = Lautst.-Huellkurve aender
n"
1290 PRINT #4," \ = Ton-Huellkurve aende
rn" : PRINT #4," ] = Oktave aendern" : P
RINT #4," } = Geraeusch-Periode aendern"
1300 LOCATE #4,2,7 : PRINT #4,"Ton-Huell
k. Wiederh. =  ";CHR$(24); : IF w=4 THEN
 PRINT #4,"EIN" ELSE PRINT #4,"AUS"
1310 PRINT #4,CHR$(24)
1320 LOCATE #4,16,8 : PRINT #4, "Oktave
= ";o : PRINT #4,"    Geraeusch-Periode
= ";ra
1330 m=0 : FOR t = 1 TO 4 : m=m+hk(t,3)
: NEXT t : IF m>0 THEN k=120/m ELSE k=1
1340 MOVE 10,0 : FOR t = 1 TO 4 : DRAWR
hk(t,3)*k,hk(t,2)*hk(t,1)*6,0 : NEXT t

1350 FOR t = 1 TO 4 : vw(t)=ABS(tk(t,2))
*tk(t,1) : NEXT t : n=MAX(vw(1),vw(2),vw
(3),vw(4)) : IF n>0 THEN p=50/n ELSE p=1

1360 MOVE 169,64 : FOR t = 1 TO 4 : DRAW
R 30,tk(t,2)*tk(t,1)*p,1 : NEXT t
```

```
1370 PLOT 0,146 : DRAW 639,146
1380 SOUND 1,284,0,0,1
1390 REM
1400 REM      Hauptschleife
1410 REM
1420 j=1
1430 SPEED KEY 255,255
1440 a$=INKEY$ : IF a$="" OR a$<CHR$(44)
 OR a$=CHR$(127) THEN 1440 ELSE IF a$="
" THEN 1540 ELSE IF a$="\" THEN 1700 ELS
E IF a$="]"THEN 1840 ELSE IF a$="}" THEN
 1930
1450 a=ASC(UPPER$(a$))-44 : b=k(a,0) : c
=k(a,1)+o
1460 IF n(b,c)<>0 THEN SOUND 128+j,n(b,c
),0,0,2,2,ra
1470 j=j*2 : IF j>4 THEN j=1
1480 GOTO 1440
1490 +++
1500 +++       veraendern der
1510 +++
1520 +++    Lautstaerken-Huellkurve
1530 +++
1540 FOR t = 1 TO 10
1550 LOCATE #2,1,10 : PRINT #2
1560 NEXT t
1570 CLS #4 : PRINT #4,"Lautst.-Huellkur
ve"
1580 FOR t = 1 TO 4
1590 GOSUB 2020 : IF sa*sg>15 OR sa*sg<-
15 OR p1<0 OR p1>125 THEN 1590
1600 hk(t,1)=sa : hk(t,2)=sg : hk(t,3)=p
1
1610 NEXT t
1620 p=0 : FOR t = 1 TO 4 : p=p+hk(t,2)*
hk(t,1) : NEXT t
1630 IF p>15 OR p<0 THEN GOSUB 2160 : GO
TO 1580
1640 GOTO 2230
1650 %%%
1660 %%%       veraendern der
1670 %%%
1680 %%%       Ton-Huellkurve
1690 %%%
1700 FOR t = 1 TO 10
1710 LOCATE #3,1,10 : PRINT #3
1720 NEXT t
1730 CLS #4 : PRINT #4,"Ton-Huellkurve"
1740 FOR t = 1 TO 4
1750 GOSUB 2020 : IF sa<0 OR sa>127 OR s
g<-128 OR sg>127 OR p1<0 OR p1>125 THEN
1750
1760 tk(t,1)=sa : tk(t,2)=sg : tk(t,3)=p
1
1770 NEXT t
1780 LOCATE #4,2,8 : PRINT #4,"Wiederhol
ung (J/N) "
1790 a$=INKEY$ : IF UPPER$(a$)="J" THEN
w=4 ELSE IF UPPER$(a$)="N" THEN w=0 ELSE
 GOTO 1790
1800 GOTO 2230
1810 ###
1820 ###     veraendern der Oktave
1830 ###
1840 CLS #4 : PRINT #4,"Oktave veraender
n"
1850 LOCATE #4,1,3 : PRINT #4,"Oktave (0
-4) "; : GOSUB 2330 : o=eing
1860 IF o<0 OR o>4 THEN GOSUB 2160 : GOT
O 1840
1870 GOTO 2230
1880 \\\
1890 \\\     veraendern der
1900 \\\
1910 \\\  Geraeusch-Periode
1920 \\\
1930 CLS #4 : PRINT #4," Geraeusch-Perio
de veraendern"
1940 LOCATE #4,2,3 : PRINT #4," Geraeusc
h-Periode (0-31) "; : GOSUB 2330 : ra=ei
ng
1950 IF ra<0 OR ra>31 THEN GOSUB 2160 :
GOTO 1930
1960 GOTO 2230
1970 ***
1980 ***   Unterprogramm zur Eingabe
1990 ***
2000 ***       der Parameter
2010 ***
2020 LOCATE #4,1,3 : PRINT #4,t;". Absch
nitt"
2030 FOR i = 4 TO 6
2040 LOCATE #4,19,i : PRINT #4,"
   " : NEXT i
2050 LOCATE #4,2,4 : PRINT #4,"Schrittan
zahl   : ? "; : GOSUB 2330 : sa=eing
2060 IF sa<0 THEN 2030
2070 LOCATE #4,19,4 : PRINT #4," "
2080 LOCATE #4,2,5 : PRINT #4,"Schrittgr
oesse : ? "; : GOSUB 2330 : sg=eing
2090 LOCATE #4,19,5 : PRINT #4," "
2100 LOCATE #4,2,6 : PRINT #4,"Pausenlae
nge   : ? "; : GOSUB 2330 : pl=eing
2110 LOCATE #4,19,6 : PRINT #4," "
2120 RETURN
2130 REM
2140 REM  falsche Parametereingabe
2150 REM
2160 LOCATE #4,2,8 : PRINT #4,"geht nich
t!!"
2170 FOR t = 1 TO 1000 : NEXT t
2180 LOCATE #4,2,8 : PRINT #4,"
   "
2190 RETURN
2200 ^^^
2210 ^^^  Zurueck zum Hauptprogramm
2220 ^^^
2230 FOR t = 2 TO 4 : CLS #t : NEXT t :
GOTO 1210
2240 #
2250 #   Uhrzeit anzeigen
2260 #
2270 se=se+1 : IF se>59 THEN se=0 : mi=m
i+1 : IF mi>59 THEN mi=0 : st=st+1 : IF
st>23 THEN st=0
2280 IF uhr=1 THEN LOCATE #5,1,1 : PRINT
 #5,USING "##:##:##";st;mi;se;
2290 RETURN
2300 @@@
2310 @@@  INPUT-Routine
2320 @@@
2330 eing$="" : x=0 : PRINT #4," _";CHR$(
8);
2340 a$=INKEY$ : IF a$="" THEN 2340
2350 IF a$=CHR$(127) THEN IF x>0 THEN x=
x-1 : eing$=LEFT$(eing$,x) : PRINT #4,CH
R$(8);"  ";CHR$(8);CHR$(8); : GOTO 2340
ELSE GOTO 2340
2360 IF a$=CHR$(13) THEN eing=VAL(eing$)
 : PRINT #4," " : RETURN
2370 IF (ASC(a$)<48 OR ASC(a$)>(57)) AND
 a$<>"-" THEN 2340
2380 x=x+1 : SOUND 130,284,0,0,3 : PRINT
 #4,a$;" ";CHR$(8); : eing$=eing$+a$ : G
OTO 2340
```

Listing. »CPC-Orgel« (Schluß)

HAPPY COMPUTER PROGRAMM-SERVICE

Programme aus früheren Ausgaben

Commodore 64

Alle 12 Programme auf Diskette für den Commodore 64.
Bestell-Nr. LH 8509 A, DM 29,90*, sFr. 24,90*

Schnelle Grafik aus dem Compiler
Strukturiertes Programmieren mit Grafik- und Mathematik-Befehlen, ein Hauch von Pascal und dazu noch ein ordentlicher Schuß Geschwindigkeit: Schon ist ein Compiler erster Güte gezimmert. **Listing des Monats** aus der **Ausgabe 8/85**.

Mondlandung
Ein klassisches Computer-Spiel, das die Landung einer Raumfähre auf dem Mond simuliert. Der Schwierigkeitsgrad kann durch Änderung einer einzigen Basic-Zeile beliebig verändert werden. Aus **Ausgabe 8/85**.

Komfort-Routinen
Die Befehle »Auto, Delete und Trace« helfen ganz besonders beim Abtippen längerer Basic-Programme. Aus **Ausgabe 8/85**.

Grafik-Hardcopy
Druckerroutine für den Ausdruck von Grafik-Hardcopies mit dem MPS 802. Aus **Ausgabe 8/84**.

Psycho
Ein Abenteuerspiel, das sich aus verschiedenen Grafiken und Texten zusammensetzt. Beweisen Sie Ihre Nervenstärke und entfliehen Sie der magischen Burg Towerhill. Aus **Ausgabe 8/85**.

Tab-Calc
Ein Tabellenkalkulationsprogramm mit dem Sie Ihre Finanzlage leichter im Griff behalten. Aus **Ausgabe 8/85**.

Aus Ausgabe 9/85
Woodshot, Sprite Mover, Short Save, Sprite-Dreher, Echtzeituhr, Animation

Sonderheft: Spectrum

Kassette/Bestell-Nr. LH 8551 D, DM 19,90*, sFr. 16,90*

Commodore 64

Alle 5 Programme auf Diskette für den Commodore 64.
Bestell-Nr. LH 8507 A, DM 29,90*, sFr. 24,90*

Risiko
Taktisches Spiel mit dem Ziel, die Welt von den feindlichen Armeen zu befreien. Grafische Darstellung der Weltkarte mit den durch farbliche Kennzeichnung dargestellten Armeen der verschiedenen Parteien. Aus **Ausgabe 7/85**.

Mini-Grafik
Mit Hilfe der Sprite-Technik wird auf dem C 64 ein Grafik-Fenster erzeugt, das sich ideal zur Gestaltung von Textprogrammen oder Spielen eignet, die Text und Grafik gemeinsam auf einem Bildschirm benutzen wollen. Aus **Ausgabe 7/85**.

Muso
Ein Musikstück wird in die Interrupt-Routine des C 64 eingebunden und dudelt dann während der Arbeit am Computer so vor sich hin. Das Musikstück komponiert ständig neue Melodien selbständig nach harmonischen Grundsätzen, das heißt, es kommt kein falscher Ton aus dem Commodore. Aus **Ausgabe 7/85**.

Maskenbildner
Mit Maskenbildner kann man den Bildschirm mit dem C 64 beschreiben. Das, was dann zu sehen war, wird durch das Programm Maskenbildner in ein Basic-Programm mit Print-Anweisungen umgewandelt und kann dann in jedes Programm eingemischt werden. Aus **Ausgabe 7/85**.

Aller Anfang ist schwer
Hier wird speziell für Anfänger der Dateiaufbau und deren Anwendung erklärt. Als Beispiel dient eine Adreßverwaltung. Aus **Ausgabe 7/85**.

Schneider CPC 464

Disassembler
Wer Software entwickeln und dazu in Maschinensprache programmieren will, der braucht das richtige Werkzeug. Mit dem hier vorgestellten Disassembler werden Sie schon bald das Innenleben Ihres Schneider-Computers begreifen. Aus **Ausgabe 5/85**.

Grafik
Mit unserem Listing des Monats, einem komfortablen Grafikprogramm, läßt sich die »Malerei« – auch für jeden Anfänger – bestens in den Griff bekommen. Aus **Ausgabe 4/85**.

Dateiverwaltung
Damit in Ihrem Archiv immer Ordnung herrscht. Eine universelle Dateiverwaltung für jeden Anwendungszweck. Aus **Ausgabe 4/85**.

Alle 3 Programme auf Kassette für den Schneider CPC 464.
Bestell-Nr. LH 8505 G, DM 29,90*, sFr. 24,90*

Text
Mit diesem Textverarbeitungsprogramm können Sie problemlos Ihre gesamte Korrespondenz erledigen. Aus **Ausgabe 3/85**.

Gespensterjagd
Das schnelle Reaktionsspiel zeigt die Basic-Fähigkeiten des CPC 464. Aus diesem Grund ist die Gespensterjagd ein gutes Anwendungsbeispiel für eine Schneider-spezifische Programmgestaltung. Aus **Ausgabe 2/85**.

Alle 2 Programme auf Kassette für den Schneider CPC 464.
Bestell-Nr. LH 8503 G, DM 29,90*, sFr. 24,90

Schneider CPC 464

Alle 4 Programme auf Kassette für den Schneider CPC 464.
Bestell-Nr. LH 8508 G, DM 29,90*, sFr. 24,90*

BW-COM-Compiler
Fast so schnell wie Assembler werden Ihre Basic-Programme mit diesem **Listing des Monats** aus der **Ausgabe 7/85**, ohne daß Sie in Maschinensprache programmieren müssen.

CHAIN MERGE
Zwei Programme vom Diskettenlaufwerk zu laden und zu verbinden, das geht mit AMSDOS nur unvollständig. Diese kleine Routine hilft weiter. **Tips & Tricks-Listing** aus der **Ausgabe 6/85**.

Protokollfunktion
Das Betriebssystem des Schneider CPC zu erweitern ist nicht schwer. Eine sinnvolle Ergänzung finden Sie hier mit diesem Programm, das alle Bildschirmausgaben auf dem Drucker mitprotokolliert. Aus **Ausgabe 8/85**.

Schneider-Kurs
Mit den Tips & Tricks-Listing aus der **Ausgabe 6** und **7/85** finden Sie sich sicherlich sehr schnell in dem Innenleben Ihres Schneider-Computers zurecht.

Atari

Magic Painter
Unser Listing des Monats in der **Ausgabe 3/85** ist ein Grafikprogramm, das sich mit anderen Malprogrammen dieser Art durchaus messen kann.

Grafikdemo
Alle 256 Farben werden auf dem Bildschirm dargestellt. Eine Farbspielerei, die die hervorragenden Grafikfähigkeiten der Atari-Computer beweist (Rainbow-Effekt). Aus **Ausgabe 3/85**.

Variablen-Dump
Mit diesem Programm können Sie die verwendeten Variablen eines anderen Programms auf dem Bildschirm listen. Aus **Ausgabe 2/85**.

Wie die Bilder laufen lernen
Mit dem Utility »Power-Mover« können Sie laufende Bilder schnell und problemlos erzeugen. Aus **Ausgabe 2/85**.

Statuszeile mit Uhr
Mit diesem Programm können Sie eine zusätzliche Statuszeile oberhalb des Bildschirms generieren. Aus **Ausgabe 1/85**.
Bestell-Nr.: LH 8503B DM 29,90*, sFr. 24,90*

Commodore 64

Alle 7 Programme auf Diskette für den Commodore 64.
Bestell-Nr. LH 8506 A, DM 29,90*, sFr. 24,90*

Maze
Aus **Ausgabe 6/85**.
Grafische Impressionen
Aus **Ausgabe 6/85**.
Happysynth
Listing des Monats – aus Ausgabe 6/85.
Dasher
Listing des Monats – aus Ausgabe 5/85.
Alle Neune
Aus **Ausgabe 5/85**.
Renumber 64
Aus **Ausgabe 5/85**.
Fensterl-Künstler
Tips&Tricks-Listing – aus Ausgabe 5/85.

Commodore 64

Lonely Driver
Aus **Ausgabe 4/85**.
Das Haus des Magiers
Aus **Ausgabe 4/85**.
Kalte Zeiten
Aus **Ausgabe 2/85**.
Rettet den letzten Baum
Aus **Ausgabe 2/85**.
Optik mit Simons Basic
Aus **Ausgabe 3/85**.
Software-Basic 3.0
Aus **Ausgabe3/85**.
Niemandsland
Aus **Ausgabe 3/85**.
Der rasende Raider
Aus **Ausgabe 3/85**.

Alle 8 Progr. a. Diskette f. d. Commodore 64.
Bestell-Nr. LH 8504 A, DM 29,90*, sFr. 24,90*

* Alle Preise inklusive Mehrwertsteuer, unverbindliche Preisempfehlung. Listing-Service-Produkte sind nur für Endkunden, nicht für Wiederverkäufer.

Bestellungen aus der Schweiz richten Sie bitte direkt an: Markt&Technik Vertriebs AG, Kollerstr. 3, CH-6300 Zug, Tel.: 042/223155.
Bestellungen aus anderen Ländern bitte per Auslandspostanweisung! Achtung: Nicht die eingeheftete Zahlkarte verwenden!
Bestellungen aus Österreich richten Sie bitte direkt an:
Bücherzentrum Meidling, Schönbrunner Str. 261, A-1120 Wien, Tel. 0222/833196,
microcomput-ique, E. Schiller, Fasangasse 21, A-1030 Wien, Tel. 0222/785661

Bitte verwenden Sie für Ihre Bestellung die eingeheftete Postscheck-Zahlkarte zur Überweisung des Rechnungsbetrags. Sie erleichtern uns die Auftragsabwicklung und Sie sparen sich die Versandkosten!

Ihr Schneider spielt Bach

Zeigen Sie doch Ihren Bekannten einmal, daß auch Ihr Computer das Bach-Jahr zu würdigen weiß.

Die Sound-Fähigkeiten der Schneider-Computer sind vielfältig, werden aber fast nie gewürdigt. Das Programm »Bach« spielt die Kantate »Wohl mir, daß ich Jesum habe«. Damit der Bildschirm nicht zu nüchtern aussieht, wird zuerst die Orgel des Straßburger Münster gezeichnet.

Im zweiten Teil des Programms steht dann die Kantate. Zeile 330 legt die Tonhöhe (h) und Geschwindigkeit (s) fest. Die verschiedenen Teile des Musikstücks stehen in Unterprogrammen, die über »GOSUB«-Befehle aufgerufen werden. Orgel und Chor unterscheiden sich durch die Klangfarben. Diese sind in dem Befehl »ENV« definiert, die DATA-Anweisungen steuern die Tonhöhe (a,c,e) und -länge (b,d,f). (Wolfgang Volz)

Variablenliste

ka, kb, kc, kd, ke, kf	Koordinaten für den Orgelkasten
breite	Breite der Orgelpfeifen
px, py	x(y)-Koordinaten für die Orgelpfeifen
hoehe	Höhe der Orgelpfeifen
spy	y-Koordinaten für den Pfeifenfuß
h	Tonhöhe (Oktave)
s	Geschwindigkeit
a, c, e	Tonhöhen
b, d, f	Tonlängen

```
10 '**********************************
20 '**Orgel im Strassburger Muenster**
30 '**              und               **
40 '** Wohl mir,dass ich Jesum habe **
50 '**          (BWV 147)             **
60 '**                                **
70 '***              von              ***
80 '****       Wolfgang Volz         ****
90 '****       Laemmerstieg 58       ****
100 '***       2400 Luebeck 1        ****
110 '**********************************
120 CLS
130 INK 0,0:INK 1,26:INK 2,14:INK 3,6:BO
RDER 0:PAPER 0:MODE 1
140 LOCATE 10,1:PRINT"Johann Sebastian B
ach"
150 LOCATE 3,25:PRINT">> Wohl mir,dass i
ch Jesum habe <<"
160 '
170 'Orgelkasten
180 '
190 RESTORE 210
200 FOR kz=1 TO 2:READ kc,kd:PLOT kc,kd,
1:FOR kn=1 TO 35:READ ka,kb:DRAWR ka,kb,
1:NEXT kn,kz
210 DATA 100,50,0,200,100,0,0,10,-86,0,-
14,-10,100,0,0,-200,54,-10,104,0,54,10,1
00,0,0,200,-14,10,-86,0,0,-10,100,0,-100
,0,0,30,-54,50,0,40,-104,0,0,-40,-54,-50
,0,-100,54,10,104,0,54,-10,0,70,0,-200,-
54,-10,0,330,-104,0,0,-330,-54,10,-100,0
220 DATA 98,48,0,204,104,0,0,10,-88,0,-1
6,-12,104,0,0,-202,54,-10,102,0,54,10,10
2,0,0,200,-16,10,-86,0,0,-6,98,0,-100,0,
0,32,-54,50,0,36,-100,0,0,-36,-54,-50,0,
-102,54,10,100,0,54,-10,0,70,0,-202,-54,
-10,0,328,-100,0,0,-328,-54,8,-102,0
230 FOR ke=0 TO 10 STEP 2:PLOT 98+ke,252
+ke,1:DRAW 200,252+ke,1:NEXT
240 FOR kf=0 TO 10 STEP 2:PLOT 514-kf,25
0+kf,1:DRAW 410,250+kf,1:NEXT
250 '
260 'Orgelpfeifen
270 '
280 FOR pf=1 TO 29:READ breite,px,py,hoe
he,yfu
290 FOR a=px TO px+breite:PLOT a,py,2:DR
AWR 0,hoehe:PLOT  px+breite/2,yfu:DRAW a
,py,2:PLOT a,py,3:DRAWR 0,10,3:PLOT a,py
+5,0:NEXT a,pf
300 DATA 8,104,80,90,54,8,118,75,110,54,
8,132,70,130,54,8,146,65,150,54,8,160,70
,130,54,8,174,75,110,54,8,188,80,90,54,6
,208,70,190,54,6,220,65,210,52,6,232,61,
230,50,6,244,59,248,48,8,260,70,250,44,8
,274,65,270,44,8,288,60,290,44,8,302,55,
310,44
310 DATA 8,316,60,290,44,8,330,65,270,44
,8,344,70,250,44,6,362,59,248,48,6,374,6
1,230,50,6,386,65,210,52,6,398,70,190,54
,8,416,80,90,54,8,430,75,110,54,8,444,70
,130,54,8,458,65,150,54,8,472,70,130,54,
8,486,75,110,54,8,500,80,90,54
320 '
330 'Kantate
340 '
350 s=2:h=2:'Schnelligkeit und Tonhoehe
360 RESTORE 700
370 GOSUB 620:'Takt 1 bis 7
380 GOSUB 620:GOSUB 800:'Takt 8
390 FOR wdh=1 TO 2
400 RESTORE 960
410 GOSUB 860:'Takt 9-11 bzw. 24-26
420 GOSUB 990:'Takt 12 bzw. 27
430 GOSUB 620:GOSUB 800:'Takt 13 bzw. 28
440 GOSUB 860:'Takt 14-16 bzw. 29-31
450 GOSUB 990:'Takt 17 bzw. 32
460 RESTORE 720:GOSUB 620:'Takt 18-23 bz
w. 33-38
470 NEXT
480 GOSUB 620:RESTORE 1160:GOSUB 800:'Ta
kt 39
490 GOSUB 860:'Takt 40-42
500 GOSUB 990:'Takt 43
510 GOSUB 620:'Takt 44-45
520 GOSUB 860:'Takt 46-48
530 GOSUB 990:'Takt 49
540 GOSUB 620:'Takt 50
550 GOSUB 800:'Takt 51
560 GOSUB 860:'Takt 52-54
570 GOSUB 990:'Takt 55-56
580 GOSUB 860:'Takt 57-59
590 GOSUB 990:'Takt 60
600 GOSUB 620:'Takt 61-70
610 GOTO 1490:'Takt 71
620 ENV 1,10*s,-1,10
630 READ a:IF a=-1 THEN RETURN
640 SOUND 1,a/h,30*s,9,1
650 FOR z=1 TO 3:READ c
660 SOUND 2,c/h,10*s,10,1
670 NEXT
680 GOTO 630
690 'Takt 1
700 DATA 1276,0,319,284,638,253,213,239,
758,239,190,213
710 'Takt 2-7,18-23,33-38
720 DATA 1012,213,159,169,758,159,213,25
3,1517,319,284,253
730 DATA 1136,239,213,190,1012,213,239,2
53,956,284,253,319
740 DATA 851,338,319,284,676,426,338,284
,851,239,253,284
750 DATA 638,253,319,284,758,253,213,239
,956,239,190,213
760 DATA 1012,213,159,169,758,159,213,25
3,851,319,284,253
770 DATA 956,379,213,239,902,253,284,319
,851,426,319,338,-1
780 'Takt 8,39(1-3)
790 DATA 1276,319,253,213,-1
800 READ a,c:IF a=-1 THEN RETURN
810 SOUND 1,a/h,10*s,9,1
820 SOUND 2,c/h,10*s,10,1
830 GOTO 800
840 'Takt 8
850 DATA 0,159,0,213,638,253,851,319,851
,253,1012,213,-1,1
860 WHILE SQ(1)>127:WEND
870 READ a,b
880 IF a=-1 THEN RETURN
890 SOUND 1,a/h,b*s,3
900 READ c,d
910 SOUND 4,c/h,d*s,4
920 READ e,f
930 SOUND 2,e/h,f*s,4
940 GOTO 870
950 'Takt 9-11,24-26
960 DATA 1276,30,319,30,253,30,676,30,31
9,30,253,30,758,30,319,30,239,30
970 DATA 676,30,284,30,213,30,758,30,319
,30,213,30,851,30,338,30,213,30
980 DATA 758,30,319,30,239,30,676,30,284
,30,239,30,638,30,319,30,253,30,-1,1
990 READ a:b=10
1000 IF a=-1 THEN RETURN
1010 SOUND 1,a/h,b*s,9,1
1020 READ c
1030 SOUND 2,c/h,10*s,10,1
1040 READ e
1050 SOUND 4,e/h,10*s,4
1060 GOTO 990
1070 'Takt 12-17,27-32
1080 DATA 851,0,284,851,426,284,851,379,
284,0,338,284,0,284,284,851,319,284,851,
284,1,851,239,1,851,253,1,-1
1090 DATA 1703,239,284,338,-1
1100 DATA 0,426,0,338,851,284,851,239,85
1,253,851,284,-1,1
1110 DATA 638,30,319,30,253,30,676,30,31
9,30,253,30,758,30,319,30,239,30
1120 DATA 506,30,319,30,213,30,1012,30,4
26,30,213,30,758,30,319,30,253,30
1130 DATA 956,14,379,14,284,14,956,8, 37
9,8, 253,8,956,8, 379,8, 239,8,851,30,33
8,30,253,30,1703,30,338,30,284,30,-1,1
1140 DATA 1276,319,319,1276,253,319,1276
,284,319,638,253,319,638,213,319,638,239
,319,758,239,1,758,190,1,758,213,1,-1
1150 'Takt 39(4-9)-71
1160 DATA 0,159,0,213,638,253,676,319,67
6,253,758,225,-1,1
1170 DATA 851,30,338,30,284,30,956,30,33
8,30,284,30,1012,30,301,30,253,30
1180 DATA 1136,30,284,30,239,30,568,15,2
84,15,239,15,638,15,284,15,239,15,716,15
,284,15,239,15,758,15,284,15,239,15
1190 DATA 851,30,284,30,253,30,1012,14,2
84,14,253,14,1012,8,284,8,239,8,1012,8,2
84,8,213,8,758,30,301,30,253,30,-1,1
1200 DATA 1136,284,284,1136,239,284,1136
,253,284,716,239,284,716,190,284,716,213
,284,851,213,1,851,179,1,851,190,1,-1
1210 DATA 956,190,142,150,716,142,190,23
9,758,284,253,239
1220 DATA 851,179,190,213,804,239,253,28
4,758,379,284,301,-1
1230 DATA 1136,30,284,30,239,30,568,30,2
84,30,239,30,506,30,319,30,213,30
1240 DATA 478,30,319,30,190,30,568,30,23
9,30,190,30,758,30,319,30,190,30
1250 DATA 716,30,358,30,213,30,851,14,28
4,14,213,14,851,8,284,8,190,8,851,8,284,
8,179,8,638,30,319,30,213,30,-1,1
1260 DATA 956,190,239,956,159,239,956,17
9,239,758,159,239,758,190,239,758,239,23
9,956,319,1,956,284,1,956,268,1,-1
1270 DATA 716,284,239,253,851,239,284,35
8,638,426,379,358,956,379,478,426,-1
1280 DATA 0,379,0,319,956,338,1012,319,1
012,253,851,284,-1,1
1290 DATA 1276,30,319,30,253,30,638,30,3
19,30,253,30,758,30,319,30,239,30
1300 DATA 1012,30,319,30,213,30,758,30,3
19,30,213,30,1517,30,319,30,213,30
1310 DATA 1136,15,319,15,239,15,1136,15,
338,15,239,15,1012,30,319,30,253,30,956,
30,319,30,253,30,-1,1
1320 DATA 851,338,284,851,426,284,851,37
9,284,0,338,284,0,284,284,851,319,284,85
1,284,1,851,239,1,851,253,1
1330 DATA 1703,239,1,1703,284,1,1703,338
,1,1,426,1,1,338,1,851,284,1,851,239,1,8
51,253,1,851,284,1,-1
1340 DATA 638,30,319,30,253,30,676,30,31
9,30,253,30,758,30,319,30,239,30
1350 DATA 506,30,319,30,213,30,1012,30,4
26,30,213,30,758,30,319,30,253,30
1360 DATA 956,14,379,14,284,14,956,8,379
,8,253,8,956,8,379,8,239,8,851,15,338,15
,284,15,851,15,379,15,284,15,1703,30,338
,30,284,30,-1,1
1370 DATA 1276,319,319,1276,253,319,1276
,213,319,1276,159,319,1276,213,319,1276,
253,319,1276,319,319,1276,253,319,1276,2
13,319,-1
1380 DATA 1276,176,213,253,1276,319,253,
213,1276,190,239,284
1390 DATA 1276,338,284,239,1276,213,253,
319,1276,379,319,253
1400 DATA 1276,239,284,338,1276,426,338,
284,1276,239,253,284
1410 DATA 1276,253,319,284,638,253,213,2
39,758,239,190,213
1420 DATA 1012,213,159,169,758,159,213,2
53,1517,319,284,253
1430 DATA 1136,239,213,190,1012,213,239,
253,956,284,253,319
1440 DATA 851,338,319,284,676,426,338,28
4,851,239,253,284
1450 DATA 638,253,319,284,758,253,213,23
9,956,239,190,213
1460 DATA 1012,213,159,169,758,159,213,2
53,851,319,284,253
1470 DATA 956,379,213,239,902,253,284,31
9,851,426,319,338,-1
1480 DATA 1276,30,319,30
1490 READ a,b,c,d
1500 SOUND 1,a/h,b*s,9,1
1510 SOUND 2,c/h,d*s,10,1
1520 GOTO 1520
```

Listing. Unser Beitrag zum Bach-Jahr: Die Kantate »Wohl mir, daß ich Jesum habe«

Basic Special: Besonderheiten und Neues

Das Schneider-Basic hat eine Reihe von Eigenschaften, die nicht im Standard-Basic vorhanden sind und im Handbuch nicht beschrieben werden. Wir zeigen, was diese Kommandos können.

Einer der Vorzüge der Schneider-Computer ist ihr ausgeklügeltes und teilweise weit ausgereiztes Basic. Dabei ist der Name Basic eigentlich schon nicht mehr zutreffend, da eine Reihe von Funktionen, die in dem, was man als Schneider-Basic bezeichnet, enthalten sind, von anderen Sprachen übernommen und eingepaßt wurden. Damit kann der Schneider mit einer Reihe von Fähigkeiten aufwarten, die man bis jetzt nur in anderen Hochsprachen vorfand. Es werden Hilfen zum strukturierten Programmieren gegeben, die eine deutliche Gliederung in den Programmen ermöglichen. Daneben sind im CPC-Basic auch bereits Multi-Programming-Ansätze beziehungsweise Multi-Tasking-Elemente (siehe Artikel zur Echtzeitverarbeitung, (Seite 67) enthalten. Das heißt: Es ist möglich mit dem Schneider mehrere Programme zur gleichen Zeit unabhängig voneinander ablaufen zu lassen. Die CPU bearbeitet dann scheinbar gleichzeitig verschiedene Aufgaben.

Um diese etwas weitergehenden Befehle und auch um Kommandos, die nicht im Handbuch stehen, soll es hier gehen. Beginnen wir zunächst einmal mit einigen Spezialitäten im Schneider-Basic. Eine sehr hübsche Methode ist, Variablentypen en bloc zu definieren. Was heißt das? Zunächst ein Beispiel. Definieren Sie einmal: »A=3.45«. Auf »PRINT A« erhalten Sie nun, wie nicht anders zu erwarten war, die Ausgabe »3.45«. Wenn Sie aber statt dessen »DEFINT A-Z« eingeben und die ganze Prozedur noch einmal wiederholen, so lautet jetzt das Ergebnis »3«. Unter dem normalen Namen »A« wurde jetzt eine Integervariable angelegt.

Wie kommt das zustande? Wenn Sie normalerweise eine Variable ablegen, so gibt es dazu drei Möglichkeiten. Zum ersten: Sie schreiben einfach eine Kombination von Buchstaben. In diesem Fall wird diese vom Computer als eine Real-Variable betrachtet, das heißt der CPC legt unter diesem Namen fünf Bytes im Speicher ab, in denen dann die Fließkommazahl binär gespeichert ist. Etwas anders verläuft die Abspeicherung, wenn Sie auf Integerwerte zurückgreifen. Hier werden nur zwei Bytes abgelegt und natürlich der Name. Ein Integerwert ist beim Schneider durch das nachgestellte »%« gekennzeichnet. Eine wieder etwas andere Variante erhalten wir bei der Definition eines Strings. Hier gibt das nachgestellte »$« zu erkennen, daß die Variable als String betrachtet werden soll. Die Anzahl der abzuspeichernden Bytes ergibt sich dabei aus der Länge des Strings.

Die unterschiedlichen Variablentypen benötigen also unterschiedliche Mengen von Speicherplatz und können auch unterschiedliche Funktionen wahrnehmen. Speziell bei größeren Variablenmengen, wie zum Beispiel im Bereich der Matrizenrechnung und ähnlichen Anwendungen, ist es daher sinnvoll, sich zu überlegen, welche Variablentype man denn nun wirklich benötigt. Bei zum Beispiel 1 000 abzuspeichernden Variablen bedeutet der Unterschied zwischen einem Real- und einem Integerfeld schon einen Speicherplatzverlust, beziehungsweise -gewinn von 3 000 Bytes. Wenn das Programm dann noch einigen reservierten Speicherplatz braucht und eventuell noch eine Reihe von Lade- und Speicheroperationen unumgänglich sind, kann es nicht nur vorkommen, daß sich die Bearbeitung der Variablen erheblich verlangsamt, sondern auch, daß das System wegen Überlastung überhaupt nicht mehr voran kommt.

Nun ist es aber relativ mühsam jede Variable einzeln auf einen Variablentyp zu definieren und außerdem: Wer schreibt schon gern im Programm die Variablennamen immer noch mit »%« und »$«? Dies kostet Speicherplatz und im Endeffekt auch Zeit, nicht nur zur Eingabe, sondern auch zur Fehlerkorrektur, falls der falsche Variablentyp verwendet wurde. Eine sehr effiziente Möglichkeit diese Probleme zu umgehen, bieten die DEF-Befehle des Schneider. Wir wollen uns ihre Wirkungsweise etwas näher anschauen.

Nehmen wir einmal an, wir brauchen für ein Programm zehn Stringvariablen, zehn Realvariablen und drei vom Typ Integer. Jetzt gibt es zwei Möglichkeiten. Wir definieren A%=123, B%=7 und so weiter, das heißt wir legen die Variablen einzeln mit an den Namen angehängten Variablentyp fest. Als Typ-Kürzel gelten dabei das »%«-Zeichen für Integer-Variablen, der »$« für Strings und — möglicherweise haben Sie es noch nie benutzt — das »!« als Definitionsmerkmal für einen Real-Wert. Zweitens — und dies ist sehr viel effizienter — wir greifen auf die DEF-Befehle zurück. Mit diesen ist es möglich, einem einzelnen Buchstaben oder einer Reihe von Buchstaben vorab einem Typ zuzuordnen. Alle Variablennamen, die diesen Buchstaben tragen, werden dann als String-, Real- oder Integervariablen gekennzeichnet. Wir brauchen dann also das »%«- oder das »$«-Zeichen nicht mehr einzugeben, denn der CPC nimmt eine automatische Definition vor.

Machen wir uns dies einmal anhand eines Beispiels klar. Wenn wir alle Variablen, die mit den Buchstaben A bis C beginnen, für Integervariablen benutzen wollen, so können wir das mit »DEFINT A-C« erreichen. Wenn wir nach diesem Kommando nun eine Variable definieren, zum Beispiel »A=3.24«, so ist nicht mehr der Wert 3,24 in A gespeichert, sondern nur noch 3. Der Grund? Mit DEFINT haben wir alle Variablen mit den Anfangsbuchstaben A bis C auf Integer festgelegt, damit natürlich auch A. Da Integervariablen keine Nachkommastellen haben, kommt es zu der beschriebenen Rundung. Gerundet wird dabei natürlich nicht nur bei der Definition einer Variablen mit festem Wert, sondern auch, wenn die Integervariable das Ergebnis einer Rechnung aufnimmt.

Beim Benutzen eines Namens, zum Beispiel A ohne angehängtes Typenkennzeichen, wird also die durch den DEF-Befehl vorgegebene Typenkennzeichnung benutzt. Dies gilt übrigens nicht nur für Integers. Wenn wir zum Beispiel definieren, »DEFSTR W-Z«, so werden alle Buchstabenkombinationen, die mit W bis Z beginnen, jetzt als Strings aufgefaßt. Dies hat nun einen gewaltigen Vorteil. Während wir nämlich bei einem normalen »W="ABCD"« einen »Syntax Error« erhalten, so dürfen wir diese Operationen nun ausführen. Die Eingabe des nachgestellten »$« ist nicht mehr nötig. Weniger Eingaben sind nötig, wodurch Platz im Listing gespart wird.

Dabei ist allerdings zu beachten, daß wir es im Endeffekt immer noch mit einem String zu tun haben, obwohl dieser kein Typenkennzeichen (»$«) mehr im Namen trägt. Intern wird die Variable aber weiterhin als String gespeichert.

Aller guten Dinge sind drei und so ist es natürlich auch möglich, einen Variablentyp auf Real zu definieren. Dies geschieht mit DEFREAL und die mit dieser Buchstabenkombination bezeichneten Variablen werden nun als Fließkommazahlen betrachtet.

Die Definition von Variablen mit den DEF-Befehlen hat nun aber Vor- und Nachteile. Zum einen ermöglicht sie eine sehr gute Übersicht, denn bereits am Programmanfang kann man festlegen, welche Variablentypen welchen Buchstaben zugeordnet sind. Man erhält damit eine klare Variablendefinition, bringt Ordnung ins Programm und darüber hinaus baut man auch Konfusionen ab, wie sie des öfteren entstehen, wenn mit verschiedenen Variablennamen ohne Konzept operiert wird. Man zwingt sich damit zu einiger Disziplin beim Programmieren. Wer allerdings diesen Schlendrian gewohnt ist, wird am Anfang mit derartigen Methoden Probleme haben.

Die Definition mit DEF-Kommandos ist aber nur eine Art Auffangdefinition. Man hat mit dieser Art der Bestimmung die Benutzung der Buchstaben durch andere Variablentypen nicht völlig ausgeschlossen, denn die normalen Typenkennzeichen haben immer noch Vorrang vor jeglicher DEF-Definition. Ein Beispiel soll dies klar machen. Definieren Sie einmal »DEFINT A-K«. Alle Variablen, die als ersten Buchstaben irgendeinen zwischen A und K aufweisen, werden nun als Integer aufgefaßt. Wenn Sie also »B=15.4« eintippen und sich danach B ausgeben lassen, so erhalten Sie das Ergebnis 15. Sie können jedoch auch anders vorgehen. Jetzt definieren Sie einmal »B!=15.4«. Wenn Sie sich nun B! ausdrucken lassen, so erhalten Sie 15.4, das heißt den eingegebenen Fließkommawert.

Unter dem Variablennamen ohne Typenkennzeichen sind jetzt Integers gespeichert. Bei angefügtem Typenkennzeichen wird nun eine Realvariable oder wieder ein String angelegt. Das Typenkennzeichen für die Realvariablen ist dabei wie oben schon gesagt, das etwas ungewohnte »!«.

Obwohl man diese Typoperationen grundsätzlich auch bei Strings benutzen kann, ist das Hauptanwendungsgebiet doch die schnelle Definition von Real- und Integervariablen. Oft steht man vor dem Problem, daß man eine ganze Reihe von Variablen eigentlich als Integers schreiben könnte, aber aus Faulheit das »%« wegläßt, was unnötig Speicherplatz vergeudet und natürlich die Bearbeitungsgeschwindigkeit vermindert.

Schnellere Bearbeitung

Sollte es einmal notwendig sein von Integer- auf Realvariablen oder umgekehrt überzugehen, weil zum Beispiel die Nachkommastellen in einer bestimmten Operation doch interessieren, so stehen zwei weitere Spezialkommandos des CPC zur Verfügung: CINT und CREAL. Diese beiden ermöglichen eine Konvertierung in Integer- beziehungsweise in Realwerte. Geben Sie einmal »S=3.14« ein. Wenn Sie nun »PRINT S« eingeben, so wird erwartungsgemäß S mit dem Wert 3.14 zurückgegeben. Sie können sich nun mit »PRINT CINT(S)« den Integerwert dieser Fließkommazahl ausgeben lassen oder damit auch eine neue Integer besetzen. Umgekehrt verläuft das Ganze mit CREAL.

Ein Punkt ist allerdings bei dieser Art der Definition noch zu beachten und wird Ihnen möglicherweise bei den ersten Testversuchen mit diesen Kommandos einige Probleme bereiten. Setzen Sie einmal nach dem Einschalten zwei Variablen (beispielsweise A und X) auf eine beliebige Zahl, zum Beispiel den Wert 27. Wenn Sie nun unser oben beschriebenes Kommando »DEFINT A-K« eingeben, hat A jetzt plötzlich den Wert 0, während X ohne Blessuren davongekommen ist. Des Rätsels Lösung ist relativ einfach. Mit den ersten Eingaben haben Sie zwei Real-Variablen definiert, da beim Einschalten alle Variablen ohne Typenkennung auf Real definiert sind. Nach dem DEFINT werden nun die Variablennamen mit Anfangsbuchstaben zwischen A und K als Integers aufgefaßt. Bei der Abfrage sucht der CPC also nach einer Integervariable namens A, da nur eine Fließkommazahl definiert wurde, findet er keine. Darum erfolgt die Ausgabe der Null.

Nichtsdestotrotz sind die Variablen immer noch vorhanden. Man muß sie jetzt nur mit »!« adressieren. »PRINT A!« bringt uns daher die 27 wieder zurück. Von derartigen kleinen Problemen sollten Sie sich aber auf keinen Fall abhalten lassen, diese Befehle anzuwenden.

Denn bei richtiger Nutzung bringen diese Kommandos neben einfacherer Eingabe und Variablenansprache Ordnung und Klarheit ins Programm.

Eine andere hübsche Möglichkeit, die der Schneider aus anderen Sprachen übernommen hat, ist die Programmierung von Schleifen unter einer Aussprungbedingung mit WHILE und WEND. Im »normalen« Basic programmiert man Schleifen meist mit FOR und TO, beispielsweise die Schleife

»FOR i=32 TO 255:PRINT i, CHR$(i):NEXT i«,

die den Zeichensatz ausgibt. Diese Art von Schleifenprogrammierung kann man immer dann anwenden, wenn die Anzahl der Schritte bekannt ist, das heißt wenn feststeht, daß eine Schleife n-mal durchlaufen werden muß. Problematisch wird es, wenn dies nicht bekannt ist, oder die relevante Größe erst in der Schleife selbst berechnet wird, wie es zum Beispiel bei jeder Art von Schachtelung der Fall ist. Durch Intervallschachtelung kann man zum Beispiel die Wurzel einer Zahl bestimmen oder auch den effektiven Zins eines Kredites. Allgemein ist zu sagen, daß derartige Annäherungsverfahren immer dann angewendet werden müssen, wenn zwar das Ergebnis und der Rechenweg zu diesem bekannt sind, aber eine Umkehrfunktion nicht bestimmt werden kann. Die zu diesem Ergebnis notwendige Eingabegröße kann nur geschätzt werden. Nun sind Schätzungen immer fehlerbehaftet, weshalb man einen maximalen Fehler vorgibt und den Eingabewert diesem immer mehr annähert. Dies kann nun mit WHILE und WEND sehr effektiv programmiert werden. WHILE legt die Bedingung fest, bis zu deren Eintritt die nachfolgende Prozedur wiederholt werden soll. WEND kennzeichnet das Ende des Vorgangs. Wenn man das Unterschreiten einer maximal erlaubten Fehlerabweichung (f) als Entscheidungskriterium wählt, so sieht eine solche Schleife im Prinzip wie folgt aus:

```
WHILE ABS(f)>0.01
.
.
.
bestimme neues x
.
.
.
f= 1-f(x)/y
WEND
```

Der Wert von 0.01 stellt dabei die Fehlergrenze (1 Prozent gleich 0,01) dar, f(x) und y die als bekannt vorausgesetzten Werte für die Funktion und das Endergebnis. f(x)/y gibt den Fehlerquotienten an, f die prozentuale Abweichung. Durch die Bedingung im WHILE-Kommando wird die nachfolgende Annäherung so lange durchgeführt, bis die gewählte Fehlerabweichung erreicht ist. Natürlich können die Kommandos WHILE und WEND auch für ganz banale Zwecke benutzt werden, zum Beispiel als Zeitschleife. »WHILE X<1000:X=X+1:WEND« ist deutlich klarer als FOR X=1 TO 1000:NEXT X. WHILE und WEND sorgen also für eine klare Struktur, da Anfang und Ende des Prozesses direkt erkennbar sind.

Nachdem wir uns zu Beginn ein wenig mit Tricks bei der effizienten, strukturierten Programmierung beschäftigt haben, gehen wir nun zu einigen Funktionen im CPC-Basic über, die nicht im Handbuch stehen. Einer der ersten Befehle, auf den man sofort stößt, wenn man sich mit dem Computer und speziell dem Basic-Interpreter etwas weitergehend beschäftigt, ist DEC$.

Die meisten Versuche, mit diesem Befehl etwas anzufangen, scheitern im Ansatz. Dabei handelt es sich eindeutig um ein Kommando. Wer noch über das alte englische Handbuch verfügt, findet sogar bei den Befehlen HEX$ und BIN$ einen Verweis auf diesen Befehl. Er muß also irgendetwas mit der Umwandlung von Zahlen in Strings zu tun haben. Dies erscheint etwas befremdend, wo doch die Umwandlung in Hex- und Binärwerte schon ausreichend abgedeckt ist.

Die Umwandlung in einen Oktalwert mit DEC$ zu beschreiben, wäre wohl nicht besonders sinnvoll. Irgendwie soll hier also ein String von einer Dezimalzahl erzeugt werden. Doch das ist nun wiederum etwas, was wir auch bereits als Funktion vorgegeben haben, nämlich bei STR$. Worum handelt es sich also? Die Antwort liefert der Basic-Interpreter und auch gleich den Grund, warum der Befehl beim CPC 464 nicht läuft. DEC$ bewirkt einen formatierten String einer Zahl, wie bei einer Ausgabe mit PRINT USING gewohnt. Allerdings ist den Entwicklern des Betriebssystems bei der Befehlsinterpretation ein Bug, ein Programmierfehler, unterlaufen. Es wird nämlich zweimal hintereinander getestet, ob eine Klammer geöffnet wurde — das ist einmal zu viel. Der Befehl führt daher bei normaler Definition als Funktion permanent zu einem Syntax-Error. Will man DEC$ beim CPC 464 ausführen, so muß man also zwei Klammern öffnen und eine schließen. Bei den Nachfolge-Modellen (CPC 664, CPC 6128) ist dieser Fehler behoben worden. Hier läuft der Befehl ganz normal mit einer Klammer am Anfang und einer am Ende.

Damit ist nun einiges über das Befehlsformat gesagt worden. Offen ist aber immer noch die Frage nach dem Sinn des Kommandos. Die Funktion des Befehls ist relativ einfach. DEC$ erzeugt einen formatierten Dezimalstring. Eine Dezimalzahl wird dabei in einem bestimmten Format ausgegeben, genauso, wie man es von PRINT USING gewohnt ist. Damit ist es möglich, Vorzeichenausgabe oder -unterdrückung, Nachkommanullen und Leerstellen bei der Ausgabe einer Zahl festzulegen und somit diese in einem bestimmten Format auszugeben, was besonders bei Listendarstellung sehr nützlich ist. Die bei anderen Computern notwendige Addition eines Leerstrings, um auf die benötigte Gesamtanzahl von Zeichen zu kommen, kann bei der Benutzung von DEC$ also entfallen. Dazu benötigt der Befehl zwei Angaben. An der ersten Stelle in der Klammer steht die zu formatierende Zeichenkette, darauffolgend die Formatierungszeichen. Dabei gelten dieselben Bedeutungen, die man schon von der Benutzung von PRINT USING her kennt. Das »#« steht als Platzhalter für eine Ziffernstelle. »####.##« bedeutet also die Ausgabe einer Zahl mit drei Vorkomma- und zwei Nachkommastellen. Dabei werden unbenutzte Vorkommastellen durch Leerzeichen aufgefüllt, die Nachkommastellen dagegen auf »0« gesetzt. PRINT DEC$((3.4,"####.##") ergäbe also als Ausgabe 3.40, wobei die gesamte Zahl um zwei Stellen nach rechts eingerückt dargestellt wird, was sich aus den ungenutzten zwei Vorkommastellen ergibt. Bei der Formatierung des Dezimalwertes ist man allerdings nicht auf stellenrichtige Ausgabe beschränkt. Weitere Möglichkeiten sind die vorangestellte oder nachgestellte Vorzeichenausgabe oder etwa die Ausgabe einer Zahl im Exponentialformat. Die Funktion der verschiedenen Platzhalter und Symbole finden Sie im Handbuch bei der Erklärung zu PRINT USING.

Der DEC$-Befehl ist nicht die einzige verborgene Funktion, auf die man bei einer näheren Analyse des CPC-Befehlssatzes trifft. Gerade wenn man häufiger mit Integervariablen arbeitet, ist eine weitere Funktion sehr nützlich, nämlich der »\«. Der CPC verfügt über zwei Möglichkeiten, eine Division durchzuführen. Die eine kennen Sie. Dies ist die normale Division von Real-Werten mit dem gewohnten Schrägstrich (»/«) als Kennzeichen. Es geht jedoch auch anders. Wenn Sie nämlich den auf der Taste rechts daneben liegenden Schrägstrich benutzen, so wird eine Integerdivision durchgeführt. Hier ist das Ergebnis eine Ganzzahl. Die Nachkommastellen werden unterdrückt. Das Resultat ist dabei dasselbe, als wenn man auf eine normale Division noch zusätzlich die Integerfunktion anwenden würde. Da man oftmals gar nicht daran interessiert ist (speziell im Bereich der Grafik bei der Bestimmung der Koordinaten für Punkte oder Linien), ein Ergebnis unter korrekter Bestimmung der Nachkommastellen auszurechnen, kann man hier auch gleich mit der Integerdivision arbeiten. Die Vorteile: die Division läuft schneller, die Grafikroutine braucht die Fließkommazahl nicht mehr in eine Integer umzurechnen, Zeit wird gespart, besonders wenn man in Basic komplizierte Figuren wie Kreise und Ellipsen zeichnet. Um einen Eindruck davon zu erhalten, welche Zeitersparnis sich erzielen läßt, führen Sie einmal den folgenden Test durch. Zunächst einmal zum »reinen« Unterschied von Integer- zu Realdivision.

```
10 e=TIME:FOR i=1 TO 1000:a=i/2:NEXT i:d=TIME:diff=(d-e)/300:PRINT" verstrichene Zeit';diff;'Sekunden"
20 e=TIME:FOR i=1 TO 1000:a=i\2:NEXT i:d=TIME:diff=(d-e)/300:PRINT"verstrichene Zeit ";diff; "Sekunden"
```

Als Ergebnis erhalten Sie 3.89 und 3.35 Sekunden. In jeder Schleife wurden dabei 1000 Divisionen auf beide Arten mit exakt denselben Zahlen durchgeführt. Die Variable TIME enthält als Systemfunktion die Zeit seit dem Einschalten in 1/300 Sekunden. Durch die Abfrage dieser Funktion vor und nach der Schleife ist es somit problemlos möglich, die verstrichene Zeit zu bestimmen. Die Integer-Division ist also bei der Anwendung auf Realwerte ungefähr 15 Prozent schneller. Dieses Ergebnis können wir noch erheblich verbessern, wenn wir die Variablen a und i auch als Integerwerte definieren.

Die neue Zeile 20

```
20 e=TIME:FOR i%=1 TO 1000:a%=i%/2:NEXT i%:d=TIME:diff=(d-e)/300: PRINT"verstrichene Zeit ";diff; " Sekunden"
```

benötigt jetzt nur noch 60 Prozent der maximalen Ausführungszeit. Durch die Anwendung der Integerdivision kann man also ein Programm erheblich schneller machen.

Der CPC verfügt aber nicht nur über die Möglichkeit einer Division ohne Nachkommastellen, sondern auch die Umkehrung dazu ist möglich. Während die Integerdivision nur die Vorkommastellen eines Ergebnisses errechnet und weitergibt, so interessiert sich die Modulofunktion für die umgekehrte Angabe. Hier interessiert nur der Rest. Nun werden Sie vielleicht fragen, wozu denn derartige Betrachtungen notwendig sind. Beim CPC hat die Modulofunktion eine sehr wichtige Bedeutung. Sie wird häufig im Betriebssystem angewandt, und zwar hauptsächlich, um beliebige Parameter in erlaubte Parameter umzuwandeln.

Bevor wir dies etwas näher betrachten, noch ein paar Worte zur Modulofunktion, die als solche im CPC-Basic bereits integriert ist. Während aber eine normale Funktion auf der einen Seite eines Gleichheitszeichens steht und in der Funktionsklammer die einzelnen Parameter beinhaltet, wird die Modulofunktion genauso wie die Integerdivision als Rechenzeichen behandelt. Das Symbol für die Modulofunktion ist MOD. Wenn Sie also zum Beispiel eingeben »PRINT 14 MOD 12«, so erhalten Sie als Ergebnis den Wert 2. 14 läßt sich also durch 12 teilen und ergibt den Rest 2, der dann als Ergebnis der Modulofunktion ausgegeben wird. Dieses muß dabei immer kleiner als der rechts von MOD angegebene Wert sein. Es ist daher mit Hilfe dieser Funktion leicht möglich, Werte aus einem beliebigen Wertebereich in einen erlaubten Bereich zu überführen.

Derartige Korrekturen sind auch unbedingt notwendig, denn stellen Sie sich einmal vor, was ohne derartige Korrekturen beim Schneider passieren würde, wenn Sie im Mode 2 PAPER 13 definieren. Das INK-Register Nummer 13 kann im hochauflösenden Modus nicht benutzt werden. Ohne Korrektur wäre der CPC also mit ungültigen Werten konfrontiert. Die Folge wäre eine Fehlermeldung: »IMPROPER ARGUMENT«. Deshalb wendet der Schneider auf die angegebenen Werte die Modulo-Funktion an (MOD 2 im Mode 2; MOD 4 im Mode 1 und MOD 16 im Mode 0). Damit erhält er dann erlaubte Werte (zwischen 0 und 1 im Mode 2, 0 bis 3 im Mode 1 und 0 bis 15 im Mode 0).

Aber auch als Basic-Programmierer kann man MOD gut verwenden, zum Beispiel bei der Korrektur von Bewegungen. Wenn beispielsweise in einem Action-Spiel ein auf der Grundlinie bewegtes Objekt, nachdem es am rechten Rand angekommen ist, wieder links starten soll, so können wir dies auf zwei Wegen erreichen. Zum einen: Mit einer IF-Abfrage feststellen, ob eine Überschreitung der linken Grenze (negativer Wert) oder des rechten Maximalwertes stattgefunden hat. In diesem Fall wäre dann die x-Koordinate auf 0, beziehungsweise bei Überschreitung der linken Grenze auf den Maximalwert zu setzen. Einfacher geht es jedoch mit MOD. Die Abweichung nach links ist kontrollierbar, indem man die X-Koordinate durch X=(X+640) MOD 640 bestimmt. Dann hat auch ein in diese Richtung flüchtendes Objekt keine Chance. (Carsten Straush)

Simultan auf zwei Hochzeiten: Echtzeitverarbeitung auf dem Schneider

Ein altes Sprichwort besagt, daß es unmöglich sei, auf zwei Hochzeiten gleichzeitig zu tanzen. Was für den Menschen gilt, stellt aber nicht unbedingt eine Schranke für den CPC. Er kann ohne Probleme durch zwei oder mehr Programme gleichzeitig »walzen«.

Eine der größten Neuerungen im Schneider-Basic, im Gegensatz zu anderen Versionen, ist die Fähigkeit des Computers, mehrere Aufgaben oder Programme »gleichzeitig« ausführen zu können. Das sind Techniken, die auf größeren Computersystemen mit den Namen Multi-Tasking und Multi-Programming verbunden sind. Multi-Tasking bezeichnet dabei, daß der Computer gleichzeitig mehrere verschiedene Aufgaben parallel ausführen kann. Beim Multi-Programming laufen zur gleichen Zeit zwei verschiedene Programme mit derselben CPU. Bei den meisten Heimcomputern sind diese Fähigkeiten nicht realisierbar. Dies hat seinen Grund darin, daß die Art, wie diese Computer Befehle abarbeiten, darauf angelegt ist, mehrere Kommandos nacheinander auszuführen. Beim CPC arbeitet das Betriebssystem zwar grundsätzlich ebenso, das Ende der Bearbeitung eines Befehls oder einer Reihe von Kommandos, muß aber nicht abgewartet werden. Es ist auch möglich während der Durchführung einer Anweisung kurz noch eine andere einzuschieben. Wichtiger ist jedoch, daß die Überleitung in einen anderen Programmteil nicht durch eine Anweisung des Programmierers direkt erfolgen muß, sondern vom CPC selbst gelenkt wird. Der Computer legt also selbst fest, wann er sich aus dem einen Programmbereich »verabschieden« will, um an einer anderen Stelle fortzufahren.

Was vielleicht noch interessanter ist: Er kehrt auch selbständig wieder an den Ort des Geschehens zurück. Und eine letzte Besonderheit: Der Computer legt für diese Hand-

lungen nicht seine eigene Arbeitszeit, die sich nach Takt- oder Befehlszyklen bemißt, zugrunde, sondern all diese Operationen sind in unserer Arbeitszeit konkret in 1/50 Sekunden, bestimmbar.

Dies ist besonders dann von großem Nutzen, wenn man zwei Programme, von denen das eine zeitkritisch abläuft, miteinander kombinieren muß. Bei diesem muß die Reaktion immer zu einem bestimmten Zeitpunkt erfolgen, während das andere Programm etwa unterschiedliche Zeiten für die Ausführung benötigt. In solch einem Fall ist es nicht mehr möglich, einfach von einem Programm ins andere und wieder zurück zu springen, da der Rücksprung zu verschiedenen Zeitpunkten erfolgen würde. Die Zeitsynchronisation im ersten Programm würde dadurch gestört. Wenn zum Beispiel ein Fernschreiber über einen Computer als Ausgabegerät angesteuert werden soll, so benötigt er präzise jede 1/50 Sekunde ein neues Bit. Beabsichtigt man während dieser Prozedur einen Text auf dem Bildschirm auszugeben, treten schon die ersten Probleme auf. Die Länge aller Zeilen eines Textes ist meist nicht identisch. Unterschiedlich lange Zeilen benötigen unterschiedliche Ausgabezeiten. Als Folge davon klappt die Synchronisation nicht mehr. Wenn man beispielsweise nach jeder Zeile ein Zeichen ausgeben will, so erfolgt der Ausdruck zu den unterschiedlichsten Zeiten — meist zu den falschen.

Beim CPC stellen derartige Schwierigkeiten kein größeres Problem dar. Das Lösungswort heißt Echtzeitverarbeitung. Die zugehörigen Basic-Befehle lauten EVERY, AFTER und REMAIN(...).

Zunächst zu den ersten beiden: EVERY und AFTER produzieren beide einen sogenannten Interrupt, das heißt den Abbruch eines Programms nach einer bestimmten vorher definierten Zeit. Das Programm wird jedoch nicht mit einem »Ready« beendet, sondern es erfolgt der Sprung in einen anderen Programmteil. Aus diesem können wir dann bei geeigneter Gestaltung der Befehle wieder in das Hauptprogramm zurückkehren. EVERY sorgt hier für eine immer wiederkehrende Unterbrechung, während AFTER nur einen einmaligen Interrupt erzeugt.

Basis beider Befehle sind vier Zeitgeber, sogenannte Timer, die von 0 bis 3 durchnumeriert sind. Sie wirken als setzbare elektronische Zählwerke. Allerdings ist dabei zu sagen, daß die Unterbrechungen der Timer hier keine elektronischen Signale sind, die von einem anderen Baustein an die CPU gegeben werden.

Das Betriebssystem bildet vielmehr die Unterbrechungsfunktionen über den Event-Mechanismus nach. Der CPC arbeitet mit einem Hauptunterbrechungstakt, den sogenannten Fast Ticker, in dessen Rhythmus (300 mal pro Sekunde) die CPU eine Unterbrechung ausführt. Dazu bricht sie die gerade bearbeitete Routine ab und geht in einen Teil des Betriebssystems, der sich mit zyklischen Aufgaben beschäftigt, Tätigkeiten also, die der Computer immer wieder vornehmen muß. Diese sind in sogenannten Event-Listen aufgeführt. Beim schnellen Takt von 300 Hertz (300 Unterbrechungen pro Sekunde) sind dies noch relativ wenige Aufgaben. Viel mehr hat der Computer beim Auftreten des eigentlichen Ticker (Low Ticker) zu tun. Dies ist ein langsamerer Takt (50 mal pro Sekunde), der durch Teilung aus dem Fast Ticker gewonnen wird und in dessen Rhythmus fast alle zyklischen Prozesse ablaufen. Eine der ersten Aktionen beim Auftreten des Tickers ist zum Beispiel die Tastaturabfrage. In diesem Takt werden auch unsere Timer weitergeschaltet und gegebenenfalls Unterbrechungen in Basic ausgeführt.

Zeit per Software

Auf die Software-Timer können wir mit den Kommandos EVERY und AFTER zurückgreifen. Die Berechnung der Zeiteinheiten geschieht dabei in Fünfzigstelsekunden. Geben wir also »AFTER 50,1 GOSUB 100« ein, so geht der CPC eine Sekunde, nachdem er diesen Befehl überlaufen hat, zur Zeile 100. Die erste Stelle des Kommandos gibt den Zeitwert in Fünfzigstelsekunden an, die zweite Stelle den benutzten Timer. Der Schneider führt dann die in dem Programm ab Zeile 100 angegebenen Kommandos aus. Trifft er auf ein RETURN, so kehrt er in das Hauptprogramm zurück. Das Prinzip demonstrieren die nachfolgenden Zeilen:

```
10    AFTER 50,1 GOSUB 100
20    PRINT "Hier läuft das Hauptprogramm"
30    GOTO 20
100   PRINT "Eine Sekunde ist vorüber !":RE-
      TURN
```

Am Beginn des Programms wird der Timer Nummer 1 auf 50 gesetzt. Danach läuft der CPC in eine fortlaufende Schleife in den Zeilen 20 und 30, die permanent den Text ausgeben. Nach einer Sekunde tritt dann der Interrupt ein. Das Programm ruft Zeile 100 auf und druckt den dort angegebenen Text aus. Danach geht es zurück ins Hauptprogramm.

Diese drei Zeilen bilden natürlich noch keine sinnvolle Anwendung. Dafür ist das Prinzip aber gut erkennbar. Aus einem Hauptprogramm erfolgt, nachdem die vorgegebene Zeit abgelaufen ist, der Sprung in die Unterroutine. Diese kann dann zusätzliche oder ergänzende Aufgaben wahrnehmen. Danach wird zu der Stelle zurückgesprungen, an der die Bearbeitung des Hauptprogramms abgebrochen wurde. AFTER sorgt für das einmalige Ablaufen dieses Vorgangs, mit EVERY wiederholt sich das Ganze regelmäßig. Um eine mit EVERY eingeleitete Unterbrechungsroutine zu beenden oder einen mit AFTER gesetzten TIMER vor dem Zeitablauf zu stoppen, besitzt das Schneider-Basic den Befehl REMAIN. Mit »REMAIN(0)« wird der Timer 0 zurückgesetzt und der Unterbrechungsvorgang damit gelöscht. »REMAIN(1)« bis »REMAIN(3)« setzen die anderen drei TIMER zurück. Mit »PRINT REMAIN("Timernummer")« ist es darüber hinaus möglich, sich jederzeit die Restlaufzeit eines Zählers ausgeben zu lassen — oder diese in eine andere Variable zu übernehmen. In jedem Fall wird aber der angesprochene Timer zurückgesetzt und damit wird die Unterbrechung nicht mehr ausgeführt. Kurz noch ein paar Worte zur Wahl des Timers. Grundsätzlich ist es egal, mit welchen man arbeitet, ob also Timer 0 oder Timer 2 das Signal zum Interrupt liefert.

Interessant wird es, wenn mehrere Zeitgeber gleichzeitig im Einsatz sind. Dann gibt es nämlich eine Hierarchie der Timer. Der Zeitgeber Nummer 3 hat dabei Priorität, nachfolgend die Nummer 2 und so weiter. Wenn also zwei Timer gleichzeitig eine Unterbrechung verlangen, so wird zuerst der Anfrage stattgegeben, die von dem Timer mit der höheren Nummer kommt. Erst nachdem dieser seine Routine durchgeführt hat, ist das nächste Unterbrechungsprogramm an der Reihe, bevor dann die Rückkehr in das Hauptprogramm erfolgt. Ein Beispiel:

```
10    AFTER 50, 1 GOSUB 100: AF-
      TER 50, 2 GOSUB 200
20    GOTO 20
100   PRINT »Hier 100«: RETURN
200   PRINT »Hier 200«: RETURN
```

Es wird mit zwei parallel gesetzten Zeitgebern gearbeitet. Beide Befehle verzweigen nach Ablauf der vorgegebenen Zeit in die zugehörigen Unterprogramme (ab Zeile 100 für Timer 1; ab 200 für Timer 2). Da Timer 1 zuerst gesetzt wird, sollte man erwarten, daß auch zuerst die Meldung »Hier 100« erscheint. Das Gegenteil ist der Fall. Da beide Timer zur selben Zeit eine Unterbrechungsanforderung ausgeben, wird zuerst die Anfrage mit der höheren Priorität berücksichtigt, in unserem Fall also die von Nummer 2.

Als Regel kann man sich daher einprägen, daß das Ergebnis, das am zeitkritischsten ist (von Zeitverschiebungen am meisten beeinflußt ist), mit dem Timer mit der höchsten Nummer gesteuert werden sollte.

Diese Befehle eröffnen uns nun weite Felder. Eine Standardanwendung der Echtzeitkommandos, speziell auch von EVERY, ergibt sich bei Programmen, die die Uhrzeit permanent anzeigen sollen. Jede Sekunde muß hier eine Weiterschaltung der Uhr erfolgen. Wenn man zum Beispiel die Uhrzeit ständig in einem Fenster in der rechten oberen Ecke angezeigt bekommen möchte, so wäre dies mit EVERY und dem Timer Nummer 2 zum Beispiel wie im Listing zu programmieren:

Das Programm selbst wirft keine Probleme auf. Durch das EVERY am Anfang wird der CPC angewiesen, jede Sekunde das Unterprogramm ab Zeile 40 anzuspringen, welches die Sekunden (se) hochzählt und gegebenenfalls die notwendigen Korrekturen bei Minuten und Stunden vornimmt. Bevor die Ausgabe erfolgt, werden darauf die drei Zahlenwerte durch eine Stringaddition zusammengebaut. Es sind noch zwei Punkte zu beachten. Zum ersten hat der Schneider die unangenehme Eigenschaft, bei der Umwandlung einer Zahl in einen String eine Leerstelle voranzustellen, so daß die eigentliche Zahl erst ab der zweiten Stelle im String beginnt. Da wir nur an dem Zahlenwert interessiert sind, muß dieser »innere Zahlstring« mit MID$ herausgelöst werden. Der

CPC und Multi-Tasking

nächste Punkt betrifft die Ausgabe des zusammengesetzten Gesamtstrings. Das Ausgabefenster ist auf acht Zeichen ausgelegt. Gibt man in einem Fenster einen String aus, der genauso lang wie die Windowbreite ist, so führt der CPC nach der Ausgabe einen Zeilenvorschub (line feed) aus. Da unser Window nur eine Zeile hoch ist, wird die gerade beschriebene Zeile sofort wieder entfernt. Mit angefügtem »;« nach dem PRINT-Kommando läßt sich dieses Ärgernis umgehen.

Die Anwendung der Befehle EVERY und AFTER ist jedoch nicht auf die Angabe der Uhrzeit beschränkt. Dies wird dann als Multi-Tasking bezeichnet, das heißt der Computer nimmt mehrere Aufgaben gleichzeitig wahr. Zum Beispiel könnte in einer langwierigen Berechnung zur selben Zeit jede Fünftelsekunde einmal die Tastatur abgefragt werden, um zu prüfen, ob bereits neue Anweisungen des Bedieners vorliegen. Durch diese Art der Programmgestaltung werden besonders rechenintensivere Programme sehr viel schneller. Im Spielebereich ist es möglich, Ereignisse durch den AFTER-Befehl erst mit einer gewissen Verzögerung auftreten zu lassen, während das Hauptprogramm an anderer Stelle bereits den Bildschirmaufbau für das nächste Bild vornimmt. Die normalerweise durch eine Zeitschleife nach dem Schema »FOR I=1 TO 1000:NEXT I« vertane Zeit, kann somit sinnvoll für weitere Aufgaben genutzt werden. Ein typisches Anwendungsbeispiel sind auch Erklärungstexte. Da der Computer schneller Daten ausgeben kann, als der Mensch aufnimmt, ist man normalerweise in solchen Programmen nach der Ausgabe einer Textseite gezwungen, in eine Zeit-, das heißt Warteschleife, zu gehen. Viel sinnvoller ist es jedoch, bereits jetzt den Text für die nächste Erläuterungsseite zusammenzusetzen oder Variablen vorzubereiten und nur von Zeit zu Zeit in einer Tastaturabfrageroutine zu prüfen, ob der Benutzer bereits fertig gelesen hat. Man kann die Echtzeitkommandos auch für die Feinabstimmung benutzen. Ein speziell im Spielebereich beliebter Effekt ist das Hereinrollen eines Titelbildes bei gleichzeitiger Ausgabe der Titelmelodie. Dabei sollte der Computer dann allerdings darauf achten, ob der Benutzer mit einem Tastendruck zu verstehen gibt, daß er diesen Titel nun schon zum 25. Mal gesehen hat und jetzt endlich spielen will. In diesem Fall müßte die Routine abgebrochen werden. Was auf den ersten Blick relativ einfach aussieht, ist auf den meisten Heimcomputern schwer zu realisieren und erfordert eine komplizierte Verschachtelung der drei Teilprogramme für die Text- beziehungsweise Titelausgabe, die Soundansprache und die Tastaturabfrage.

Mit den Echtzeitbefehlen geht dies jedoch relativ problemlos. Allerdings müssen wir dabei noch auf ein anderes Kommando zurückgreifen, welches mit der Echtzeitsteuerung im Soundbereich zusammenhängt: »ON SQ (Kanalnummer) GOSUB«. Dieses Namenskürzel steht für eine bedingte Unterbrechung. Falls in der Warteschlange für den angegebenen Kanal noch Platz für ein weiteres Sound-Kommando vorhanden ist, so erfolgt via Unterbrechung der Sprung in das nach dem GOSUB angegebene Unterprogramm. Kanal A hat dabei die Num-

```
10 ' ----------------------------
20 ' Uhrzeit-Programm mit EVERY
30 ' ----------------------------
40 CLS
50 WINDOW#7,33,40,1,1:INK 2,0:INK 3,21:P
APER#7,2:PEN#7,3
60 INPUT"Aktuelle Uhrzeit Stunden";st
70 INPUT"Aktuelle Uhrzeit Minuten";mi
80 INPUT"Aktuelle Uhrzeit Sekunden";se
90 GOTO 160
100 se=se+1:IF se>59 THEN se=0:mi=mi+1
110 IF mi=60 THEN mi=0:st=st+1
120 IF st=24 THEN st=0
130 z$=RIGHT$("0"+MID$(STR$(st),2),2)+":
"+RIGHT$("0"+MID$(STR$(mi),2),2)+":"+RIG
HT$("0"+MID$(STR$(se),2),2)
140 PRINT#7,z$;
150 RETURN
160 EVERY 50,2 GOSUB 100
170 ' Hier beginnt das eigentliche Progr
amm
180 ' Die Uhr laeuft unabhaengig davon!
190 GOTO 190
```

Listing. Ein Beispiel für den Befehl »Every«

mer 1, B wird durch 2 repräsentiert und C durch die 4. Im Prinzip gleichen sich also die Unterbrechungen, die durch EVERY und AFTER beziehungsweise ON SQ(...) ausgelöst werden. Daher ist es auch kein Wunder, daß der Schneider auch den Aufruf nach ON SQ wie einen Interrupt behandelt. Damit stellt sich dann auch die Frage, was passiert, wenn eine Unterbrechungsanforderung durch ON SQ mit einer anderen durch EVERY beziehungsweise AFTER zusammentrifft. Auch hier ist die Prioritätsfrage eindeutig geregelt. Die Unterbrechungsanforderung einer Tonwarteschlange liegt auf gleicher Ebene mit dem Timer 2, wird aber vor diesem ausgeführt. Die Tonschlangen selbst haben alle dieselbe Ranghöhe.

Wir wollen uns nun einmal die Möglichkeit einer Titelausgabe anschauen. Sie könnte zum Beispiel folgendermaßen aussehen:

```
10      EVERY 10,1 GOSUB 10000
20      PRINT"Hier folgt nun das Programm, welches den Titel aus-
        gibt."
30      PRINT"Dieser kann zum Beispiel in DATA-Zeilen abgelegt
        sein."
40      PRINT"die dann nacheinander auf dem Bildschirm ausgege-
        ben werden."
50      IF err<>1 THEN REMAIN (1):GOTO 20
60      PRINT"Hier Hauptprogramm!!":GOTO 60
10000   ON SQ(1) GOSUB 11000
10010   IF INKEY$<>"" THEN err=1
10020   RETURN
11000   DATA 200,100,400,100,X
11010   READ a$:IF a$="X" THEN RESTORE
11000   :READ a$
11030   RETURN
```

Wenn Sie dieses Beispiel eingetippt haben, sehen Sie auf dem Bildschirm im permanenten Scrollen die drei PRINT-Zeilen. Gleichzeitig wird die »Titelmelodie« — zwei Töne im Wechsel — gespielt. Wenn Sie nun auf eine Taste drücken, geht die Titelroutine in das eigentliche Hauptprogramm (ab Zeile 60) über. Der Titelvorspann ist dabei in zwei getrennten Teilen ausgeführt. Der eine Teil (Zeile 20 bis 40) beschäftigt sich mit der rollenden Titelausgabe. Über EVERY damit verknüpft, sind die beiden anderen Funktionen: Soundausgabe und Tastaturabfrage. Jede Fünftelsekunde erfolgt über EVERY der Einsprung in Zeile 10000. Wenn Platz in der Warteschlange ist, erfolgt der Aufruf von Zeile 11000, wo die Soundparameter aus DATA-Zeilen eingelesen werden. Das »X« dient dabei als Endmarkierung. Wird es vom Computer gelesen, so wird der DATA-Zeiger wieder auf den Anfang des DATA-Feldes zurückgesetzt (RESTORE) und ein neuer Durchlauf beginnt. Statt der zwei Töne kann in den DATAs natürlich auch ein »richtiges« Lied, sogar in mehreren Stimmen, gespeichert sein. Das Prinzip bleibt das gleiche. Allerdings müssen sie darauf achten, daß der Soundteil, genau wie die Tastaturabfrage, nur jede Fünftelsekunde angesprungen wird, da Timer 1 nur in diesem Abstand die erforderlichen Signale gibt. Bei sehr schnellen Melodien, die dann natürlich auch sehr kurze Notenlängen aufweisen, muß daher gegebenenfalls häufiger in das Unterprogramm verzweigt werden. Dazu setzen Sie einfach die Zeitangabe in EVERY herab.

Die Echtzeitbefehle haben also ein breites Anwendungsfeld bei der Koordination von Programmen und Programmteilen. Doch auch hier, wie für viele andere Bereiche des CPC, gilt: Man wird erst damit vertraut, wenn man damit arbeitet. Speziell die Kombination von verschiedenen Unterbrechungen erfordert einige Erfahrung, damit nicht ein Programmteil Variablen ändert, die anderswo noch dringend benötigt werden. Dennoch überwiegen die Vorteile dieser Befehle die Konsequenzen möglicher Fehlprogrammierungen bei weitem. Spielen Sie daher ruhig einmal ein wenig damit herum, und Sie werden sehen, welche fantastischen Variationen sich Ihnen eröffnen werden. Wenn Sie dabei alle Register ziehen (vier Timer und drei Tonwarteschlangen) so haben Sie wirklich das Ziel dieses Artikels erreicht: Ihr CPC tanzt auf mehreren Hochzeiten.

(Carsten Straush)

Dem Basic aufs Byte geschaut

Heimcomputer speichern die Befehle von Programmen in einer Hochsprache codiert in Form sogenannter Tokens. Welche Tricks und Besonderheiten angewandt werden, zeigen wir Ihnen hier.

Einer der Vorzüge, die der CPC gegenüber anderen Heimcomputern besitzt, ist sein gutes Basic. Das Interessante dabei ist, daß der beim Schneider benutzte Dialekt nicht nur mehr Befehle hat als die der Konkurrenz, was ja mit genügend großem ROM (der Schneider hat 16 KByte für den Basic-Interpreter) leicht möglich ist; der CPC ist trotz dieser Erweiterungen — oder vielleicht wegen dieser — vergleichsweise sehr schnell. So schnell, daß sich viele Probleme, bei denen man bei anderen Geräten auf die Maschinensprache oder ein compiliertes Basic zurückgreifen muß, problemlos mit dem Interpreter lösen lassen.

Doch warum ist das so? Die Antwort kann nur der Quelltext eines Basic-Programms und der Interpreter liefern. Starten wir also zu einer Reise durch den Speicher unseres Computers.

Zunächst brauchen wir ein paar Eckdaten, damit wir nicht völlig orientierungslos umherirren. Der CPC legt die Programme ab der Speicherstelle 368 im RAM ab. Das obere Ende des Programms gibt ein Zeiger an, der in den Speicherstellen AE83 hex und AE84 hex abgelegt ist. Die Adresse des obersten Programmbytes ist als 16-Bit-Wert gespeichert, in AE83 hex stehen die untersten (rechten, niederwertigen) acht Bits, AE84 hex enthält den oberen Teil der Adresse. Wir können die obere Grenze des Programms also jederzeit durch »PRINT PEEK (&AE83)+256*PEEK(&AE84)« abfragen.

Das Zielgebiet unserer Untersuchung ist nun bekannt. Als nächstes benötigen wir eine brauchbare Routine dafür. Dazu dient die Schleife »Untersuchungsroutine«. Sie gibt vom Beginn des Basic-Programms an Byte für Byte den Inhalt der einzelnen Speicherstellen aus, und zwar dezimal, hexadezimal und als Zeichen (Charakter). Aber nur sofern der Inhalt der Speicherstelle größer als 31 ist, da die unter 32 liegenden Zahlen Steuerfunktionen haben. Sie bewirken zum Beispiel die Definition einer neuen Farbe, die Bewegung des Cursors und so weiter, was zu seltsamen Effekten auf dem Schirm führen kann, wenn man diese Codes nicht richtig anwendet. Die Zeichenausgabe bei diesen Zahlenwerten wird deshalb von unserem Programm unterdrückt.

```
100 Untersuchungsroutine
110 ZONE 10:FOR i=368 TO PEEK
    (&AE83)+256*PEEK(&AE84)
120 PRINT i,PEEK(i),HEX$(PEEK
    (i),2),
130 IF PEEK(i) >31 THEN PRINT
    CHR$(PEEK(i)) ELSE PRINT
140 NEXT
```

Bevor wir nun tiefer in den Speicher eindringen, noch ein paar grundsätzliche Bemerkungen zur Struktur des Basic. Die Abspeicherung des Quelltextes erfolgt Zeile für Zeile. Dabei können wir zwei Bereiche unterscheiden: den Zeilenkopf mit Zeilennummer und Hinweis auf die nächste Zeile, sowie den eigentlichen Inhalt der Programmzeile.

Die Basic-Zeilen sind durch Zeiger verbunden

Der Kopf nimmt je Programmzeile vier Byte in Anspruch. Die ersten beiden enthalten die Länge der Programmzeile im Format Lo-Hi (zuerst das niederwertige, dann das höherwertige Byte). Dieser Abstand zur nächsten Zeile errechnet sich also, indem man den Wert des zweiten Bytes mit 256 multipliziert und dazu das erste Byte addiert. Analog funktioniert die Abspeicherung der Zeilennummer in den nächsten beiden Bytes.

Es stellt sich die Frage, warum soll der Schneider überhaupt eine Zeilenlänge von mehr als 255 Zeichen bearbeiten können, obwohl die Tastatureingabe doch auf maximal 255 Zeichen beschränkt ist. Die Ursache liegt darin, daß der CPC zum Beispiel für die Trennung von Funktionen oder Zahlenwerten im Text noch einmal besondere Steuerzeichen einfügt, die angeben, ob es sich bei den nachfolgenden Werten um Zahlenwerte oder Funktionen handelt. Dadurch erklärt sich, wieso eine Zeile im Quelltext, auch wenn sie auf dem Bildschirm nur 240 oder 250 Zeichen einnimmt, dennoch bei der Speicherung mehr als 255 Zeichen hat und der Zeiger muß diesen größeren Bereich enthalten können.

Die Möglichkeit beim CPC mehr als 255 Zeichen in eine Zeile zu packen, hat aber auch für uns große Vorteile, speziell wenn der Speicherplatz einmal besonders knapp wird. Wir können damit theoretisch ein ganzes Programm in einer Zeile unterbringen, solange es nur linear programmiert ist, das heißt keine Sprünge oder Verzweigungen aufweist. Der Interpreter interessiert sich nämlich nicht für die Gesamtlänge einer Zeile. Wir sparen dabei aber jedesmal 4 Bytes für Zeilenlänge und -nummer, was speziell bei vielen kurzen Zeilen ein Programm im Speicher erheblich aufblähen kann. Allerdings hat diese Sparmethode auch einen Nachteil. Ein solches Programm kann nachträglich nicht mehr editiert werden, da der Editor auf 255 Zeilen begrenzt ist. Aus diesem Grund muß die Verkürzung auch durch POKE-Befehle oder mit einer Maschinencode-Routine geschehen. Normalerweise dürfte das Verfahren daher auch von geringem Nutzen sein. Wer aber sein Programm als reine Laufversion ohne Compiler schneller machen will, sollte auf eine derartige Technik zurückgreifen, denn schließlich kostet das Suchen nach der Zeilennummer Zeit. Gerade einfache Schleifen kann man dadurch erheblich schneller bearbeiten lassen.

Der eigentliche Inhalt einer Basic-Zeile kann in drei Gruppen unterteilt werden. Die erste Gruppe bilden die Schlüsselwörter. Um Speicherplatz zu sparen, ist jedem Befehl in Basic ein sogenannter Token, ein Code, zugeordnet. Eine Zahl steht als Abkürzung für den Befehlsnamen, der in Einzelbuchstaben abgespeichert, erheblich mehr Platz beanspruchen würde. Diese Token liegen oberhalb des vom ASCII-Code benutzen Bereiches, also zwischen 128 und 255. Da das Schneider-Basic mehr als 128 verschiedene Befehle besitzt und somit mehr als 128 Token braucht, hat man beim Speichern der Basic-Befehle einen kleinen Trick benutzt: Das Zeichen mit der Nummer 255 dient als Erweiterungszeichen. Findet der CPC diesen Code, so wechselt er in die Funktionstabelle und interpretiert dann das nachfolgende Byte als ein Token der Erweiterungstabelle. Welche Werte dabei den verschiedenen Befehlen entsprechen zeigen die Tabellen 1 und 2. Bei den Token fallen ein paar Besonderheiten auf. Zunächst gibt es Befehle (wie zum Beispiel FOR...TO.., ON SQ) die, obwohl sie eine Einheit darstellen, doch aus mehreren Einzelbefehlen zusammengesetzt sind und auch als solche gespeichert sind. Dann finden Sie einen scheinbar etwas seltsamen Befehl: ON ERROR GOTO 0 (Code B4 hex). Er stellt das Gegenstück zu dem gerade Besprochenen dar: Eine Befehlskette, die als ein Token abgelegt ist. Mit diesem Befehl hat es eine besondere Bewandtnis. Sie kennen sicherlich das ON ERROR-Kommando. Es weist den Computer an, beim Auftreten eines Fehlers einen Sprung zu einer bestimmten Zeile auszuführen, wo dann die Fehlerbehandlung erfolgen kann. Jede Zeilennummer zwischen 1 und 65535 ist dabei erlaubt. Eine Zeile mit der Nummer 0 kann dagegen nicht eingegeben werden. Hier streikt der Schneider. Mit einem Trick können wir diese Zeile jedoch trotzdem benutzen. Wir brauchen bloß mit einem POKE die Zeilennummer zu ändern. Dazu definieren

00	ABS	1C	UPPER$
01	ASC	1D	VAL
02	ATN	.	
03	CHR$	.	
04	CINT	40	EOF
05	COS	41	ERR
06	CREAL	42	HIMEM
07	EXP	43	INKEY$
08	FIX	44	PI
09	FRE	45	RND
0A	INKEY	46	TIME
0B	INP	47	XPOS
0C	INT	48	YPOS
0D	JOY	.	
0E	LEN	.	
0F	LOG	71	BIN$
10	LOG10	72	DEC$
11	LOWER$	73	HEX
12	PEEK	74	INSTR
13	REMAIN	75	LEFT$
14	SGN	76	MAX
15	SIN	77	MIN
16	SPACE$	78	POS
17	SQ	79	RIGHT$
18	SQR	7A	ROUND
19	STR$	7B	STRING$
1A	TAN	7C	TEST
1B	UNT	7D	TESTR
		7F	VPOS

Tabelle 1. Token der Funktion

wir zunächst den Inhalt unserer »Geheimzeile« — zum Beispiel mit der Nummer 10. Wenn sie richtig eingegeben ist, können wir diese Zeile jederzeit durch Änderung des ersten Bytes für die Zeilennummer — und zwar auf 0 — unsichtbar machen. Denn dies ist das Besondere: Die Zeile 0 kann nicht nur nicht direkt eingegeben werden, sondern sie läßt sich auch nicht editieren und nicht auflisten. Trotzdem funktioniert sie problemlos. Probieren Sie es einmal aus.

ASCII — der Code für Texte und Variablennamen

Die zweite Gruppe einer Basic-Zeile bilden Textangaben, wie sie bei der Definition von Strings oder der Ausgabe von Texten auftreten. Diese, wie auch die Variablennamen und einige Sonderzeichen, wie Kommas und Punkte sind im ASCII-Code angegeben. Auch hier gibt es eine Besonderheit. Der CPC kann nämlich verschieden lange Variablennamen verarbeiten und auch abspeichern. Woran erkennt der Computer aber nun, daß eine Namensdefinition zu Ende ist? Die Antwort ist relativ einfach: am letzten Buchstaben. Für diesen wird nämlich nicht der ASCII-Code verwendet, sondern zum letzten ASCII-Wert werden noch 128 (oder 80 hex) addiert. Wenn also der Variablenname »ABC« im Speicher abgelegt wird, so werden drei aufeinanderfolgende Speicherstellen wie folgt belegt: 65, 66, 195. Da die 195 (und generell alle Zahlen größer als 128) vom Schneider benutzt werden, um das Ende eines Namens zu erkennen, kann man diese Werte natürlich nicht in der Mitte eines Namens verwenden. Die Grafiksymbole (sie alle haben Werte größer als 128) scheiden daher als erlaubte Zeichen aus.

Spezialzeichen sorgen für einen schnelleren Ablauf

Die dritte Gruppe bilden die Steuer- und Sonderzeichen. Diese sind sicher die größte Neuerung, die das Basic des Schneider gegenüber anderen Computern aufweist und hauptverantwortlich dafür, daß alles beim Schneider relativ schneller bearbeitet wird. Die Steuercodes haben Werte zwischen 0 und 31. Sie dienen der Strukturierung innerhalb einer Zeile. So definiert eine Null das Zeilenende und eine Eins gibt das Ende eines Befehls an. Steht in einer Zeile eine der Zahlen 02, 03, 04 oder 0D (alle hexadezimal), so werden die nachfolgenden Werte als Definition einer Variablen angesehen. Sie speichern die Typenkennzeichen »/«, »!« und »$«, beziehungsweise eine Variable ohne Kennung. Eine weitere Besonderheit des CPC-Basic stellen die nachfolgenden Werte dar. Die Codes zwischen 0E und 1C hex dienen ebenfalls der Speicherung von Zahlenwerten und zwar in verkürzter Form. Da es relativ häufig vorkommt, daß in Basic-Programmen nur mit einstelligen Dezimalzahlen gearbeitet wird — zum Beispiel bei Zählschleifen — ist man beim Entwurf des Basic neue Wege gegangen. Statt immer für jede Zahl eine Realvariable anzulegen, wurde beim CPC eine platzsparende Variante gewählt. Eine Zahl zwischen Null und Neun wird als Konstante (Codes 0E hex bis 17 hex) abgelegt und benötigt somit nur ein Byte Speicherplatz, was dann auch eine schnellere Bearbeitung des Programms garantiert.

Auch Zahlenwerte kleiner als 256 werden einer Spezialbehandlung unterzogen. Der Code 19 hex (beziehungsweise 25 dezimal) erlaubt diese Werte verkürzt zu speichern. Das nachfolgende Byte gibt dabei den Wert an. Für größere ganze Zahlen gibt es die Möglichkeit der Integerdarstellung in 2 Bytes, wobei je nachdem, ob die Eingabe dezimal hexadezimal oder binär erfolgt ist, verschiedene Codes für die Abspeicherung benutzt werden. Mit 1F hex wird eine Fließkommagröße gekennzeichnet. Diese stärkere Unterteilung der Variablendefinition, die Beschränkung auf den minimal nötigen Speicherplatz, erlaubt es, daß die Rechenoperationen bedeutend schneller ablaufen. In puncto Geschwindigkeit hat der CPC aber noch eine Spezialität parat: die direkte Adressierung von Variablen und Sprüngen. Neben dem Typenkürzel gehören zu jeder Variablen oder Verzweigung mit GOTO und GOSUB zwei Byte, die beim Start des Programms auf Null gesetzt sind. Beim erstmaligen Aufruf der Befehle legt der CPC die Variablen an oder er durchsucht den Speicher nach der angesprochenen Adresse. Danach legt er die absolute Position der Variablen oder Basic-Zeile im Programm in eben jenen zwei freien Bytes ab. Beim zweiten Durchlauf entfällt somit die Suche nach den benötigten Variablen oder Ansprungpunkten. Der Interpreter arbeitet bei der Ablage des Quelltextes fast wie ein Compiler, wodurch die Programmausführung noch einmal deutlich schneller wird. Die Steuercodes finden Sie als Ganzes in Tabelle 3.

80	AFTER	A0	GOTO	C0	'	E4	FN
81	AUTO	A1	IF	C1	RAD	E5	SPC
82	BORDER	A2	INK	C2	RANDOMIZE	E6	STEP
83	CALL	A3	INPUT	C3	READ	E7	SWAP
84	CAT	A4	KEY	C4	RELEASE		
85	CHAIN	A5	LET	C5	REM		
86	CLEAR	A6	LINE	C6	RENUM	EA	TAB
87	CLG	A7	LIST	C7	RESTORE	EB	THEN
88	CLOSEIN	A8	LOAD	C8	RESUME	EC	TO
89	CLOSEOUT	A9	LOCATE	C9	RETURN	ED	USING
8A	CLS	AA	MEMORY	CA	RUN	EE	>
8B	CONT	AB	MERGE	CB	SAVE	EF	=
8C	DATA	AC	MID$	CC	SOUND	F0	>=
8D	DEF	AD	MODE	CD	SPEED	F1	<
8E	DEFINT	AE	MOVE	CE	STOP	F2	<>
8F	DEFREAL	AF	MOVER	CF	SYMBOL	F3	<=
90	DEFSTR	B0	NEXT	D0	TAG	F4	+
91	DEG	B1	NEW	D1	TAGOFF	F5	—
92	DELETE	B2	ON	D2	TRON	F6	*
93	DIM	B3	ON BREAK	D3	TROFF	F7	/
94	DRAW	B4	ON ERROR GOTO 0	D4	WAIT	F8	^
95	DRAWR	B5	ON SQ	D5	WEND	F9	ö
96	EDIT	B6	OPENIN	D6	WHILE	FA	AND
97	ELSE	B7	OPENOUT	D7	WIDTH	FB	MOD
98	END	B8	ORIGIN	D8	WINDOW	FC	OR
99	ENT	B9	OUT	D9	WRITE	FD	XOR
9A	ENV	BA	PAPER	DA	ZONE	FE	NOT
9B	ERASE	BB	PEN	DB	DI	FF	Funktion siehe Erweiterungstabelle
9C	ERROR	BC	PLOT	DC	EI		
9D	EVERY	BD	PLOTR				
9E	FOR	BE	POKE				
9F	GOSUB	BF	PRINT	E3	ERL		

Tabelle 2. Basic-Token der einfachen Befehle

Wie sieht dies nun aber konkret aus? Schauen wir einmal im Basic-Quelltext nach. Weisen wir der Integervariable a/ den Wert 7 zu, so finden wir dies im Basic wie folgt abgelegt.:
2 0 0 225 239 21 0

Sie erhalten diese Zahlenfolge, wenn Sie nach dem Einschalten die Zeile 10 a/=7 eingeben. Dabei sollten Sie, wie auch im folgenden, immer auf die exakte Eingabe der Leerzeichen achten, sonst werden Sie zwischen den Werten, die wir jetzt mit »PRINT PEEK(372)« bis »PRINT PEEK(378)« lesen können, den Code 32 (Leerzeichen, Space) vorfinden. Der CPC speichert nämlich im Normalfall jedes eingegebene Leerzeichen mit ab.

Zurück zur Interpretation dieser Zahlenfolge. Die Zwei definiert den Variablentyp, eine Intergervariable. Die nachfolgenden zwei Bytes bestimmen die Speicherstelle, an der sich die Integervariable im Speicher befindet. Es handelt sich hierbei nicht um eine absolute Adresse, sondern um die relative Position ab Programmende. Alle einfachen Veränderlichen sind nach dem Basic-Quelltext in einem abgeschlossenen Datenbereich, dem Variablenspeicher abgelegt. Ein Zeiger (AE85 hex, AE86 hex) kennzeichnet den Beginn dieses Feldes. Bei der erstmaligen Benutzung einer Variablen im Programm wird diese in den Variablenbereich geschrieben und in den beiden Bytes im Quelltext wird ihre Position relativ zu diesem Zeiger abgelegt. Damit können Variablennamen ohne langes Suchen sehr schnell wieder gefunden werden, was besonders bei Schleifenvariablen, die gegebenenfalls mehrere hundertmal benutzt werden, erheblich Zeit spart. Als nächsten Wert bei der Interpretation unserer Zahlenfolge stoßen wir auf die Codierung des Namens. Der Name einer Variablen ist, wie schon gesagt, in ASCII-Codierung abgelegt; zur letzten Stelle des Namens werden dabei die 128 als Enderkennung addiert. Es wird also das achte Bit des letzten Buchstabens gesetzt. Da wir nur einen Buchstaben benutzt haben, erfolgt die Addition der 80 hex hier bereits in der ersten Stelle. »PRINT CHR$(225-128)« gibt uns hier die Bestätigung, es wird ein kleines »a« ausgegeben. Als nächstes folgt für die Wertzuweisung das Gleichheitszeichen (Token 239) und danach in Kurzform der Wert 7. Die 0 schließt die Zeile ab.

Nach soviel Theorie wollen wir nun unser Wissen einmal mit Hilfe der oben angegebenen Routine anwenden. Dazu setzen wir die zu untersuchenden Zeilen mit niedrigerer Zeilennummer ein. Zuerst wollen wir einen einfachen PRINT-Befehl betrachten.
10 PRINT"abcd"

00	Zeilenende
01	Statementende
02	Kennzeichen %
03	Kennzeichen $
04	Kennzeichen !
0D	Variable ohne Kennung
0E	Ganzzahl 0
0F	Ganzzahl 1
10	Ganzzahl 2
11	Ganzzahl 3
12	Ganzzahl 4
13	Ganzzahl 5
14	Ganzzahl 6
15	Ganzzahl 7
16	Ganzzahl 8
17	Ganzzahl 9
19	Ein-Byte-Wert
1A	Zwei-Byte-Wert dezimal
1B	Zwei-Byte-Wert binär
1C	Zwei-Byte-Wert hexadezimal
1D	Sprung auf Zeilenadresse
1E	Sprung nach Zeilennummer
1F	Kennzeichen für einen Fließkommawert

Tabelle 3. Bedeutung der Strukturparameter

Wenn Sie die Routine »anwerfen«, so erhalten Sie am Anfang folgende Ausgaben:

368	12	0B	Zeilenlänge und Verweis auf die nächste Zeile
369	00	00	
370	10	0A	Zeilennummer Zeile 10
371	00	00	
372	191	BF	PRINT
373	34	22	"
374	97	61	a
375	98	62	b
376	99	63	c
377	100	64	d
378	34	22	"
379	00	00	Zeilenende

Danach erscheint die nächste Zeile, also die erste Zeile der Untersuchungsroutine.

380	27	1B	Zeilenlänge 27: auf diese Adresse zeigt die vorhergehende Zeile
381	0	00	
382	100	64	Zeilennummer 100
383	0	00	

Nach dem REM-Befehl folgt nun der Titel. Auch dieser ist im ASCII-Code Zeichen für Zeichen abgelegt. Dies kennen wir jedoch schon. Interessanter wird es in der Untersuchungsroutine ab Adresse 409. Hier finden sich die eigentlichen Befehle des Programms. Zunächst das Token für das ZONE-Kommando, welches die Breite der Spalten definiert. Dann einige besondere Zahldefinitionen: 19 hex gibt an, daß es sich um einen Ein-Byte-Wert handelt, hier 10. Der Hexcode 01 gibt an, daß der Befehl zu Ende ist. Es folgt die FOR...TO...-Schleife:

418	158	9E	FOR
419	32	20	<Space>
420	13	0D	Definiton einer Variablen ohne Kennzeichen
421	5	05	die 5 Bytes nach dem
422	0	00	Programmende abgespeichert ist
423	233	E9	Name der Variablen: i = CHR$(105)+128
424	239	EF	Gleichheitszeichen
425	26	1A	dezimaler 2-Byte-Wert
426	112	70	368
427	1	01	
428	32	20	<Space>
429	236	EC	TO
430	32	20	<Space>
431	255	FF	Funktionskennung für Erweiterungstabelle
432	18	12	PEEK
433	40	28	(
434	28	1C	2-Byte-Wert (hexadezimal)
435	131	83	AE03
436	174	AE	
437	41	29	)
438	244	F4	Operator »+«
439	26	1A	2-Byte-Wert dezimal
440	0	00	256=1*256+0
441	1	01	
442	246	F6	Operator » * «
443	255	FF	Funktionserkennung siehe Erweiterungstabelle
444	18	12	PEEK
445	40	28	(
446	28	1C	2-Byte-Wert hex AE84
447	132	84	
448	174	AE	
449	41	29	)
450	0	00	Zeilenende

Nun folgt die Zeile 120. Die darin enthaltenen Befehle stellen für uns, nachdem wir uns mit dieser vielleicht etwas ungewohnten Art Basic-Text zu lesen vertraut gemacht haben, kein Problem mehr dar. Auch hier werden Tokens und ASCII-Werte verknüpft, um das gewünschte Ergebnis zu erhalten. Sie sollten aber diese Zeilen einmal selbständig interpretieren, um ein Gefühl dafür zu bekommen, wie der Interpreter, also jener Teil im Betriebssystem, der für die Programmausführung zuständig ist, vorgeht. Damit wird nicht nur das Verständnis für die Grenzen der Maschine besser.

(Carsten Straush)

Ein-Blick in den Variablenspeicher

Jeder Computer benötigt zur Ablage von Informationen Variablen. Kennt man ihre Struktur im Speicher und weiß man ihre Adressen, so kann man viele Probleme effektiver lösen.

Eines der interessantesten Untersuchungsobjekte bei einem Computer ist die Ablage der Variablen. Kann man doch mittels direktem Zugriff seine Programme erheblich schneller machen. So verkürzt beispielsweise eine Sortierroutine in Maschinencode die Laufzeit eines Basic-Programms erheblich. Gleiches gilt für die Bestimmung von Maxima und Minima eines Zahlenfeldes. Um in Maschinencode auf die Variablen eines Basic-Programms zugreifen zu können, muß man jedoch die verschiedenen Variablentypen mit ihrem Aufbau kennen.

Zunächst einmal können wir drei Typen von Variablen unterscheiden: Real- oder Fließkomma-, Integervariable und Strings. Jeder dieser drei Typen ist auf unterschiedliche Art und Weise gespeichert. Und jeder Typ kann noch in zwei Formen vorkommen, als einzelne Variable oder als Feld — ein Array (zum Beispiel A (3,3)).

Wenden wir uns zunächst den einfachen Variablen zu. Zu Anfang müssen wir uns etwas näher mit dem Aufbau des Speichers beschäftigen. Der Speicherbereich für Basic reicht von Adresse 367 bis zum oberen Ende, der durch HIMEM definiert ist (normalerweise beim 464 ohne Diskettencontroller 43533). Dieser Bereich steht dem Benutzter für sein Basic-Programm, Variablen und Kassettenoperationen zur Verfügung. Die Unterteilung dieses Speicherbereichs in einzelne Teile wird dabei durch verschiedene Zeiger vorgenommen. So gibt es einen, der auf die Startadresse des Basic-Programmes zeigt und einen zweiten, der das Ende deutlich macht. Weitere Zeiger geben die Speicherplätze der Variablen, Arrays und so weiter an. Daneben findet man noch zwei Zeiger für den Beginn und das Ende des Bereichs — der Strings. Jeder dieser Zeiger besteht aus 2 Byte und gibt eine absolute Position im Speicher an. Das Format ist dabei wie üblich Lo-Hi (niederwertiges — höherwertiges Byte). Geben wir also »PRINT PEEK (&AE81)+256* PEEK(&AE82)« ein, so wird der Wert des Zeigers PROGSTART auf dem Bildschirm ausgegeben. Normalerweise muß er 367 betragen. Die Anordnung der einzelnen Zeiger und ihrer Funktion bei der Aufteilung des Speichers gibt unser Bild (Seite 75) wieder. Hierbei geben die Pfeile die Richtung an, in die die einzelnen Bereiche sich ausdehnen. Daneben finden Sie die einzelnen Pointer. Die Positionen sind dabei wieder im Format Lo-Hi gespeichert.

Während das Basic-Programm, die Variablen und die Arrays sich von unten nach oben ausdehnen, wachsen die Strings von der Speicherobergrenze (HIMEM) in umgekehrter Richtung. Dies ist jedoch nur die halbe Wahrheit. Denn die Ablage eines Strings erfolgt in zwei Teilen, dem eigentlichen Inhalt der Zeichenkette und einem Zeiger auf diesen. Nur die eigentliche Zeichenfolge wird in dem oberen Bereich abgelegt. Der Zeiger dagegen befindet sich im unteren Bereich, wo auch die Variablen liegen. Die Variablen sind jedoch nicht die einzigen »Bewohner« der Speicherregion oberhalb des Basic-Quelltextes. Hier liegen auch die Zwischenpuffer für Lade- und Speicheroperationen. Deshalb ist zwischen Strings und den »normalen« Variablen ein Bereich (der Memory Pool) freigehalten. Hierbei können gegebenenfalls Daten ausgelagert werden, wenn einer der Bereiche ausgedehnt oder an eine andere Adresse verschoben werden soll. Aber wie geschieht dies nun? Soll zum Beispiel eine neue Variable definiert werden (wir nehmen eine einfache Variable an), so muß der Arrayblock zwischen ARRAY START und ARRAY END um die benötigten Bytes nach oben geschoben werden und in den jetzt freien Raum wird die neue Varibale abgelegt. Beim Löschen läuft dieser Vorgang in entgegengesetzter Richtung ab. Die Änderungen in den anderen Bereichen laufen analog ab.

Wie kann man nun aber eine bestimmte Variable im Speicher finden? Um die aktuelle Position der gesuchten Variablen im Speicher feststellen zu können, stellt uns das Schneider-Basic den Befehl »@«, auch Klammeraffe genannt, zur Verfügung. Jede Variable ist im Speicher mit Name und den dazugehörigen Werten abgelegt. »@« gibt uns nun an, ab welcher Stelle die eigentliche Variable zu finden ist. Das heißt, er zeigt auf das erste Byte nach dem Ende des Variablennamens. Mit »PRINT @ a« erfahren wir also die Adresse, an der die Variable a gespeichert ist.

Schauen wir uns nun die Ablage der Variablen ein wenig näher an. Dazu benutzen wir eine kleine Untersuchungsroutine. Sie gibt uns den Inhalt der Speicherstellen direkt nach dem Programmende aus.

```
100 FOR I=PEEK(&AE85)+256*
PEEK(&AE86) TO PEEK(&AE87)+
256*PEEK(&AE88)
110 PRINT I,HEX$(PEEK(I),2),
120 IF PEEK (I)>31THEN PRINT
CHR$(PEEK(I))ELSE PRINT
130 NEXT
```

Wenn wir nun im Programm (vor Zeile 100) noch verschiedene Variable definieren, so können wir diese im Speicher leicht wiedererkennen. Sie liegen hintereinander durch Nullen voneinander getrennt. Ihre Reihenfolge ergibt sich dabei aus dem Programm: Was zuerst im Programmablauf definiert wurde, liegt ganz unten. Definieren wir einmal: 10 a/=33:ab=17:abc$="alles klar?" Nun lassen wir unser Programm laufen.

```
10 INPUT"Mantisse 1";m1
20 INPUT"Mantisse 2";m2
30 INPUT"Mantisse 3";m3
40 INPUT"Mantisse 4";m4
50 INPUT"Exponent  ";ex
100 r=(1+((m4 AND &X1111111)/128+m3/(128
*256)+m2/(256^2*128)+m1/(256^3*128)))*2^
(ex-129)*-SGN(m4-127.5)
110 PRINT r
120 GOTO 10
```

Schauen Sie sich an, wie ein Fließkomma gespeichert wird

Nach einer Anfangsnull erhalten wir — als ersten interessanten Wert — eine 193. Dies ist der Name der zuerst definierten Variablen. Er setzt sich aus dem Code für A(=65) und der hinzugefügten 128 für das Ende des Namens zusammen (zu dem Wert des letzten Buchstabens eines Variablennamens werden als Kennzeichen 128 hinzuaddiert). Die nächste Speicherstelle enthält den Variablentyp: eine 1 für Integer. Die Schneider-Computer unterscheiden drei Variablentypen. Ganze Zahlen erhalten eine »1«. Die »4« zeigt an, daß ein Wert eine Fließkommazahl darstellt und der Zeiger auf einen String wird durch eine »2« gekennzeichnet. Dieses Kürzel für den Variablentyp entspricht dabei der Anzahl der für die Abspeicherung der Variable — beziehungsweise des zugehörigen Zeigers — benötigten Bytes — vermindert um eins. Name und Typkürzel werden bei dieser Berechnung allerdings nicht berücksichtigt. Bei einer Realzahl folgen also fünf weitere Bytes. Ein String gibt sich mit drei zusätzlichen Bytes zufrieden und eine Integerzahl gar nur mit den zwei folgenden. In diesen beiden Adressen erfolgt die Angabe des Wertes der Zahl. Die Abspeicherung erfolgt dabei wieder im Format Lo-Hi. Wir können somit eine Integervariable als 16-Bit-Wert betrachten. Dabei speichert das höchste (linke, am meisten signifikante) Bit das Vorzeichen. Die anderen 15 Bits sind für den Absolutwert zuständig. Man kann so Zahlen zwischen -2^{15} und $+2^{15}$ abspeichern. Wert ($2^{15}=32768$) ergibt sich wie folgt: Mit einem Bit lassen sich zwei Zustände beschreiben (0 und 1). Nimmt man ein weiteres hinzu, so ergeben sich schon vier Kombinationen: 00, 01, 10 und 11. Mit einem Bit kann man also 2^1 Zustände beschreiben. Mit zwei Bits $2^2=4$. Bei drei Bits sind es dann $2^3=8$ verschiedene Werte, die man codieren kann. In diese Richtung weitergedacht, kommt man schließlich auf die oben angegebene Zahl. Eine ganze Zahl zwischen -32768 und 32767 kann so in zwei Byte oder 16 Bit abgelegt werden.

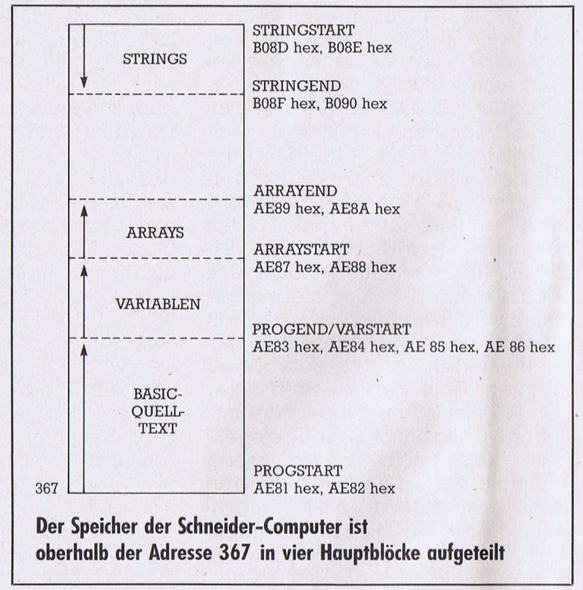
Zerlegt man diese 16 Bits in die höherwertigen und die niederwertigen acht Bits, so kommt man zu den zwei Werten, die nach der Typkennung für unsere Variable a angegeben sind, 33 und 0. Das oberste Byte wird also nicht benötigt. Dies ist auch selbstverständlich, wenn wir uns den Wert 33 einmal als einzelne Bits anschauen. Mit »PRINT BIN$(33,16)« zeigt es uns der Schneider. Die oberen, linken acht Bit sind alle 0, weshalb auch das höherwertige Byte diesen Wert hat. Dabei ist natürlich auch zu berücksichtigen, daß wir es mit einer positiven Zahl zu tun haben. Ein negativer Wert würde immer eine Angabe ungleich Null im höherwertigen Byte nach sich ziehen — zumindest eine 128, das dezimale Gegenstück zu binär 1000 0000, der Angabe für das negative Vorzeichen. Lassen Sie sich doch einmal PRINT BIN$(-33,16) auf dem Bildschirm ausgeben.

Doch zurück zu unserer eigentlichen Aufgabe, der Interpretation des Variablenspeichers. Bei einer Integerzahl deutet der Variablenpointer »@« auf die erste dieser beiden Speicherstellen hin. Wir können uns also die jeweilige Position von a/ durch »PRINT@ a/« jederzeit ausgeben lassen. Als Ergebnis müßten Sie in diesem Beispiel 555 erhalten. Vorausgesetzt, Sie haben das Basic-Programm nicht durch unnötige Leerzeichen »aufgebläht«. In diesem Fall ergäbe sich ein höherer Wert. Wenn man diesen Wert an ein Maschinenprogramm weitergibt, kann man dort den Wert der Variablen abfragen.

Kommen wir nun zur zweiten Variablen, einer Fließkommazahl. Sie liegt ab Speicherstelle 559 im Speicher.

An dieser und der nachfolgenden Adresse finden wir zunächst den aus zwei Buchstaben bestehenden Namen. 561 liefert das Typenkürzel (eine 4 für Real). Doch dann sieht es etwas ungewohnt aus. Um sie zu verstehen, muß man sich die Art ansehen wie eine Fließkommazahl gespeichert wird. Sie ist in fünf Bytes abgelegt. Die ersten vier bilden dabei die sogenannte Mantisse, wobei das erste Byte das niederwertigste ist. In unserem Fall lauten sie 0,0,0 und 8. Diese vier Bytes müssen wir nun zu einer 31-Bit-Binärzahl zusammensetzen, um unsere 17 zu erhalten. Wieso 31 Bits, werden Sie nun vielleicht fragen. Die Antwort ist einfach. Dem 32-ten Bit, dem obersten Bit des vierten Mantissen-Bytes kommt nämlich eine Sonderfunktion zu. Es gibt, genauso wie bei den ganzen Zahlen, das Vorzeichen an. Insgesamt steht uns somit eine Binärzahl mit 31 Stellen, einem Bit für das Vorzeichen und weiteren acht Bit (dem Binärexponenten) zur Verfügung. Der Binärexponent (in unserem Fall 133) wirkt im Prinzip wie ein Exponent im Dezimalsystem. Nur daß hier mit einem Vielfachen von 2 multipliziert wird. Hat der Exponent den Wert +5, so wird die Mantisse mit $2^5=32$ multipliziert. Nun ist es aber auch möglich, einen negativen Exponenten zu benutzen. Der in der Mantisse enthaltene Wert wird dann verkleinert. Deshalb wurde nicht die 0 zur Basis des Exponenten, sondern die 128. Ein Wert des Exponenten von 133, wie in unserem Beispiel, veranlaßt also die Multiplikation mit 2^5.

Um eine solche Zahl in einen Dezimalwert zu wandeln, bedarf es aber noch einiger Arbeit. Zuerst müssen die Mantissen-Bytes umgerechnet werden. Der Wert der Mantisse schwankt immer zwischen 0,5 und 1 (binär zwischen $2^{-1}=0{,}5$ und $2^0=1$) — jeweils ohne Berücksichtigung des Vorzeichens. Jedes Mantissen-Byte hat dabei die Wertigkeit von $1/256$ des nächsthöheren. Das höchstwertige Byte hat dann die Wertigkeit $1/128$. Den besten Eindruck, wie eine solche Umrechnung funktioniert, erhält man natürlich nicht durch das Lesen irgendeines Artikels, sondern durch Ausprobieren. Um daher einen Eindruck zu gewinnen, wie eine Wertänderung sich aus-



Der Speicher der Schneider-Computer ist oberhalb der Adresse 367 in vier Hauptblöcke aufgeteilt

wirkt, sollten Sie einmal das Listing auf Seite 74 eintippen. Es erlaubt Ihnen, die Mantissen-Bytes und den Exponenten Stück für Stück einzugeben und sich so mit der Umformung vertraut zu machen. Wenn Sie sich daran erinnern, daß der CPC auch eine binäre Eingabe kennt, können Sie die Änderungen Bit für Bit ausprobieren. Die Binäreingabe erfolgt übrigens, indem Sie das Kürzel »&X« benutzen und danach die einzelnen Nullen und Einsen eintippen. Einen ersten Test können Sie durchführen, indem Sie die gerade gelesenen Werte (0, 0, 0, 8, 133) benutzen. Sie erhalten dann die gesuchte 17.

Bei der Ablage eines Strings müssen zwei Einträge im Speicher gemacht werden. Zum einen wird der Inhalt des Strings im Stringbereich, also zwischen STRING START und STRING END abgelegt. Auf diesen String wird dann ein Zeiger gesetzt, der unter dem Namen des Strings gespeichert ist und ganz normal zwischen den Real- und den Integerwerten im Variablenbereich liegt. Ein String besteht somit aus seinem Namen und darauffolgend einer 2 als Kennzeichen. Die nächste Speicherstelle enthält die Länge der Zeichenkette und die darauffolgenden geben die aktuelle Position des Strings im Speicher an.

Dies sind auch die Informationen, die uns unser Programm für den String »abc« auswirft. Zunächst drei Bytes, die den Namen enthalten: die Werte für a, b und dann der um 128 erweiterte für c. Es folgt eine 2 zur Kennzeichnung des Typs. Danach folgen die Länge unseres Strings und seine Position (wieder im Format Lo-Hi (1,141). Wenn man diesen Wert ausrechnet, so erhält man als Ergebnis 397, also eine Adresse im Basic-Programm. Damit stellt sich natürlich die Frage, was denn die abzuspeichernden Strings dort verloren haben. Was auf den ersten Blick — und auch später noch manchmal — verblüfft, ist das Ergebnis eines schlauen Tricks, der auch bei anderen Computern angewandt wird. Normalerweise werden die zu einem String gehörenden Zeichen an der Speicherobergrenze abgelegt. Nun wäre es aber unter dem Gesichtspunkt »Speicherplatz sparen« völlig unnötig, eine Zeichenkette dort nochmals abzulegen, wenn sie bereits an einer anderen Stelle existiert. Wird eine Stringvariable daher nur mit einer Zeichenkette, die auch im Programm steht, geladen, so wird der Zeiger auf den String auf diese Stelle im Programm gerichtet. Erst wenn dieser String geändert wird, erfolgt die Ablage an der Speicherobergrenze. Wenn wir also beispielsweise

20 abc$=abc$+"a"

einfügen, so finden wir den daraus resultierenden String jetzt an der Speicherobergrenze wieder. Die Adressenangabe zeigt jetzt auf 43893 (171*256+117), beziehungsweise bei angeschlossenem Laufwerk auf 42599. Hier finden sich dann die einzelnen Zeichen im ASCII-Code abgelegt. Mit »FOR i=43893 TO 43903:PRINT i, CHR$(i): NEXT« bekommen wir den ganzen String ausgegeben.

Den Feldern auf der Spur

Bei Arrays haben wir es mit Variablenfeldern zu tun. Unter ein und demselben Namen sind mehrere Variable gespeichert, die durch Indizierung unterschieden werden. Jede einzelne wird also bezeichnet, indem man den Feldnamen angibt und dann in Klammern die Position im Feld angibt. Mehrdimensionale Felder müssen beim CPC prinzipiell mit dem DIM-Kommando definiert werden, eindimensionale Felder nur in dem Moment, wo der Index größer als 11 erlaubt sein soll. Dies hat seinen Grund darin, daß bei der erstmaligen Verwendung einer indizierten Variablen, zum Beispiel »M(8)« ein Feld angelegt wird, welches dann für den Rest der Arbeitszeit im Speicher verbleibt. Wollte man beliebige eindimensional indizierte Felder ohne den DIM-Befehl aufrufen, so müßte bei dem erstmaligen Aufruf ein eindimensionales Feld von unendlicher Größe definiert werden. Bei der Benutzung der nächsten eindimensionalen Variablen ebenfalls wieder und so weiter. Daß dies natürlich nicht möglich ist, ist sofort einsichtig und man hat sich deshalb darauf beschränkt, beim Aufruf einer eindimensionalen Feldvariablen immer nur ein Feld der Größe 10 zu definieren. Es können also nur die Indizes von 0 bis 10 ohne vorherige Dimensionierung benutzt werden. Dies ist aus zwei Gründen jedoch nicht zu empfehlen. Zum ersten wird man bei einer Felddefinition am Programmanfang darauf hingewiesen, wieviele Werte das Feld aufnehmen kann. Das Verfahren schafft also speziell bei mehreren Variablen Übersicht.

Zweitens: Auch wenn man weniger als elf indizierte Variablen benötigt, wird jedesmal ein dimensioniertes Feld auf elf Variable eröffnet. Das kostet natürlich Speicherplatz. Übersicht und Speicherplatz lassen es sinnvoll erscheinen, auch kleine eindimensionale Felder grundsätzlich zu dimensionieren.

Betrachten wir nun einmal wie diese Felder im Speicher liegen. Als erstes wollen wir uns einmal den Aufbau eines Stringarrays anschauen. Als Hilfe dazu dient uns wieder ein kleines Untersuchtungsprogramm:

```
 5 DIM m$ (12,12,12)
 7 m$ (0,0,0)="13 Bytes voll":m$
   (12,12,12)="7 Bytes"
 8 m$ (1,0,0)="1"
10 FOR i=PEEK (&AE87)+256*
   PEEK (&AE88) TO PEEK
   (&AE89) +256*PEEK (&AE8A)
20 PRINT i,PEEK(i),HEX$(PEEK(i))
   :NEXT
```

Dieses Programm stellte zunächst ein Stringarray der Größe 13x13x13 zur Verfügung. Die Indizes dürfen von 0 bis 12 laufen. Danach werden drei Variablen in diesem Feld definiert.

Variablenfelder stehen beim Schneider in einem Bereich des Benutzerspeichers, der zwischen den Zeigern ARRAY START und ARRAY END liegt (also oberhalb der einfachen Variablen). Über diesen Bereich läuft die Zählschleife in Zeile 10. Zeile 20 gibt dann die Adresse der Speicherstelle und den Inhalt — sowohl dezimal wie auch hexadezimal — aus. Lassen wir das Programm laufen, so erhalten wir die folgende Ausgabe.

Position	dez	Kommentar
565	0	
566	0	
567	205	Name "M" + 128
568	2	Variablentyp String
569	198	Länge des Descriptorfeldes
570	25	
571	3	Dimension
572	13	Anzahl der Elemente in der 1. Dimension Low-Byte
573	0	High-Byte
574	13	Anzahl der Elemente in der 2. Dimension Low-Byte
575	0	High-Byte
576	13	Anzahl der Elemente in der 3. Dimension Low-Byte
577	0	High-Byte
578	13	Länge erster String m$(0,0,0)
579	150	Position im Speicher
580	1	
581	1	Länge zweiter String 1 m$(1,0,0)
582	207	Position im Speicher
583	1	

Auf der Suche nach einem String

Sollten Sie für die Speicherstelle höhere Adressen bekommen, so liegt dies daran, daß sie in dem Programm an irgendeiner Stelle Leerzeichen eingegeben haben, die, wie wir schon gesehen haben, das Programm verlängern und damit auch zu einer Verschiebung des Arraybereiches nach oben führen. Bei normaler Eingabe des Programms beginnt die Ausgabe ab Speicherstelle 565. Die Ablage des Feldes läuft ähnlich ab, wie die Abspeicherung einer einfachen Variable. Zuerst steht der Name, wobei zum letzten Zeichen 128 als Endkennung addiert werden.

Es folgt der Feldtyp, in unserem Beispiel eine 2 für Strings. Dieser Typ ist mit dem der Variablen, den wir ja schon kennen, identisch. Als nächstes stoßen wir auf einen ungewohnten Wert, die Länge des Descriptorfeldes, auf die wir später noch zurückkommen. Es folgt die Angabe der Dimensionen.

Wir haben es in unserem Beispiel mit einem dreidimensionalen Feld zu tun, das in jeder Dimension 13 Elemente hat. Die Anzahl der Dimensionen — und der maximal möglichen Elemente je Dimension — finden wir als Ein-Byte— beziehungsweise Zwei-Byte-Wert angegeben.

Zum Schluß folgt die Angabe der Position des ersten Strings (Speicherstelle 578). Jede einzelne Zeichenkette ist, wie bei den einfachen Strings, mit seiner Länge und Position aufgeführt. Begonnen wird dabei mit dem niedrigsten Index, also der Null. Bei mehrdimensionalen Feldern gilt darüber hinaus die Regel: Vorderer Index kommt vor hinterem Index. Wenn wir also in unserem Fall mit drei Dimensionen arbeiten, so kommt zunächst die Variable mit dem Index (0,0,0), dann die mit (1,0,0). Der erste Index wird bis zum Maximalwert erhöht bevor sich der nächste ändert. Während wir meist dazu neigen dem ersten Index die höchste Bedeutung zuzumessen, arbeitet der CPC bei der Speicherung genau umgekehrt. Hier ist die letzte Angabe die wichtigste.

Dieses Wissen ist besonders bei größeren Feldern von Nutzen. Will man nämlich mit einem Maschinencode-Programm auf ein solches Feld zugreifen, so sollte man die Terminologie des Schneider übernehmen. Es ist sinnvoll dem hintersten Index die höchste Ebene zuzuordnen. Sonst sind die verschiedenen Strings, die man selbst hintereinander wählt, nicht wie erwartet schön sauber hintereinander mit ihren Zeigern abgelegt, sondern wild im gesamten Variablenfeld verstreut. Dies bedeutet, daß man die Position jedes Wertes immer neu berechnen muß. Ein unnötiger Aufwand, der ein Programm nur langsamer macht.

Um uns mit der Abfolge der einzelnen Variablen etwas näher vertraut zu machen, gehen wir im Speicher ein paar Adressen weiter. Zwischen 581 und 583 folgt der nächst String :m$(1,0,0). Diesen hatten wir nur auf ein Zeichen, die »1«, definiert. Als Länge finden wir deshalb eine 1 und 463 (=1x256 + 207) gibt uns die Position des Strings (im Programm da ja noch keine Änderung erfolgt ist) an. Definieren Sie doch einmal ein paar weitere Strings und schauen Sie dann nach, wo diese im Speicher auftauchen.

Abschließend wollen wir auf die Größe des Feldes, das heißt die Speicherstellen 569 und 570, eingehen. Sie enthalten die gesamte Länge des für die Zeiger der Strings benötigten Feldes. Wie wir gesehen haben, benötigt jeder String für die Abspeicherung drei Bytes (seine Länge und die Positionsangabe im Stringspeicher beziehungsweise im Basic-Quelltext als absolute Adresse). Bei drei Dimensionen und jeweils 13 erlaubten Indizes benötigen wir somit 6591 Byte (=3x13x13x13). Addieren wir noch die für die Dimensionsangaben benötigten sieben Bytes (zwischen 571 und 577), so erhalten wir als Ergebnis 6598 Byte und dies ist genau die Zahl, die wir oben in der Länge des Descripterfeldes als 25 * 256 + 198 wiederfinden.

Wie sieht das Ganze nun für die anderen beiden Variablentypen, das heißt Integer- und Realvariable aus? Wir können dies mit wenigen Änderungen überprüfen. Das neue Programm sieht wie folgt aus:

```
5 DIM m (12,12,12)
7 m (0,0,0)=999:m(12,12,12,)=7
8 m (1,0,0)=1
10 FOR i=PEEK($AE87)+256*PEEK
(&AE88) TO PEEK(&AE89)+256*
PEEK(&AE8A)
20 PRINT i,PEEK(i),HEX$(PEEK(i)):
NEXT
```

Es wird nun ein Feld für Fließkommavariable mit drei Dimensionen und jeweils 13 Elementen angelegt und seine Struktur aufgezeigt. Da unser Programm durch die Änderung etwas kürzer geworden ist, beginnt die Abspeicherung der Arrays jetzt schon ab Speicherstelle 543. Wenn Sie sich das Ergebnis anschauen, so fallen die Parallelen zu den Strings sofort ins Auge.

Der Kopf ist dem eines Stringfeldes sehr ähnlich. Die einzelnen Fließkommawerte sind wie beim Speichern einzelner Variabler abgelegt, beginnend mit m(0,0,0). Natürlich ist der Variablentyp geändert, und auch die Größe des Feldes hat sich erweitert. Dies liegt daran, daß, wie wir ja schon gesehen haben, eine Realvariable fünf Bytes Speicherplatz benötigt. Der Platzbedarf berechnet sich wie folgt:

Abzuspeichernde Variable:	
13 * 13 * 13 *	5 Bytes
	= 10985 Bytes
+Dimensionierung	= 7 Bytes
	10992 Bytes

Diese Zahl könnern wir auch als 42 * 256 + 240, also die Größe unseres Feldes, bestimmen.

Das gleiche Prinzip finden wir auch bei den Integerarrays wieder. Das Feld ist aber kleiner als bei den Stringvariablen, da ja nur zwei Bytes im Speicher benötigt werden. Als Kennzeichen finden wir die 1 für Integer und nach den Dimensionsangaben dann im 2-Byte-Versatz die Variablen. Wir können uns dies anschauen, indem wir einfach in dem letzten Programm hinter jeder Variable noch das Prozentzeichen für Integervariablen einfügen.

Wir wissen nun, wie Variable und Felder im Schneider gespeichert sind. Damit stellt sich aber nun eine Frage: Was fangen wir eigentlich mit unserem Wissen an?

Ein Anwendungsbereich bietet sich natürlich sofort an. Es ist dies die Kommunikation zwischen Basic

Position	dez	Kommentar
543	0	
544	0	
545	205	Name "M" + 128
546	4	Variablentyp Real
547	240	Größe des Feldes 10992 Bytes
548	42	
549	3	Dimensionen
550	13	1. Dimension
551	0	Trennungsnull
552	13	2. Dimension
553	0	Trennungsnull
554	13	3. Dimension
555	0	Trennungsnull
556	0	1. Variable m(0,0,0) Mantisse Byte 1
557	0	Mantisse 2
558	192	Mantisse 3
559	121	Mantisse 4
560	138	Exponent
561	0	Mantisse 1 2. Variable m(1,0,0)
562	0	Mantisse 2
563	0	Mantisse 3
564	0	Mantisse 4
565	129	Exponent

So werden Variablenfelder gespeichert

und Maschinensprache. Einige Worte dazu wurden schon am Anfang gesagt. Es ist für eine Maschinenroutine einfach auf einen String zurückzugreifen und zum Beispiel nach einem Zeichen ab dem Ende zu suchen. Jedoch ist auch der umgekehrte Weg möglich.

Aber auch bei reinen Basic-Programmen ist das Wissen um den Aufbau von Feldern nicht nutzlos. Mit ein paar Tricks soll daher zum Abschluß gezeigt werden, wie man durch direkten Zugriff Nutzen für seine Programme erzielt.

Nehmen wir einmal an, Sie haben ein Textverarbeitungsprogramm geschrieben, das die einzelnen Textteile in Strings ablegt. Die verschiedenen Zeilen sollen dabei als Zeichenketten hintereinander in einem Feld von, zum Beispiel 500 Variablen, abgespeichert sein. Nun stellen Sie nachträglich fest, daß Ihnen in der Mitte noch eine freie Zeile fehlt. Es gibt zwei Möglichkeiten, den Fehler zu beheben. Zum ersten könnten Sie mit einer FOR...TO...-Schleife alle höheren Zeilen um eine Position nach oben schieben und die aktuelle löschen. Oder aber Sie ändern einfach nur die Pointer.

Die Position des Strings, der die aktuelle Zeile enthält (zum Beispiel z$(20)) erhalten wir mit »PRINT @z$(20)«. Diese Adresse enthält die Länge unseres Strings. Drei Byte höher liegt die nächste Variable. Was wir nun noch benötigen, ist die Adresse der höchsten Zeile. Nehmen wir einmal an, daß bis zu diesem Zeitpunkt 100 Zeilen eingegeben sind, so ist der letzte String z$(100) und seine Position im Speicher können wir mit »PRINT @z$(100)« feststellen. Da noch zwei Byte für die Abspeicherung der Position unseres Strings benötigt werden, reicht das benötigte Descriptorfeld (die Summe aller Zeiger) zwei Byte weiter. Jetzt können wir die Einfügung der Zeile mit einer ganz einfachen FOR...TO...-Schleife durchführen. »FOR i=@z$(100)+2 TO@z$(20) STEP -1:POKE i+3,PEEK (i):NEXT i«

Das ist alles. Wenn Sie einmal ein Testfeld mit DIM z$(500) eröffnen, dann einige Variable definieren (zum Beispiel z$(19), z$(20), z$(21), z$(99), z$(100), z$(101), z$(102), so werden Sie feststellen, daß nach dem Durchlauf der FOR...TO...-Schleife der Stringbereich mit den Indices von 20 bis 100 wie gewünscht verlagert ist. Die alte Zeile 101 existiert nicht mehr. Sie wurde mit dem Inhalt von Zeile 100 überschrieben. Zeile 20 existiert jetzt doppelt als Zeile 20 und Zeile 21. Zeile 19 hat ihren alten Platz und Inhalt behalten. Nun soll Zeile 20 aber auf »leer« gesetzt werden. Auch das ist mit einer einfachen Schleife möglich.

»FOR i=@z$(20) TO@z$(20)+2: POKE i,0:NEXT i« macht das. Diese Schleife setzt einfach die drei Angaben für Länge und Stringposition auf Null. Damit wird der zur Zeile 20 gehörende String als nicht existent gekennzeichnet, also gelöscht. Sie können diese Art Variablen zu löschen, speziell in Feldern, natürlich auch im größeren Rahmen anwenden. Muß zum Beispiel eine Reihe von Meßwerten in einem Feld auf Null gesetzt werden, dann ist dieses Verfahren deutlich schneller als eine Neudefinition aller Variable. Eine andere nützliche Anwendung ist ein eingeschränktes CLEAR. Mit CLEAR werden alle Variablen gelöscht. Mit Hilfe einer solchen POKE-Schleife kann man differenzieren. Ihnen werden sicher noch andere Anwendungsmöglichkeiten einfallen, wenn Sie ein wenig mit diesen Methoden arbeiten und experimentieren.

(Carsten Straush)

So funktioniert die Stringverarbeitung

Anspruchsvolle, schnelle Stringverarbeitung in Basic ist nicht ganz einfach. Hier finden Sie Tips, um mit dem Basic der Schneider-Computer Zeichenketten optimal zu bearbeiten.

Ein String ist nichts anderes als eine beliebige Folge von Buchstaben, Zahlen, Sonderzeichen und grafischen Symbolen, die beim Schneider aus 256 Zeichen ausgewählt werden können. Die maximal zulässige Länge eines Strings ist dabei auf 255 Zeichen beschränkt.

Die im Bedienungshandbuch aufgeführten Basic-Befehle zur Stringverarbeitung (wie wir später noch sehen werden, verschweigt uns das Handbuch einige interessante Befehle) lassen kaum Wünsche offen. Strings können verknüpft werden (einfach mit dem »+«-Zeichen), es können Teilstrings in allen Variationen gebildet werden (LEFT$, RIGHT$ und MID$) und es können Zahlen in Strings und Strings in Zahlenwerte verwandelt werden (STR$ und VAL).

Prinzipiell ist der Anwender oder besser der Basic-Programmierer dabei keinen Beschränkungen unterworfen. Praktisch jedoch muß sich jeder, der etwas umfangreichere oder anspruchsvollere Stringverarbeitung programmiert (beispielsweise Text- oder Tabellen-Kalkulationsprogramme), mit einer unangenehmen Eigenart der meisten Basic-Dialekte herumschlagen: der sogenannten »Garbage-Collection«.

Diese äußert sich in sekunden- oder sogar minutenlangem »Stehenbleiben« des Programms. Der Computer reagiert während dieser Zeit auf keinerlei Eingaben mehr, der Bildschirm erscheint wie eingefroren. Wenn Sie das bisher noch nicht erlebt haben sollten, dann tippen Sie doch einmal folgendes kleines Programm in Ihren Schneider ein:

```
10  DEF INT A-Z: MEMORY
    & A000
15  DIM A$(500): B$ =
    SPACE$(70)
20  FOR I=1 TO 500: PRINT I,
    FRE(0):A$(I) = B$:NEXT
25  PRINT"JETZT GEHT'S
    GLEICH LOS..."
30  PRINT"BITTE TASTE
    DRUECKEN"
35  I$ = INKEY$ : IF I$ =
    "" THEN 35
40  FOR I = 0 TO 499: PRINT I,
    FRE(0) : A$(I) = A$(I+1):NEXT
45  GOTO 40
```

Wenn Sie dieses Programm starten, können Sie zunächst mitverfolgen, wie durch die Wertzuweisungen an die einzelnen Elemente des Stringfeldes A$ der freie Speicherplatz immer kleiner wird. Nachdem alle Strings den Wert von B$ haben (nämlich 70 Leerzeichen), sollten Sie eine Taste drücken. In einer Endlosschleife (Zeilen 40 bis 45) werden weitere Wertzuweisungen an Strings vorgenommen. Am Bildschirm sehen Sie den freien Speicherplatz weiter zusammenschrumpfen — bis die Auflistung auf einmal unvermittelt stehenbleibt, und sich für fast eine halbe Minute nichts mehr auf dem Bildschirm tut.

Was ist passiert? Der freie Speicherplatz reicht nicht mehr aus, um noch eine Wertzuweisung an einen String vornehmen zu können. Der Computer versucht jetzt, durch eine Neu-Organisation des Speichers, eben die sogenannte Garbage Collection, wieder genügend freien Speicher zu schaffen, um die Wertzuweisung doch noch vornehmen zu können. Diesen Vorgang wollen wir jetzt einmal etwas genauer unter die Lupe nehmen.

Die Müllabfuhr im Computer

Bei jeder Veränderung eines Strings bleibt der alte und nunmehr ungültige Inhalt im Speicher stehen, das ist der »Garbage« (Müll). Ist durch eine entsprechend hohe Anzahl von Stringveränderungen der gesamte verfügbare Speicher mit gültigen und ungültigen Strings gefüllt, muß, um wieder Platz zu schaffen, dieser »Müll« aufgesammelt und beseitigt werden. Dies ist die Garbage-Collection (die übrigens auch mit FRE("") gezielt eingeleitet werden kann). Hierbei wird zunächst der gültige String mit der höchsten Adresse gesucht und an die höchste zur Ablage verfügbare Speicherstelle gelegt (Strings werden im Speicher »von oben nach unten« abgelegt). Die Zeichenketten zwischen diesem gültigen String (mit der höchsten Adresse) und der höchsten zur Stringablage erlaubten Adresse sind logischerweise »Müll«. Anschließend wird der nächste gültige String mit der nächstniedrigeren Adresse gesucht und unter dem ersten direkt angefügt. Und so geht es weiter, bis alle gültigen Strings lückenlos aneinandergereiht sind. Dieses Suchen kostet Zeit, gelegentlich mehrere Minuten. Leider gibt es keine Möglichkeit, die Garbage Collection abzukürzen, denn der Computer muß jeden einzelnen String überprüfen und mit allen noch nicht überprüften Strings adressenmäßig vergleichen. Die für die Garbage Collection benötigte Zeit ist also ausschließlich von der Gesamtzahl aller definierten Strings abhängig, nicht jedoch vom freien Speicher oder von der Anzahl der Stringoperationen im Programm. Lediglich die Zeit bis zum Einsetzen der nächsten Garbage Collection wird von den letzten beiden Faktoren beeinflußt.

Wir wollen nun eine Reihe von typischen Stringverarbeitungen untersuchen und Möglichkeiten aufzeigen, die »Stringmüll«-Produktion so niedrig wie möglich zu halten.

Zuvor jedoch müssen zwei Basic-Befehle erläutert werden, die sich (noch) nicht allzu großer Bekanntheit erfreuen, da sie im Bedienungshandbuch einfach weggelassen worden sind.

Zunächst ist da der »Klammeraffen«-(@-)Befehl. Mit diesem hat es eine interessante Bewandtnis. Zu einer Stringvariablen gehört nicht nur die irgendwo im Speicher abgelegte Zeichenkette, sondern auch der sogenannte »Stringdescriptor«. Dieser besteht aus drei Byte. Das erste gibt die Länge des Strings an, die anderen beiden zeigen auf die Adresse, an welcher das erste Zeichen steht. Trifft der Basic-Interpreter auf eine Stringvariable, so sucht er zuerst den Stringdescriptor (der in einem bestimmten Bereich beim ersten Auftreten der Variablen vom Interpreter angelegt wird) und weiß nun, wo der gesuchte String steht und wie lang er ist. Zeichenketten im Speicher, die nicht von einem Stringdescriptor abgedeckt werden, sind ungültig. Wird nun der Klammeraffe einer Stringvariablen vorangesetzt, wird die Adresse des ersten Bytes des Stringdescriptors zurückgegeben. Versuchen Sie folgendes kleine Programm:

```
10  A$=STRING$(5,"5")
20  PRINT@A$
30  PRINT PEEK(@A$)
40  PRINT PEEK(@A$ + 1)
    + 256*PEEK(@A$ + 2)
```

In Zeile 20 wird die Adresse des Stringdescriptors ausgegeben, in Zeile 30 der Inhalt dieser Adresse (mithin die Länge von A$) und in Zeile 40 die Adresse des ersten Zeichens von A$. Wenn Sie diese und die vier folgenden Adressen »PEEKen«, erhalten Sie fünfmal den Wert 53, der den ASCII-Wert von »5« darstellt. Vielleicht ahnen Sie jetzt schon, was man mit der Manipulation dieser Stringdescriptoren alles anstellen kann. Der @-Befehl arbeitet übrigens nicht nur bei Stringvariablen, sondern ergibt generell die Adresse, an der eine Variable im Speicher steht.

Ein weiterer effizienter Befehl ist die Zuweisung von Zeichen in einen bestehenden String hinein. Dies geschieht mit dem bekannten Befehl MID$, nur daß er diesmal links vom Gleichheitszeichen (besser: Zuweisungszeichen) steht. Beispiel:

```
10  A$=STRING$(10,"x")
20  MID$(A$,5,2)="aa"
30  PRINT A$
```

In Zeile 20 wird der Variablen A$ ab der fünften Stelle der zwei Stellen lange String »aa« zugewiesen. Der dritte Wert in MID$ kann weggelassen werden, dann wird bis zum Ende von A$ übertragen (wenn der Zuweisungsstring entsprechend lang ist). Bei dieser Art der Stringzuweisung wird kein neuer String erzeugt, es fällt also auch kein »Müll« an. Ohne diesen dritten Wert in MID$-Befehl würde unsere kleine Austauschoperation viel komplizierter ausfallen:

```
20  A$=LEFT$(a$,4)+"aa"+MID$
    (A$,7)
```

Der hierbei anfallende »Stringmüll« wäre beträchtlich: Werden nämlich mehr als zwei Strings verknüpft, so wird für jede weitere Verknüpfung ein neuer String angelegt, der nach der Zuweisung als »Garbage« zurückbleibt. Dazu kommt, wie bereits bekannt, der alte Inhalt von A$.

Müll muß nicht sein

Wird in unser kleines Beispiel der Befehl »PRINT FRE(0)« als Zeile 15 und Zeile 40 eingefügt, wird der verfügbare Speicherplatz vor und nach der Austauschoperation angezeigt. Die Differenz ist Abfall (dies ist zwar nur dann völlig korrekt, wenn die Länge von A$ vor und nach der Zuweisung gleich ist, genügt aber für unsere Betrachtung).

Anzumerken ist noch, daß der MID$-Befehl auf der linken Seite einer Wertzuweisung nur bei Stringvariablen angewendet werden kann, die bereits durch eine Standardzuweisung (zum Beispiel »A$=B$« oder »A$="HALLO"«) einen Wert erhalten haben.

Ein Grundsatz sollte in jedem Fall sein, mehr als zwei Strings nur sehr selten zu verknüpfen. Mehrere MID$-Befehle tun im allgemeinen das gleiche. Zu beachten ist auch, daß STRING$- und SPACE$-Befehle in einer Verknüpfung zusätzlichen

Speicherplatz belegen — und zwar in Größe des von ihnen erzeugten Strings.

Wir wollen nun alle Operationen, die einen String verändern, in zwei Klassen unterteilen. Wir betrachten auf der einen Seite die Operationen, die die Länge des Strings unverändert lassen, auf der anderen Seite solche, die die Länge des Strings verändern. Zwei kurze Beispiele zeigen, wie dies zu verstehen ist: Die Standardzuweisung: A$=B$ verändert die Länge von A$, falls A$ und B$ vor der Operation unterschiedliche Länge hatten, andernfalls nicht. Bei A$=MID$(B$,4,22) wird die Länge von A$ nicht verändert, falls A$ vor der Operation genau 22 Zeichen lang war. Beide Standardzuweisungen erzeugen »Stringmüll«, unabhängig davon, ob die Länge von A$ nun verändert wird oder nicht.

Bei allen Operationen, die die Länge eines Strings nicht verändern, kann die Standardzuweisung durch den MID$-Befehl in der Form »MID$(A$,1)=B$« ersetzt werden. Das Ergebnis ist das gleiche. Anwendungen können Austauschoperationen sowie Zuweisungen von gleichlangen Strings sein. Schwieriger wird es, Speicherplatz zu sparen, wenn die Länge des Strings verändert wird.

Ist der neue String kürzer, so wird er zunächst durch einen oder mehrere MID$-Befehle erzeugt. Anschließend POKEn wir an die Adresse des ersten Bytes des Stringdescriptors (Länge) die neue, kürzere Länge, und fertig ist unsere neue Zeichenkette. Wieder ganz ohne »Garbage«-Produktion.

Ist der neue String hingegen länger, so bleibt uns meistens nur der saure Apfel der Standardzuweisung mit der damit verbundenen Abfall-Produktion. Stringverknüpfungen lassen sich aber auch hier optimieren, indem man in zwei Schritten vorgeht. Zunächst wird der Variablen eine beliebige Zeichenkette mit der Gesamtlänge aller zu verknüpfender Strings zugewiesen. Anschließend werden mit mehreren MID$-Befehlen die zu verknüpfenden Strings an die entsprechenden Stellen übertragen. Als »Müll« fällt nur der alte Inhalt unserer Variablen an. Beispiel:

```
10   A$=B$+C$+D$ wird ersetzt
     durch
10   A$=SPACE$(LEN(B$)+LEN
     (C$)+LEN(D$))
20   MID$(A$,1)=B$+C$
30   MID$(A$,LEN(B$)+LEN(C$)
     +1)=D$
```

Ob sich das ungleich umfangreichere Programm lohnt, muß von Fall zu Fall entschieden werden. Noch komplizierter wird die Angelegenheit, wenn ein String verlängert werden muß und der alte Inhalt nicht verlorengehen darf. Beispiel:

```
10 A$=A$+B$+C$
```

Würden wir, wie gerade beschrieben, Speicherplatz sparen wollen, wäre A$ zerstört, bevor es aufgerufen wird, da ihm schon vorher ein neuer Wert zugewiesen wird. Hier hilft eine beliebige Hilfsvariable (nur ein String muß es sein, und dem Interpreter beispielsweise durch eine Leerzuweisung wie X$="" schon bekannt sein, damit dieser den Stringdescriptor angelegt hat). Wir kopieren den Stringdescriptor von A$ an die Stelle der Hilfsvariablen X$. Jetzt haben wir zwei Variablen, die auf den gleichen String zeigen. Zerstören wir nun A$, zeigt X$ immer noch auf die benötigte Zeichenkette im Speicher. Unser Programm sieht dann folgendermaßen aus:

```
10  POKE@X$,PEEK(@A$)
20  POKE@X$ + 1,PEEK
    (@A$ + 1)
30  POKE@X$ + 2,PEEK
    (@A$ + 2)
40  A$=SPACE$(LEN(X$)+LEN
    (B$)+LEN(C$))
50  MID$(A$,1)=X$+B$
60  MID$(A$,LEN(X$)+LEN(B$)
    +1)=C$
```

Ein völlig anderes, jedoch in den meisten Fällen gleichwertiges Verfahren macht sich eine Besonderheit des Basic-Interpreters zunutze. Dieser legt bei Standardzuweisungen jeden String der Reihe nach von oben nach unten im Speicher ab. Wir rollen nun einfach die Verknüpfungen von hinten auf und ersetzen sie durch Standardzuweisungen. Zum Schluß erhält unser neuer String mittels POKE noch die richtige Länge. Wir können dann für das erste Beispiel schreiben:

```
10  A$=C$+D$
20  A$=B$
30  POKE @A$,LEN(B$)+LEN
    (C$)+LEN(D$)
```

In Zeile 10 wird der String C$+D$ angelegt, dabei ist es völlig gleichgültig, ob die Zuweisung schon auf A$ erfolgt oder auf irgendeine andere Variable. Es kommt auf die Zuweisung an sich an. In Zeile 20 wird direkt an C$+D$ anschließend (es darf sich zwischen Zeile 10 und 20 keine weitere Zuweisung befinden) der String B$ angelegt. Nun befinden sich B$, C$ und D$ direkt nacheinander im Speicher. Da die letzte Zuweisung bereits auf A$ erfolgte, ist der Stringdescriptor von A$ bis auf die Länge bereits in seinem Endzustand. In Zeile 30 wird er noch auf die richtige Länge gesetzt. Die der ursprünglichen Verknüpfungsreihenfolge entgegengesetzte Folge der Zuweisungen ist unbedingt erforderlich, da wie gesagt, die Zeichenketten im Speicher von oben nach unten abgelegt werden. In unserem Beispiel darf anstelle von B$ keinesfalls ein String in Hochkommata (zum Beispiel »Hallo«) stehen. Wird nämlich ein solcher String, ohne daß er Teil einer Verknüpfung ist, in einer Standardzuweisung angelegt, so zeigt der Stringescriptor der Variablen anschließend mitten in das Basic-Programm, an genau die Stelle, wo der String steht. Ein anschließendes Vergrößern der Länge führt zu einem unsinnigen Ergebnis.

Das zweite Beispiel kann folgendermaßen aussehen:

```
10  A$=B$+C$
20  A$=A$
30  POKE@A$,LEN(A$)+
    LEN(B$)+LEN(C$)
```

Wir sehen, die Methode ist kürzer als die erste und in sicherlich fast allen Fällen gleichwertig. Dennoch ist die erste Methode nicht ganz wertlos. Sie ist leichter nachzuvollziehen und vertieft unser Verständnis von der Art und Weise, mit welcher der Basic-Interpreter Zeichenketten behandelt.

Ein besonderer Fall einer Stringzuweisung liegt dann vor, wenn nach der Zuweisung der Quellstring nicht mehr benötigt wird (unter Quellstring oder -variable wollen wir den String rechts vom Zuweisungszeichen verstehen, als Zielstring oder -variable den String links davon). Dies ist öfter der Fall, als es zunächst den Anschein haben mag. Dann kann nämlich immer und unabhängig von der Stringlänge eine Stringzuweisung ohne »Abfall«-Produktion erfolgen. Der Stringdescriptor des Quellstrings wird auf den des Zielstrings kopiert (siehe auch Zeile 10 bis 30 aus dem drittletzten Programm). Anschließend wird die Länge des Quellstring auf »0« gesetzt (mit dem entsprechenden POKE) und der Quellstring damit neutralisiert. Wird dies unterlassen, so zeigen bis zur nächsten Garbage-Collection oder bis zur nächsten Standardzuweisung Ziel- wie Quellvariable auf den gleichen String.

Die Änderung der einen wirkt sich sofort auch auf die andere Variable aus. In besonderen Fällen mag dies sogar von Vorteil sein. Aber wie gesagt, nur bis zur nächsten Garbage-Collection beziehungsweise Standardzuweisung. Dann werden die Variablen wieder getrennt.

Schnelle Strings durch Assembler

Wem all diese Kniffe zur Umgehung der Garbage-Collection noch nicht ausreichen, und wer sein Programm noch weiter beschleunigen möchte, dem mögen die folgenden beiden Assemblerroutinen eine hilfreiche Stütze und Anregung sein.

Das Kopieren von Stringdescriptoren ist in Basic eine ziemlich zeitraubende Angelegenheit. Drei PEEKs und POKEs, sechs Klammeraffen-Befehle und vier Additionen wollen interpretiert und ausgeführt sein. Wesentlich schneller ist folgende kleine Routine. Sie kann an jeder beliebigen Stelle im Speicher stehen und kopiert den Stringdescriptor der ersten Variablen auf die zweite Variable. Es kann in obigem Programm die POKE-Befehle in den Zeilen 10 bis 30 ersetzen. Der Aufruf erfolgt mit »CALL ADR,@VON$,@NACH$« wobei VON$ und NACH$ beliebige, schon eröffnete Stringvariablen sind.

```
adr: DD 56 01  ld d,(IX+1)
     DD 5E 00  ld e,(IX+0)   de=@nach$
     DD 66 03  ld h,(IX+3)
     DD 6E 02  ld l,(IX+2)   hl=@von$
     01 03 00  ld bc,3       Zähler auf 3
     ED B0     ldir          Blocktransfer
                             3 Byte
     C9        ret
```

Die nächste Assemblerroutine ist schon wesentlich komplexer, spart allerdings auch im Bedarfsfall gut und gerne 99 Prozent der Zeit, die ein gleichwertiges Basic-Programm braucht. Mit dieser Routine wird nicht nur ein Stringdescriptor kopiert, sondern mehrere auf einmal. Stellen wir uns ein eindimensionales Stringarray mit 500 belegten Elementen vor. Wollen wir jetzt das 36. Element löschen und alle folgenden um einen Index aufrücken lassen, so hätten wir 464mal eine Schleife zu durchlaufen und den Stringdescriptor von dem Element n+1 auf das Element n zu kopieren. Zum Abschluß muß noch das 500. Element gelöscht werden. Ein anderes Beispiel: Wir wollen zwischen das 36. und 37. Element ein neues einfügen. Es ist das gleiche Problem mit vertauschten Vorzeichen.

Nun geht aber der Basic-Interpreter beim Anlegen der Stringdescriptoren eines Arrays (wir reden hier nur von einem eindimensionalen Array) schön logisch und einfach vor: die Stringdescriptoren der einzelnen Elemente sind, mit dem von Element 0 beginnend, aufsteigend und lückenlos aneinandergereiht. Steht also an der Adresse 1200 der Stringdescriptor von Element 7 eines Array, so steht der Stringdescriptor von Element 13 an der Adresse 1218 (drei Byte je Stringdescriptor). Wollen wir nun in ein bestehendes Stringarray neue Elemente einfügen oder Elemente löschen, so müssen wir nur die Stringdescriptoren entsprechend verschieben beziehungsweise umkopieren und die Stringdescriptoren der eingefügten beziehungsweise gelöschten Elemente auf 0 setzen. Dies alles macht die nachfolgende kleine Assemblerroutine. Sie kann überall im Speicher stehen und wird mit »CALL ADR,@ARRAY$ (VON/),@ARRAY$ (NACH/),LAENGE« aufgerufen.

Zwei Beispiele sollen die Anwendung verdeutlichen. ARRAY$ sei ein mit 1 000 Elementen definiertes, eindimensionales Array. Es sei mit 500 gültigen Elementen belegt, das heißt, die Elemente 1 bis 500 sind besetzt. Die Elemente 36 bis einschließlich 45 sollen gelöscht werden. Es müssen also die Zeichen ab 46 nach 36 kopiert werden. Die Anzahl der zu kopierenden Stringdescriptoren beträgt 455, das ergibt eine zu kopierende Länge von 1 365 Byte. Gelöscht werden anschließend die Elemente 491 bis 500. Der Aufruf sieht also folgendermaßen aus: »CALL ADR,@ARRAY(46),@ARRAY(36),1365«

Sollen nun im umgekehrten Fall zwischen das 36. und das 37. Element neun neue eingefügt werden, so sind die Elemente von 500 abwärts bis 37 nach Element 509 zu kopieren. Die Anzahl der zu kopierenden Stringdescriptoren beträgt 464, die gesamte Länge mithin 1 392 Byte. Gelöscht werden anschließend die Elemente 37 bis 45. Unser Aufruf: »CALL ADR,@ARRAY(500),@ARRAY(509), 1392«

Der Anwender muß in jedem Fall vor dem Aufruf darauf achten, daß die folgende Randbedingungen eingehalten werden, andernfalls besteht die Gefahr, daß der Computer sich verabschiedet:

Die zu kopierende Länge darf nicht kleiner als drei sein (es muß al-

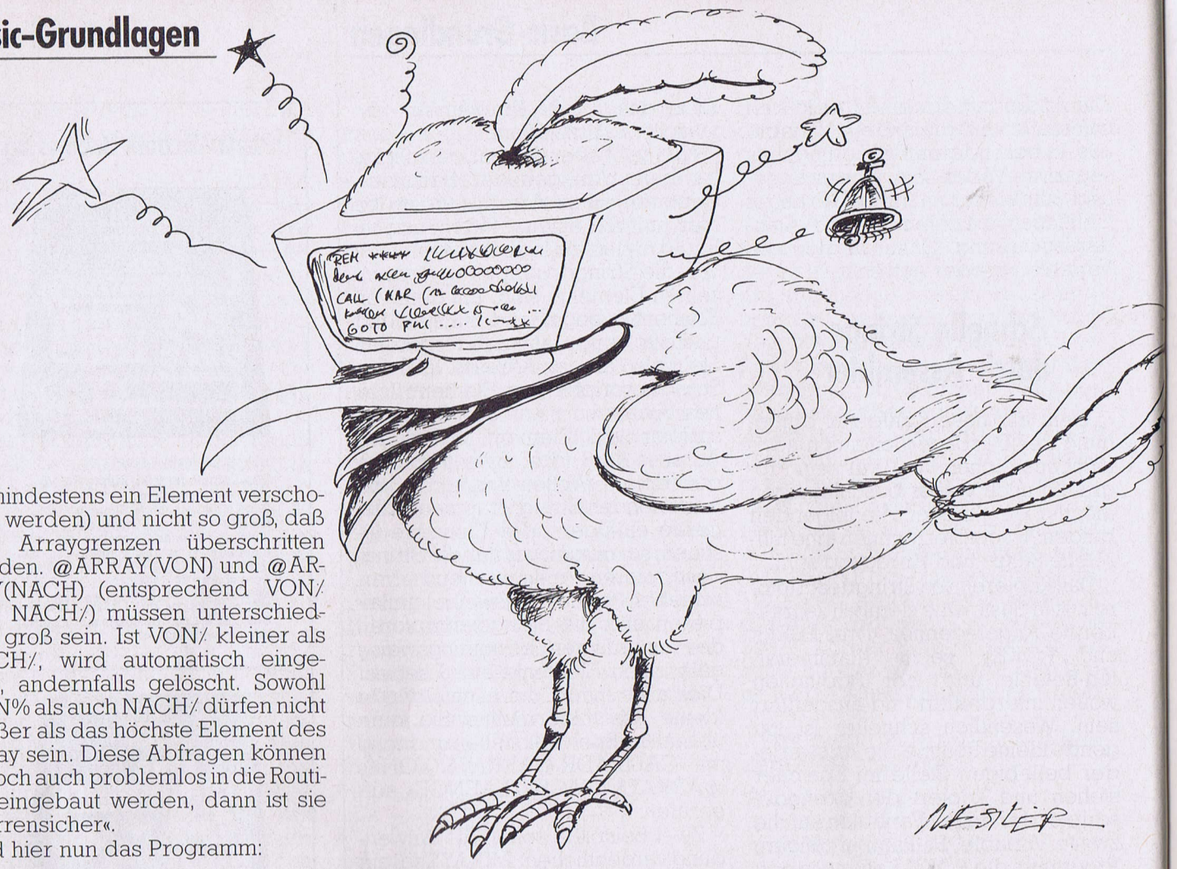
Markt&Technik-Buchverlag

H. L. Schneider/W. Eberl
Das Commodore 64-Buch, Bd. 5
Juli 1984, 322 Seiten
Ein Leitfaden durch Simon's Basic · ausführliche Besprechung aller Befehle · viele erklärende Beispiele · mit kommentierter Assembler-Listing · das richtige Nachschlagewerk für den geübten Commodore 64-Benutzer.
Best.-Nr. MT 599 (Buch) **DM 38,—**
(Sfr. 35,—/öS 296,40)
Best.-Nr. MT 600 (Beispiele auf Diskette)
(Sfr. 58,—/öS 522,—) **DM 58,—**

H. L. Schneider/ W. Eberl
Das Commodore 64-Buch, Bd. 6
1984, 190 Seiten
Programmieren auf dem Commodore 64 spielend gelernt · Programmlisting mit anschließender Programmbeschreibung · Variablenübersicht · Tips zum Ändern und Ergänzen des Programms.
Best.-Nr. MT 619 (Buch) **DM 38,—**
(Sfr. 35,—/öS 296,40)
Best.-Nr. MT 620 (Beispiele auf Diskette)
(Sfr. 58,—/öS 522,—) **DM 58,—**

H. L. Schneider
Das Commodore 64-Buch, Bd. 7
August 1984, 210 Seiten
Der Commodore 64 als Klaviatur · Noten schreiben mit hochauflösender Grafik · relative Dateien am Beispiel einer kleinen Adreßverwaltung · Joystick und Paddles · Grafikspeicher unter Kernal · Interrupt-Manager · für Profis.
Best.-Nr. MT 731 **DM 38,—**
(Sfr. 35,—/öS 296,40)
Best.-Nr. MT 784 (Beispiele auf Diskette)
(Sfr. 38,—/öS 342,—) **DM 38,—**

Markt&Technik
Verlag Aktiengesellschaft
Buchverlag
Hans-Pinsel-Straße 2,
8013 Haar bei München,
☎ 089/4613-220
Schweiz: Markt&Technik-Vertriebs AG, Kollerstraße 3, CH-6300 Zug, ☎ 042/223155
Österreich: Rudolf Lechner & Sohn, Heizwerkstraße 10, A-1232 Wien, ☎ 0222/677526



so mindestens ein Element verschoben werden) und nicht so groß, daß die Arraygrenzen überschritten werden. @ARRAY(VON) und @ARRAY(NACH) (entsprechend VON/ und NACH/) müssen unterschiedlich groß sein. Ist VON/ kleiner als NACH/, wird automatisch eingefügt, andernfalls gelöscht. Sowohl VON% als auch NACH/ dürfen nicht größer als das höchste Element des Array sein. Diese Abfragen können jedoch auch problemlos in die Routine eingebaut werden, dann ist sie »narrensicher«.
Und hier nun das Programm:

```
adr:    DD E5     push IX
        E1        pop hl          hl=IX
        4E        ld c,(hl)
        23        inc hl
        46        ld b,(hl)       bc=Länge
        23        inc hl
        5E        ld e,(hl)
        23        inc hl
        56        ld d,(hl)       de=@array(nach)
        23        inc hl
        6E        ld l,(hl)
        DD 66 05  ld h,(IX+5)     hl=@array(von)
        E5        push hl
        AF        xor a           carry=0
        ED 52     sbc hl,de       vergleiche hl,de
        E1        pop hl          hl wiederherstellen
        30 0B     jr nc,adr1      hl>=de => ldir
        23        inc hl          hl und de
        23        inc hl          auf das
        13        inc de          Ende der String-
        13        inc de          descriptoren setzen
        ED B8     lddr            einfügen
        23        inc hl
        13        inc de
        EB        ex hl,de
        18 02     jr adr2
adr1:   ED B0     ldir            löschen
adr2:   AF        xor a           carry=0, a=0
        D5        push de
        ED 52     sbc hl,de       zu löschende Länge
        E5        push hl
        C1        pop bc          bc=Länge
        0B        dec bc          bc=Länge-1
        E1        pop hl          hl=de
        13        inc de          de=de+1
        77        ld (hl),a       (hl)=0
        ED B0     ldir            löschen
        C9        ret
```

Zum Laden der ersten Assemblerroutine können Sie folgendes Basis-Programm verwenden, das zum Beispiel am Anfang ihres Programmes stehen kann, in welchem Sie die Assemblerroutinen aufrufen wollen:

```
10  MEMORY AFTER 40000

20  DATA &DD,&56,&01,&DD,&5E,
    &00,&DD,&66,&03,&DD,&6E,
    &02

30  DATA &01,&03,&00,&ED,
    &B0,&C9

40  FOR z/=40001 TO
    40018:READ y/:POKE
    z/,y/:NEXT

98  a$="":b$=""

99  CALL 40001,@a$,@b$
```

Analog kann auch die zweite Assemblerroutine in Ihr Basic-Programm eingebunden werden.
Damit ist unser Ausflug in die Geheimnisse der Schneider-Stringverarbeitung beendet. Sicher können Sie die hier gewonnenen Erkenntnisse mit einiger Überlegung auch in Ihren eigenen Programmen verwenden.

(Matthias Engelbach)

Trainieren Sie Ihren Schneider!

Spielerein- und verkäufe, Mannschaftsaufstellungen, Europapokal und Finanzprobleme, das sind die Aufgaben, die Sie bei »Fußball-Manager« lösen müssen. Ein abwechslungsreiches Strategiespiel für drei Leute.

Viele Stammtischdiskussionen gehen um das Thema Fußball. Jeder hätte es besser gemacht, als der zuständige Trainer, aber keiner durfte. Das ändert sich jetzt, denn nun liegt es an Ihnen, die Entscheidungen zu treffen. Zuständig sind Sie für alle Arbeiten, die ein Manager — und Trainer — während einer Saison zu fällen hat.

»Fußball-Manager« simuliert eine Saison in der Bundesliga mit den zugehörigen Pokal- und Europapokalrunden. Bis zu drei Personen können miteinander spielen, die restlichen Vereine übernimmt der Computer. Die Schlagkraft einer Mannschaft hängt von der Stärke der einzelnen Spieler ab. Sie wird bei der Mannschaftsaufstellung angezeigt. Je nachdem wie stark Ihr Team ist und das Ihres Gegners, wird das Ergebnis ausfallen. Ein bißchen Zufall ist natürlich auch im Spiel. Auf dem Transfermarkt können Sie Spieler ein- und verkaufen. Bis zu 20 dürfen Sie gleichzeitig unter Vertrag haben. Aber Vorsicht: Die Gehaltszahlungen steigen natürlich mit der Größe der Mannschaft.

```
10 ' ****** Lader fuer Fussballmanager
*****
20 MODE 2
30 PRINT:PRINT
40 POKE &B1C8,&0:POKE &B1CF,&F0:POKE &B1
D0,&F:PRINT "FUSSBALL-MANAGER    "
50 POKE &B1CF,&E0:POKE &B1D0,&E:PRINT "
FUSSBALL-MANAGER    "
60 POKE &B1CF,&C0:POKE &B1D0,&C:PRINT "
 FUSSBALL-MANAGER   "
70 POKE &B1CF,&80:POKE &B1D0,&8:PRINT "
  FUSSBALL-MANAGER "
80 POKE &B1CF,&A0:POKE &B1D0,&A:PRINT "
   FUSSBALL-MANAGER"
90 PRINT:PRINT
95  POKE &B1C8,&2:PRINT "Original fuer A
pple  von";
100 POKE &B1C8,&1:POKE &B1CF,&C0:POKE &B
1D0,&30:POKE &B1D1,&C:POKE &B1D2,&3:PRIN
T " Bodo Beschke"
110 PRINT: POKE &B1C8,&2:PRINT "Angepass
t an CPC 464 von";
120 POKE &B1C8,&1:PRINT " Alfred Otto"

130 PRINT: POKE &B1C8,&2:PRINT "Pokes fu
er Modes von    ";
140 POKE &B1C8,&1:PRINT " Helmut Tischer
"
150 FOR i=1 TO 2500:NEXT:RUN "MANAGER"
```

Listing 1. Die Laderoutine für »Fußball-Manager«

Das Programm ist menügesteuert und jede Option wird mit einem Tastendruck aufgerufen. Zu Anfang haben Sie eine Mannschaft mit zwei Torhütern, vier Verteidigern, fünf Mittelfeldspielern und vier Stürmern. Ihr Startkapital beträgt zwei Millionen Mark. Nur vor jeder Runde kann man den aktuellen Spielstand speichern, um allen Mitspielern immer die gleichen Bedingungen zuzusichern.

Die Mannschaftsaufstellung mit den aktuellen Daten aller Spieler erfahren Sie über eine Bildschirmausgabe. Ein inverses »>« zeigt die eingesetzten Fußballer an. Um Änderungen vorzunehmen, müssen Sie nur die Nummer des auszutauschenden und des neuen Spielers eingeben. Alle Eingaben erklären sich von selbst. Muß eine Frage mit »Ja« beantwortet werden, so reicht der Druck auf die »J«-Taste. Alle anderen Tasten bedeuten Nein. Verletzte Spieler sind übrigens durch ein »V« gekennzeichnet.

Die Transferliste wird jeden Spieltag, für jeden Manager, bis zu dreimal neu ermittelt. Mit »K« kann man einen Spielerkauf tätigen und das fünfmal pro Spiel. Behalten Sie aber Ihre Finanzen im Auge, da Spieler Geld kosten und außerdem noch Gehalt haben wollen.

Der Spielerreport zeigt die aktuellen Daten der einzelnen Mitglieder Ihres Teams. Gehalt pro Jahr, Zahl der gelben Karten (nach der dritten, siebten und zehnten Karte ist der Spieler automatisch für ein Spiel gesperrt) und der Stärkegrad sind die wichtigen Informationen, die Sie für Ihre Entscheidungen brauchen. Fällt der Stärkegrad unter Null, so ist der Spieler verletzt und kann nicht eingesetzt werden.

Um die Spielstärke Ihrer Mannschaft zu bestimmen, werden die Stärkegrade der einzelnen Spieler addiert. Als Punktwert gilt dafür aber nur der Vorkommateil. »3,87« zählt also für das Team mit »3«.

Zwischen den einzelnen Spieltagen werden die verschiedenen Pokalrunden ausgetragen. Das Spielgeschehen zwischen den Toren wird symbolisch dargestellt. Erreicht der Ball die linke Seite, so hat Ihr Gegner ein Tor geschossen, rechts waren Sie erfolgreich.

Am Ende des Spiels wird der Verlauf der Saison bewertet. Die Punktzahl setzt sich wie folgt zusammen:
— Bundesliga (maximal 75 für die Meisterschaft)
— Vermögen (End- minus Startvermögen geteilt durch 150000)
— Europapokal (maximal 25 für Pokalgewinn)
— DFB-Pokal (maximal 20 für Pokalgewinn)
— Tore (positive Tordifferenz geteilt durch 10).

Die Abschlußtabelle der Simulation ähnelt sehr häufig der einer echten Saison. Wenn Ihre Mannschaft schlecht spielt, so liegt es nicht am Programm, sondern an Ihren Entscheidungen. Also Vorsicht beim nächsten Stammtischgespräch.

Noch ein Wort zum Eingeben. Listing 1 ist der Lader für das Hauptprogramm (Listing 2). Es muß also auf Kassette zuerst gespeichert werden. Das Hauptprogramm trägt den Namen »Manager«.

(Alfred Otto)

```
10 MODE 1:PEN 1
20 INK 3,24,1
30 GOSUB 5410
40 v=1
50 GOSUB 2820
60 x=FRE(0)
70 FOR h=1 TO sp
80 flag=0:flag1=0
90 IF v=1 THEN 110
100 GOSUB 440
110 CLS:PRINT CHR$(24):MID$(x$(sp(h)),3)
:CHR$(24);SPC(5);"Spieltag";v
120 IF v=34 THEN LOCATE 4,22:PEN 3:PRINT
 CHR$(24);"Letzter Bundesligaspieltag !!
!";CHR$(24):PEN 1
```

Listing 2. »Manager«: Werden Sie deutscher Meister mit Ihrer Lieblingsmannschaft

```
130 LOCATE 1,8:PRINT "    1 - Mannschafts
aufstellung"
140 PRINT "    2 - Transfermarkt"
150 PRINT "    3 - Bundesligatabelle"
160 PRINT "    4 - Spielerreport"
170 PRINT "    5 - Allgemeines"
180 PRINT "    6 - Vereinsuebersicht"
190 PRINT "    7 - Spielpaarungen"
200 PRINT "    8 - Weiter"
210 PRINT "    * - Spielabbruch mit Abrec
hnung"
220 IF flag=0 AND h=1 THEN PRINT "    S -
 Sichern"
230 IF flag1=0 THEN GOSUB 710
240 LOCATE 1,4:PRINT"Ihre Entscheidung?"
:TAB(27);"[":PEN 3:PRINT CHR$(24);" ";C
HR$(24);:PEN 1:PRINT"]";
250 es$=INKEY$:IF es$="" THEN 250 ELSE L
OCATE 28,4:PRINT es$
260 IF es$="*" THEN 4240
270 IF NOT (flag=0 AND h=1) THEN GOTO 28
0 ELSE IF es$="S" OR es$="s" THEN 5840 E
LSE GOTO 280
280 IF VAL(es$)<1 OR VAL(es$)>8 THEN PRI
NT CHR$(7):: GOTO 240
290 es=VAL(es$)
300 flag=255
310 ON es GOTO 800,1300,2490,2750,4640,4
030,1860,320
320 qq=0:tr=0:f=0:c=0:FOR i=1 TO 5:t5(i)
=0:NEXT i
330 NEXT h:GOSUB 1860
340 y=y+1:IF y<=34 THEN 50
350 y=34
360 GOTO 4240
370 ' *** Ende ***
380 CLS:IF y>30 THEN FOR h=1 TO sp:PRINT
 MID$(x$(sp(h)),3):GOSUB 5350:PRINT:NEXT
390 LOCATE 1,16:PRINT CHR$(24);"Das  war
 eine  Begegnung mit  der harten  Wirkl
ichkeit im Fussballgeschaeft!          W
 A G E N   S I E   E I N E             Z
 W E I T E   S A I S O N
      ? ? ?          (J/N)    ";CHR$(2
4);
400 a$=INKEY$:IF a$="" THEN 400 ELSE IF
a$="J" OR a$="j" THEN RUN
410 PRINT "Okay, dann bis zum naechsten
Mal !":END
420 LOCATE 1,24:PRINT CHR$(24);:a$=MID$(
x$(sp(h)),3):GOSUB 4230:PRINT CHR$(18):C
HR$(24):RETURN
430 ' *** Spielerstatus ***
440 CLS:PRINT CHR$(24);:FOR i=1 TO 23:LO
CATE 39,i:PRINT " ":NEXT
450 LOCATE 1,10:PRINT" Spieltag";y"  ":L
OCATE 1,12:PRINT " "MID$(x$(sp(h)),3)" "
;CHR$(24)
460 FOR i=1 TO 20:IF ma(h,i,1)=0 THEN 55
0
470 ma(h,i,1)=ma(h,i,1)+RND(1)-RND(1)
480 IF ma(h,i,1)>=6 THEN ma(h,i,1)=5.999
490 IF ma(h,i,1)<1 THEN ma(h,i,1)=1
500 IF ma(h,i,2)<0 THEN ma(h,i,2)=ma(h,i
,2)+1:GOTO 550
510 IF RND(1)>0.98 THEN ma(h,i,2)=ma(h,i
,2)-(RND(1)*(RND(0))*10):GOTO 530
520 ma(h,i,2)=ma(h,i,2)-RND(1)+RND(1)
530 IF ma(h,i,2)>5 THEN ma(h,i,2)=5
540 IF ma(h,i,3)>99 THEN ma(h,i,3)=ma(h,
i,3)-100
550 NEXT i
560 ' ** Gelbe Karten **
570 FOR i=1 TO 11
580 IF RND(1)>0.9 THEN ma(h,el(i,h),3)=m
a(h,el(i,h),3)+1:IF ma(h,el(i,h),3)=4 OR
 ma(h,el(i,h),3)=7 OR ma(h,el(i,h),3)=10
 THEN ma(h,el(i,h),3)=ma(h,el(i,h),3)+10
0
590 NEXT
600 ef(h)=0:FOR i=1 TO 11
610 IF ma(h,el(1,h),2)<0 OR ma(h,el(i,h)
,3)>99 THEN 630
620 ef(h)=ef(h)+INT(ma(h,el(i,h),1))
630 NEXT
640 IF ef(h)>=45 THEN a=5:GOTO 690
650 IF ef(h)>=36 THEN a=4:GOTO 690
660 IF ef(h)>=27 THEN a=3:GOTO 690
670 IF ef(h)>=18 THEN a=2:GOTO 690
680 a=1
690  x$(sp(h))=STR$(a)+MID$(x$(sp(h)),3)
700 RETURN
710 ' *** Gehaelter ***
720 IF (y+2)/4<>INT((y+2)/4) THEN RETURN
730 IF qq=1 THEN RETURN
740 IF dm(h)<0 THEN LOCATE 19,4: zi=INT(
dm(h)*0.02):dm(h)=dm(h)-zi:a$="Schuldzin
sen   :"+STR$(zi):PRINT CHR$(24);LEFT$(a$
+"            ",25);CHR$(24);
750 gh(h)=0:FOR j=1 TO 20:gh(h)=gh(h)+ma
(h,j,0):NEXT j
760 LOCATE 19,5:a$="Gehaltszahlung:"+STR
$(gh(h)):PRINT CHR$(24);LEFT$(a$+"
   ",25):dm(h)=dm(h)-gh(h)
770 LOCATE 19,6:a$="Kapital          :"+STR
$(dm(h)):PRINT LEFT$(a$+"           ",25);C
HR$(24)
780 qq=1
790 RETURN
800 ' *** Mannschaft ***
810 'ef(h)=0:FOR j=1 TO 11:ef(h)=ef(h)+e
l(j,h):NEXT
820 CLS:PRINT CHR$(24);"  SPIELERFORM/MA
NNSCHAFTSAUFSTELLUNG";CHR$(18):CHR$(24):
PRINT
830 FOR i=1 TO 20:PRINT i;TAB(5);:GOSUB
4770:PRINT g$;TAB(10);INT(ma(h,i,1)*100)
/100
840 IF ma(h,i,2)<0 THEN LOCATE 15,i+2:PR
INT CHR$(24);"VERLETZT";CHR$(24):GOTO 86
0
850 IF ma(h,i,3)>99 THEN LOCATE 15,i+2:P
RINT CHR$(24);"GESPERRT";CHR$(24)
860 NEXT
870 LOCATE 23,3:PRINT"NR POS.  SP ST"
880 ef(h)=0
890 FOR j=1 TO 11:LOCATE 23,j+4:PRINT MI
D$(STR$(j),2);TAB(26);:i=el(j,h):GOSUB 4
770:PRINT q$;el(j,h);
900 LOCATE 10,i+2:PRINT CHR$(24);">";CHR
$(24)
910 IF ma(h,el(j,h),2)<0 OR ma(h,el(j,h)
,3)>99 THEN LOCATE 35,j+4:PRINT  CHR$(24
);"X";CHR$(24): GOTO 930
920 LOCATE 34,j+4:PRINT INT(ma(h,el(j,h)
,1)):ef(h)=ef(h)+INT(ma(h,el(j,h),1))
930 NEXT
940 LOCATE 23,17:PRINT "SPIELSTAERKE:";e
f(h):LOCATE 23,18:PRINT"STAERKEGRAD :";V
AL(LEFT$(x$(sp(h)),2))
950 GOSUB 420
960 LOCATE 23,20:PRINT"Soll Mannschaft":
LOCATE 23,21:PRINT"geaendert werden?"
970 q$=INKEY$:IF q$="" THEN 970 ELSE IF
```



```
q$<>"j" AND q$<>"J" THEN 110
980 PRINT CHR$(24);:FOR i=1 TO 11:LOCATE
 23,i+4:PRINT MID$(STR$(i),2,2):NEXT:PRI
NT CHR$(24);
990 LOCATE 23,21:PRINT CHR$(18):LOCATE 2
3,20:PRINT CHR$(18):LOCATE 23,20:INPUT "
Welche Nr.";q:IF q<1 OR q>11 THEN LOCATE
 23,20:PRINT CHR$(7);CHR$(24);"NUR 1 - 1
1 !!!";CHR$(24);CHR$(7);:FOR i=1 TO 400:
NEXT:GOTO 990
1000 IF ma(h,el(q,h),2)>=0 AND ma(h,el(q
,h),3)<100 THEN ef(h)=ef(h)-INT(ma(h,el(
q,h),1))
1010 v(el(q,h),h)=0:LOCATE 31,q+4:PRINT
CHR$(18);:LOCATE 10,el(q,h)+2:PRINT" "
1020 LOCATE 36,17: PRINT ef(h):FOR i= 1
TO 11: LOCATE 23,i+4:PRINT MID$(STR$(i),
2,2):NEXT:PRINT CHR$(24);:FOR i=1 TO 20:
 LOCATE 2,i+2:PRINT MID$(STR$(i),2):NEXT
:PRINT CHR$(24);
1030 LOCATE 23,21:INPUT "Neuer Spieler";
r: IF r<1 OR r>20 THEN LOCATE 23,21:PRIN
T CHR$(7);CHR$(24);"NUR 1 - 20 !!!";CHR$
(24);CHR$(7);:FOR i=1 TO 400:NEXT:GOTO 1
030
1040 LOCATE 23,21:PRINT CHR$(18);:ON q G
OTO 1050,1070,1070,1070,1090,1090,1090,1
090,1110,1110,1110
1050 IF ma(h,r,4)=1 THEN 1130
1060 GOTO 1120
1070 IF ma(h,r,4)=2 THEN 1130
1080 GOTO 1120
1090 IF ma(h,r,4)=3 THEN 1130
1100 GOTO 1120
1110 IF ma(h,r,4)=4 THEN 1130
1120 LOCATE 23,21:PRINT CHR$(7);CHR$(24)
:"FALSCHE POS.!!!";CHR$(24);CHR$(7):FOR
i= 1 TO 400: NEXT:GOTO 1030
1130 IF v(r,h)>0 THEN LOCATE 23,21:PRINT
 CHR$(7);CHR$(24);"BEREITS EINGES.";CHR$
(24);CHR$(7):FOR i= 1 TO 400: NEXT:GOTO
1030
1140 v(el(q,h),h)=0:v(r,h)=1:LOCATE 32,q
+4:PRINT MID$(STR$(r),2,2);TAB(34):: IF
ma(h,r,2)>=0 AND ma(h,r,3)<100 THEN PRIN
T INT(ma(h,r,1)):ef(h)=ef(h)+ INT(ma(h,r
,1))
1150 el(q,h)=r: LOCATE 10,r+2:PRINT CHR$
(24);">";CHR$(24)
1160 LOCATE 1,3:FOR i=1 TO 20:PRINT i:NE
XT
1170 GOSUB 600:LOCATE 36,17:PRINT ef(h):
LOCATE 36,18:PRINT a
1180 GOTO 960
1190 ' AUTO-Aufstellung
1200 el(1,h)=1:v(1,h)=1:FOR j=3 TO 5:el(
j-1,h)=j:v(j,h)=1:NEXT
1210 FOR j=7 TO 10:el(j-2,h)=j:v(j,h)=1:
NEXT:FOR j=12 TO 14:el(j-3,h)=j:v(j,h)=1
:NEXT
1220 GOSUB 600:RETURN
1230 ' *** Verkaufpruefung ***
1240 i=1
1250 IF t5(i) <>0 THEN 1270
1260 t5(i)=q:RETURN
1270 i=i+1: IF i<=5 THEN 1250
1280 IF k$="k" OR k$="K" THEN RETURN
1290 kk=255:RETURN
1300 ' *** Transfermarkt ***
1310 CLS:PRINT "NR POS   FORM GEHALT":PR
INT"-- ----- ---- ------"
1320 FOR i=1 TO 20: PRINT MID$(STR$(i),2
);TAB(4);:GOSUB 4770:PRINT q$;TAB(10);MI
D$(STR$(INT(ma(h,i,1)*100)/100),2);TAB(1
5);MID$(STR$(ma(h,i,0)),2):IF ma(h,i,2)<
0 THEN LOCATE 14,i+2:PRINT CHR$(24);"V";
CHR$(24)
1330 NEXT
1340 FOR i=1 TO 11:LOCATE 9,el(i,h)+2:PR
INT CHR$(24);">";CHR$(24):NEXT
1350 tr=tr+1:IF tr>3 THEN LOCATE 23,1:PR
INT CHR$(24);"TRANSFERLISTE     ":LOCATE
23,2:PRINT "GESCHLOSSEN!      ";CHR$(24):
LOCATE 23,3:PRINT "Bitte bestaetigen":CA
LL &BB06:GOTO 110
1360 LOCATE 23,1:PRINT CHR$(24);"< TRANS
FERLISTE > ":LOCATE 23,2:PRINT ;"-------
---------- "
1370 LOCATE 23,4:PRINT"  POS    ST DM
 "
1380 FOR i=1 TO 5
1390 IF c=1 THEN 1430
1400 ts(i)=INT(RND(1)*5+1)
1410 tk(i)=INT(pr(ts(i))+(RND(1)*ts(i)*2
0000)-(RND(1)*ts(i)*20000))
1420 ta(i)=INT(RND(1)*4+1)
1430 GOSUB 4790:LOCATE 23,i+5:PRINT MID$
(STR$(i),2);" ";q$;ts(i);:IF tk(i)<10000
0 THEN PRINT " ";:IF tk(i)=0 THEN PRINT"
   ";
1440 PRINT tk(i)
1450 NEXT:c=1
1460 PRINT CHR$(24);
1470 LOCATE 23,16:PRINT CHR$(24);"KAPITA
L:";MID$(STR$(dm(h)),1);"DM":PRINT CHR$(
24);:GOSUB 420
1480 LOCATE 23,13:PRINT"SPIELERKAUF,":LO
CATE 23,14:PRINT"SPIELERVERKAUF,":LOCATE
 23,15:PRINT "MENUE (K,V,M)?";:PEN 3:PRI
NT CHR$(24);"    ";CHR$(24);:PEN 1
1490 k$=INKEY$:IF k$="" THEN 1490 ELSE L
OCATE 37,15:PRINT CHR$(24);" "k$" ";CHR$
(24)
1500 FOR i=16 TO 23: LOCATE 23,i:PRINT C
HR$(18);:NEXT
1510 IF k$="k" OR k$="K" THEN 1540
1520 IF k$="v" OR k$="V" THEN 1680
1530 GOTO 110
1540 IF dm(h) < - 999999 THEN PEN 3:LOCA
TE 23,18:PRINT CHR$(24);"NICHT FINANZIER
BAR";CHR$(24):PEN 1:FOR i=1 TO 1000:NEXT
:GOTO 1830
1550 LOCATE 23,18:INPUT "NUMMER";l:IF l<
0 OR l>5 THEN PRINT CHR$(7);:GOTO 1550
1560 IF l=0 THEN 1830
1570 j=l:IF tk(l)=0 THEN LOCATE 23,19:PR
INT CHR$(24);"BEREITS VERKAUFT";CHR$(24)
:GOTO 1550
1580 IF ma(h,j,1)<>0 THEN 1660
1590 ma(h,j,1)=ts(l)+RND(1):ma(h,j,4)=ta
(l)
1600 ma(h,j,2)=5:dm(h)=dm(h)-tk(l):ma(h,
j,0)=INT(ma(h,j,1)*6000)
1610 q=j:GOSUB 1240
1620 LOCATE 4,j+2:i=j:GOSUB 4780:PRINT q
$:INT(ma(h,j,1)*100)/100:TAB(15);MID$(ST
R$(ma(h,j,0)),2)
1630 ts(l)=0:tk(l)=0:ta(l)=0
1640 LOCATE 23,1+5:PRINT CHR$(24);"
          ";CHR$(24)
1650 GOTO 1830
1660 j=j+1:IF j<=20 THEN 1580
```

Listing 2. »Manager« (Fortsetzung)

```
1670 LOCATE 23,19: PRINT CHR$(24):"PLAET
ZE BESETZT";CHR$(24):GOTO 1470
1680 LOCATE 23,16: INPUT"SPIELER-NR.";q:
 IF q<0 OR q>20 THEN PRINT CHR$(7)::GOTO
 1680
1690 IF q=0 THEN 1830
1700 FOR i= 16 TO 23:LOCATE 23,i:PRINT C
HR$(18)::NEXT
1710 IF ma(h,q,1)=0 THEN LOCATE 23,17:PR
INT CHR$(24):"NICHT VORHANDEN";CHR$(24):
GOTO 1680
1720 IF ma(h,q,2)<0 THEN LOCATE 23,17:PR
INT CHR$(24):"SPIELER VERLETZT";CHR$(24)
:GOTO 1680
1730 j=1
1740 IF t5(j)<>q THEN 1760
1750 kk=0: LOCATE 23,17:PRINT CHR$(24);"
WEITERE VERHAND-  ":LOCATE 23,18:PRINT"L
UNGEN NAECHSTE   ":LOCATE 23,19:PRINT"
W O C H E !!     ";CHR$(24);:GOTO 1470
1760 j=j+1:IF j<=5 THEN 1740
1770 GOSUB 1240
1780 r=INT(ma(h,q,1)):vv=INT(pr(r)+(RND(
1)*ma(h,q,1)*20000)-(RND(1)*ma(h,q,1)*20
000))
1790 LOCATE 23,18:PRINT"ANGEBOT:";CHR$(2
4);MID$(STR$(vv),2);" DM";CHR$(24)
1800 LOCATE 23,20:PRINT"VERKAUFEN SIE?":
a$=INKEY$:IF a$="" THEN 1800 ELSE IF a$<
>"j" AND a$<>"J" THEN 1830
1810 FOR i=0 TO 4:ma(h,q,i)=0:NEXT i:GOS
UB 600
1820 LOCATE 3,q+2:PRINT"
    ":dm(h)=dm(h)+vv
1830 FOR i=16 TO 23: LOCATE 23,i:PRINT C
HR$(18)::NEXT
1840 GOTO 1470
1850 STOP
1860 ' *** Spielergebnisse ***
1870 CLS
1880 a$=STR$(y)+". Bundesligaspieltag":P
RINT A$:PRINT
1890 FOR j=1 TO 9
1900 IF y>17 THEN z=y-17:z(1)=VAL(MID$(m
$(z),(j-1)*4+3,2)):z(2)=VAL(MID$(m$(z),(
j-1)*4+1,2)):GOTO 1920
1910 z(1)=VAL(MID$(m$(y),(j-1)*4+1,2)):z
(2)=VAL(MID$(m$(y),(j-1)*4+3,2))
1920 FOR k=1 TO sp:IF z(1)=sp(k) THEN tb
(k)=VAL(LEFT$(x$(z(2)),2))
1930 IF z(2)=sp(k) THEN tb(k)=-1
1940 NEXT
1950 PRINT MID$(x$(z(1)),3);TAB(17);": "
;MID$(x$(z(2)),3);:IF es=7 THEN PRINT ""
:GOTO 2130
1960 wa=INT(RND(1)*6+1):wb=INT(RND(1)*6+
1):wc=wa+wb
1970 s1(1)=VAL(LEFT$(x$(z(1)),2))
1980 s1(2)=VAL(LEFT$(x$(z(2)),2))
1990 s=INT(RND(1)*5):n=INT(RND(1)*4):IF
s<=n THEN 1990
2000 wf=VAL(MID$(h$(s1(1),s1(2)),wc+1,1)
)
2010 PRINT TAB(36);
2020 ON wf+1 GOTO 2030,2070,2090
2030 n(z(1))=n(z(1))+1:np(z(1))=np(z(1))
+2
2040 s(z(2))=s(z(2))+1:pp(z(2))=pp(z(2))
+2
2050 pt(z(1))=pt(z(1))+n:nt(z(1))=nt(z(1
))+s:pt(z(2))=pt(z(2))+s:nt(z(2))=nt(z(2
))+n
2060 xx$=MID$(STR$(n),2)+":"+MID$(STR$(s
),2):PRINT xx$:GOTO 2130
2070 u=INT(RND(1)*3):FOR k=1 TO 2:u(z(k)
)=u(z(k))+1:pp(z(k))=pp(z(k))+1:np(z(k))
=np(z(k))+1:pt(z(k))=pt(z(k))+u:nt(z(k))
=nt(z(k))+u:NEXT
2080 xx$=MID$(STR$(u),2)+":"+MID$(STR$(u
),2):PRINT xx$:GOTO 2130
2090 s(z(1))=s(z(1))+1:pp(z(1))=pp(z(1))
+2
2100 n(z(2))=n(z(2))+1:np(z(2))=np(z(2))
+2
2110 pt(z(1))=pt(z(1))+s:nt(z(1))=nt(z(1
))+n:pt(z(2))=pt(z(2))+n:nt(z(2))=nt(z(2
))+s
2120 xx$=MID$(STR$(s),2)+":"+MID$(STR$(n
),2):PRINT xx$:GOTO 2130
2130 NEXT j
2140 IF es=7 THEN LOCATE 19,24:PRINT"Tas
te druecken":IF INKEY$="" THEN 2140 ELSE
 GOTO 110
2150 FOR i=1 TO sp:PRINT
2160 PRINT CHR$(24);MID$(x$(sp(i)),3);CH
R$(24);:IF tb(i)<0 THEN 2210
2170 IF y=1 THEN pl(i)=10
2180 zu=INT(((22-pl(i))*600*tb(i))+(RND(
1)*2000*tb(i))-(RND(1)*2000*tb(i)))
2190 ne=INT(zu*9+(RND(1)*5000*tb(i))-(RN
D(1)*5000*tb(i))):az(h)=az(h)+1
2200 PRINT TAB(17);"Zuschauer:";TAB(31);
zu:PRINT TAB(17);"Nettoeinnahme:";ne:PRI
NT TAB(17);"Kapital:";:dm(i)=dm(i)+ne:zu
(i)=zu(i)+zu:az(i)=az(i)+1:GOTO 2220
2210 PRINT TAB(17);"Reisekosten:";:q=ef(
i)*605:dm(i)=dm(i)-q:PRINT TAB(31);-q:PR
INT TAB(17);"Kapital:";
2220 PRINT TAB(31):dm(i)
2230 NEXT i
2240 LOCATE 1,24:PRINT CHR$(24);">>>>>>>
>TABELLE WIRD ERMITTELT<<<<<<<<";CHR$(24
);
2250 GOSUB 2300
2260 PRINT CHR$(17):CHR$(13):"Tabelle er
rechnet / Bitte Taste druecken";
2270 IF INKEY$="" THEN 2270
2280 IF y<>34 THEN GOSUB 2490
2290 RETURN
2300 ' *** Tabelle sortiert ***
2310 FOR i=1 TO 18 : pa(i)=pp(i):va(i)=i
:NEXT
2320 '
2330 m=18
2340 m=INT(m/2)
2350 IF m=0 THEN RETURN
2360 j=1:k=18-m
2370 i=j
2380 l=i+m
2390 IF pa(i)>pa(l) THEN 2460
2400 IF pa(i)=pa(l) THEN 2720
2410 t1=pa(i):pa(i)=pa(l):pa(l)=t1
2420 x1=va(i):va(i)=va(l):va(l)=x1
2430 i=i-m
2440 IF i<1 THEN 2460
2450 GOTO 2380
2460 j=j+1
2470 IF j>k THEN 2340
2480 GOTO 2370
2490 CLS
2500 PRINT"   Spieltag";:IF es=3 THEN PR
INT y-1;:GOTO 2520
2510 PRINT y;
2520 PRINT TAB(20);" S  U  N Pkte. Tore"
```



```
:PRINT
2530 FOR i=1 TO 18:IF i<10 THEN a$=" "+M
ID$(STR$(i),2)+" "+MID$(x$(va(i)),3) ELS
E a$=MID$(STR$(i),2)+" "+MID$(x$(va(i)),
3)
2540 a$=a$+SPACE$(19-LEN(a$))
2550 IF es=3 THEN 2580
2560 FOR k=1 TO sp: IF va(i)=sp(k) THEN
pl(k)=i
2570 NEXT
2580 IF s(va(i))<10 THEN a$=a$+" "+MID$(
STR$(s(va(i))),2) ELSE a$=a$+MID$(STR$(s
(va(i))),2)
2590 IF u(va(i))<10 THEN a$=a$+"  "+MID$
(STR$(u(va(i))),2) ELSE a$=a$+" "+MID$(S
TR$(u(va(i))),2)
2600 IF n(va(i))<10 THEN a$=a$+"  "+MID$
(STR$(n(va(i))),2) ELSE a$=a$+" "+MID$(S
TR$(n(va(i))),2)
2610 IF pp(va(i))<10 THEN a$=a$+"  "+MID
$(STR$(pp(va(i))),2)+":" ELSE a$=a$+" "+
MID$(STR$(pp(va(i))),2)+":"
2620 IF np(va(i))<10 THEN a$=a$+MID$(STR
$(np(va(i))),2)+" " ELSE a$=a$+MID$(STR$
(np(va(i))),2)
2630 IF pt(va(i))<10 THEN a$=a$+"  "+MID
$(STR$(pt(va(i))),2)+":" ELSE IF pt(va(i
))<100 THEN a$=a$+" "+MID$(STR$(pt(va(i)
)),2)+":" ELSE a$=a$+MID$(STR$(pt(va(i))
),2)+":"
2640 a$=a$+MID$(STR$(nt(va(i))),2)
2650 PRINT a$:NEXT
2660 PRINT
2670 PRINT TAB(21);"TASTE DRUECKEN "
2680 IF INKEY$="" THEN 2680
2690 IF es=3 THEN 110
2700 RETURN
2710 ' *** Tordifferenz - Sort ***
2720 IF pt(va(i))-nt(va(i)) > pt(va(1))-
nt(va(1)) THEN 2460
2730 IF pt(va(i))-nt(va(i)) < pt(va(1))-
nt(va(1)) OR ((pt(va(i))-nt(va(i))= pt(v
a(1))-nt(va(1)) AND pt(va(i))<pt(va(1)))
 THEN 2410
2740 GOTO 2460
2750 '************
2760 CLS:PRINT CHR$(24):"     GEHAELTER/
VERLETZTE/GELBE KARTEN":CHR$(24)
2770 J=0:FOR I=1 TO 20
2780 PRINT I:TAB(6);MA(H,I,0)*9;:J=J+MA(
H,I,0)*9
2790 PRINT TAB(16);INT(MA(H,I,2)*100)/10
0;TAB(29);MA(H,I,3)
2800 NEXT
2810 PRINT TAB(6);j:PRINT:PRINT TAB(16):
"TASTE DRUECKEN  [";:PEN 3:PRINT CHR$(24
);" ";CHR$(24);:PEN 1:PRINT"]":CALL &BB0
6:GOTO 110
2820 ' *** UEFA-/DFB-Pokal ***
2830 eu=0:fu=0:l=0:ee=0
2840 IF y=4 OR y=6 OR y=10 OR y=12 OR y=
17 OR y=19 OR y=22 OR y=24 OR y=26 OR y=
28 OR y=31 OR y=33 THEN 2870
2850 IF y=3 OR y=9 OR y=15 OR y=21 OR y=
25 OR y=32 THEN 3340
2860 RETURN
2870 ' *** UEFA-CUP ***
2880 CLS:GOSUB 5790:t3=0:LOCATE 1,21

2890 FOR i=1 TO sp:IF LEFT$(up$(i),4)="A
US" THEN PRINT up$(i);" :";TAB(24);MID$(
x$(sp(i)),3):GOTO 2910
2900 PRINT up$(i);" :";TAB(24);MID$(x$(s
p(i)),3):t3=t3+1
2910 NEXT
2920 LOCATE 12,24:PRINT "TASTE DRUECKEN
 [";:PEN 3:PRINT CHR$(24);" ";CHR$(24);:
PEN 1:PRINT"]";:CALL &BB06:MODE 1:PEN 1:
IF t3=0 THEN RETURN
2930 FOR k=1 TO sp:CLS:n=0:IF LEFT$(up$(
k),3)="AUS" THEN 3330
2940 IF qu(k)=1 THEN qu(k)=2:PRINT "AUSW
AERTSSPIEL":LOCATE 23,18:PRINT "HINSPIEL
:   ";MID$(STR$(JU(K,2)),2);":";MID$(STR
$(JU(K,1)),2):GOTO 2980
2950 IF qu(k)=2 THEN qu(k)=1:PRINT"HEIMS
PIEL":LOCATE 23,18:PRINT"HINSPIEL:   ";M
ID$(STR$(JU(K,3)),2);":";MID$(STR$(JU(K,
4)),2):GOTO 2980
2960 IF RND(1)>0.5 THEN PRINT"HEIMSPIEL"
:qu(k)=1:GOTO 2980
2970 qu(k)=2:PRINT"AUSWAERTSSPIEL"
2980 gu=qu(k):bu=6+RND(1):cu=4-RND(1)*2
2990 IF gu=2 THEN 3070
3000 zu=INT((12000*cu)+(RND(1)*2000*cu)-
(RND(1)*2000*cu)+5000)
3010 IF up$(k)="IM HALBFINALE" THEN zu=z
u*1.65:GOTO 3040
3020 IF up$(k)="IM FINALE" THEN zu=zu*2.
1:GOTO 3040
3030 zu=INT(zu/3*VAL(MID$(up$(k),8,1)))
3040 ne=INT(zu*9+(RND(1)*5000*cu)-(RND(1
)*5000*cu)):dm(k)=dm(k)+ne
3050 LOCATE 23,10:PRINT "ZUSCHAUER:";zu:
LOCATE 23,11:PRINT"EINNAHME: ";ne:LOCATE
 23,12:PRINT"KAPITAL:";
3060 GOTO 3080
3070 LOCATE 23,10:PRINT "KOSTEN:  ";:q=e
f(k)*905:dm(k)=dm(k)-q:PRINT -q:LOCATE 2
3,11:PRINT"KAPITAL:";
3080 PRINT dm(k)
3090 LOCATE 23,1:PRINT MID$(x$(sp(k)),3,
LEN(x$(sp(k)))-1)
3100 GOSUB 3830
3110 IF qu(k)=1 THEN ju(k,1)=eu:ju(k,2)=
fu:GOTO 3130
3120 IF qu(k)=2 THEN ju(k,3)=eu:ju(k,4)=
fu
3130 ju(k,0)=ju(k,0)+1:IF ju(k,0)<2 THEN
 3320
3140 ' *** Abrechnung UEFA-CUP ***
3150 LOCATE 23,19:PRINT"RUECKSPIEL: ";:G
OSUB 4000:PRINT
3160 IF iu(k,1)+ju(k,3)>ju(k,2)+ju(k,4)
THEN 3240
3170 IF ju(k,1)+ju(k,3)=ju(k,2)+ju(k,4)
AND ju(k,3)>ju(k,2) THEN 3240
3180 IF ju(k,1)+ju(k,3)=ju(k,2)+ju(k,4)
AND ju(k,3)=ju(k,2) THEN au=5: IF l<>1 T
HEN i=91:j=120: PRINT"VERLAENGERUNG! ";:
CALL &BB06:GOSUB 3870:l=1:GOTO 3160
3190 IF l=0 THEN PRINT CHR$(24);"LEIDER
AUSGESCHIEDEN ";up$(k);CHR$(24);:up$(k)=
"AUS "+up$(k):GOTO 3310
3200 IF l=1 THEN GOSUB 3710
3210 IF eu>fu THEN 3240
3220 IF fu>eu THEN l=0: GOTO 3190
3230 GOSUB 3780:GOTO 3210
3240 IF up$(k)="IN DER 1.RUNDE" THEN up$
(k)="IN DER 2.RUNDE":up(k)=5:GOTO 3300
3250 IF up$(k)="IN DER 2.RUNDE" THEN up$
(k)="IN DER 3.RUNDE":up(k)=8:GOTO 3300
```

Listing 2. »Manager« (Fortsetzung)

```
3260 IF up$(k)="IN DER 3.RUNDE" THEN up$
(k)="IN DER 4.RUNDE":up(k)=12:GOTO 3300

3270 IF up$(k)="IN DER 4.RUNDE" THEN up$
(k)="IM HALBFINALE":up(k)=16:GOTO 3300
3280 IF up$(k)="IM HALBFINALE" THEN up$(
k)="IM FINALE":up(k)=20:GOTO 3300
3290 IF up$(k)="IM FINALE" THEN up$(k)="
UEFA-POKAL-SIEGER":up(k)=25
3300 PRINT CHR$(24);"GRATULATION!SIE SIN
D ";up$(k);"!";CHR$(24)
3310 gu(k)=0:ju(k,0)=0
3320 CALL &BB06
3330 NEXT k:RETURN
3340 ' *** DFB-Pokal ***
3350 CLS:GOSUB 5740:t3=0:LOCATE 1,21

3360 FOR i=1 TO sp:IF LEFT$(dp$(i),3)="A
US" THEN PRINT dp$(i);TAB(20);MID$(x$(sp
(i)),3,LEN(x$(sp(i)))-1):GOTO 3380
3370 PRINT dp$(i);TAB(20);MID$(x$(sp(i))
,3):t3=t3+1
3380 NEXT:LOCATE 12,24:PRINT "TASTE DRUE
CKEN  [";:PEN 3:PRINT CHR$(24);" ";CHR$(
24);:PEN 1:PRINT"]":CALL &BB06:MODE 1:PE
N 1:IF t3=0 THEN RETURN
3390 FOR k=1 TO sp:CLS:n=0:IF LEFT$(dp$(
k),4)="AUS" THEN 3700
3400 IF RND(1)>0.5 THEN PRINT"HEIMSPIEL"
:gu=1:GOTO 3420
3410 PRINT"AUSWAERTSSPIEL":;gu=2
3420 PRINT TAB(23);MID$(x$(sp(k)),3,LEN(
x$(sp(k)))-1)
3430 bu=5.8+RND(1):cu=3.8-RND(1)*2
3440 zu=INT((10000*cu)+(RND(1)*2000*cu)-
(RND(1)*2000*cu)+5000)
3450 IF dp$(k)="IM HALBFINALE" THEN zu=z
u*2:GOTO 3480
3460 IF dp$(k)="IM FINALE" THEN zu=zu*3:
GOTO 3480
3470 zu=INT(zu/3.5*VAL(MID$(dp$(k),8,1))
)
3480 ne=INT((zu*9+(RND(1)*5000*cu)-(RND(
1)*5000*cu))/2):dm(k)=dm(k)+ne
3490 LOCATE 23,10:PRINT "ZUSCHAUER:":zu:
LOCATE 23,11:PRINT"EINNAHME: ":ne:LOCATE
 23,12:PRINT"KAPITAL:";dm(k)
3500 GOSUB 3830
3510 IF eu>fu THEN 3620
3520 IF eu<fu THEN l=0:GOTO 3540
3530 IF eu=fu THEN au=5:IF l<>1 THEN PRI
NT"VERLAENGERUNG!":;CALL &BB06:i=91:j=12
0:GOSUB 3870:l=1:GOTO 3510
3540 IF l=0 THEN PRINT CHR$(24):"LEIDER
VERLOREN ":dp$(k):CHR$(24);:dp$(k)="AUS
"+dp$(k):GOTO 3690
3550 IF l=1 AND ee=0 THEN 3600
3560 IF l=1 THEN GOSUB 3710
3570 IF eu>fu THEN 3620
3580 IF fu>eu THEN l=0:GOTO 3540
3590 GOSUB 3780:GOTO 3570
3600 IF l=1 THEN l=0:ee=1:PRINT"WIEDERHO
LUNGSSPIEL":;n=0:eu=0:fu=0:l=0:CALL &BB0
6:CLS:IF gu=1 THEN gu=2:PRINT"AUSWAERTSS
PIEL":;GOTO 3420
3610 gu=1:PRINT"HEIMSPIEL":;GOTO 3420
3620 IF dp$(k)="IN DER 1.RUNDE" THEN dp$
(k)="IN DER 2.RUNDE":dp(k)=3:GOTO 3680
3630 IF dp$(k)="IN DER 2.RUNDE" THEN dp$
(k)="IN DER 3.RUNDE":dp(k)=6:GOTO 3680
3640 IF dp$(k)="IN DER 3.RUNDE" THEN dp$
(k)="IN DER 4.RUNDE":dp(k)=9:GOTO 3680
3650 IF dp$(k)="IN DER 4.RUNDE" THEN dp$
(k)="IM HALBFINALE":dp(k)=12:GOTO 3680
3660 IF dp$(k)="IM HALBFINALE" THEN dp$(
k)="IM FINALE":dp(k)=16:GOTO 3680
3670 IF dp$(k)="IM FINALE" THEN dp$(k)="
DFB-POKAL-SIEGER":dp(k)=20
3680 PRINT CHR$(24);"GRATULATION!SIE SIN
D ";dp$(k);"!";CHR$(24)
3690 CALL &BB06
3700 NEXT:RETURN
3710 ' *** Elfmeterschiessen ***
3720 PRINT"ELFMETERSCHIESSEN!  ":;CALL &
BB06:l=2:eu=0:fu=0:CLS
3730 FOR i=1 TO 5:FOR i=1 TO 2
3740 IF i=1 THEN IF RND(1)<0.85 THEN eu=
eu+1:GOSUB 4000:GOTO 3770
3750 IF i=1 THEN 3770
3760 IF RND(1)<0.85 THEN fu=fu+1:GOSUB 4
000
3770 CALL &BB06:NEXT:NEXT:RETURN
3780 FOR i=1 TO 2:IF i=1 THEN IF RND(1)<
0.85 THEN eu=eu+1:GOSUB 4000:GOTO 3820
3790 IF i=1 THEN 3820
3800 IF RND(1)<0.85 THEN fu=fu+1:GOSUB 4
000
3810 CALL &BB06
3820 NEXT:RETURN
3830 ' *** Pokalergebnisse ***
3840 au=5:fu=0:eu=0:LOCATE 23,4:PRINT"SP
IELZEIT:    MIN."
3850 qc=30/(bu-cu):oc=au
3860 i=1:j=90
3870 au=au+RND(1)-RND(1)
3880 nc=(au-cu)*qc+5:IF nc<8 THEN nc=8
3890 IF nc>33 THEN nc=33
3900 LOCATE 1,16:PRINT"  - >":TAB(36):"<
 + "
3910 LOCATE oc,16:PRINT" ":LOCATE INT(nc
),16:PRINT CHR$(231):oc=INT(nc):FOR xx=1
 TO 250:NEXT
3920 LOCATE 34,4:PRINT MID$(STR$(i),2)
3930 IF i<10 THEN xx$="  " ELSE IF i<100
 THEN xx$=" "
3940 IF au>bu THEN LOCATE 1,n+3:PRINT xx
$+MID$(STR$(i),2);".MINUTE   ":;eu=eu+1:
GOSUB 4000:n=n+1:au=5:oc=nc:GOTO 3960
3950 IF au<cu THEN LOCATE 1,n+3:PRINT xx
$+MID$(STR$(i),2);".MINUTE   ":;fu=fu+1:
GOSUB 4000:n=n+1:au=5:oc=nc
3960 IF i=45 THEN LOCATE 1,18:PRINT"HALB
ZEITSTAND: ":;GOSUB 4000:au=5:GOTO 3980
3970 IF i=105 THEN LOCATE 1,18:PRINT"HAL
BZEITSTAND DER VERLAENGERUNG:  ":;GOSUB
4000:au=5
3980 i=i+1:IF i<=j THEN 3870
3990 LOCATE 1,19:PRINT"ENDERGEBNIS:   ":
:GOSUB 4000:PRINT:RETURN
4000 au=5
4010 IF gu=1 THEN PRINT MID$(STR$(eu),2)
;":"MID$(STR$(fu),2):RETURN
4020 IF gu=2 THEN PRINT MID$(STR$(fu),2)
;":"MID$(STR$(eu),2):RETURN
4030 ' *** Vereinsuebersicht ***
4040 IF f=2 THEN 4070
4050 IF y=1 THEN vv(h)=sk(h):GOTO 4070
4060 vv(h)=0:FOR i=1 TO 20:r=INT(ma(h,i,
1)):vv=INT(pr(r)+(RND(1)*ma(h,i,1)*20000
)-(RND(1)*ma(h,i,1)*20000)):vv(h)=vv(h)+
vv:NEXT:vv(h)=vv(h)+dm(h):f=2
4070 CLS:PRINT CHR$(24);:a$="Vereinsuebe
rsicht - "+MID$(x$(sp(h)),3):GOSUB 4230:
PRINT CHR$(24):
```

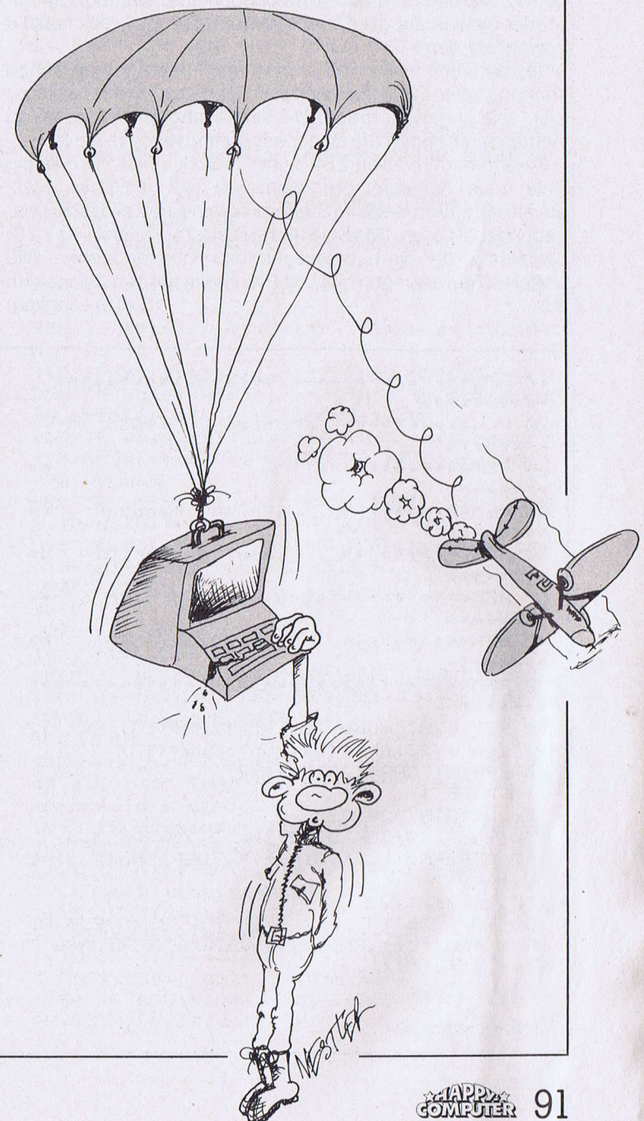


```
4080 LOCATE 1,5
4090 PRINT "   STARTVERMOEGEN  :";TAB(22
);sk(h)
4100 PRINT "   VEREINSVERMOEGEN:";TAB(22
);vv(h):PRINT
4110 PRINT "   SPIELTAG        :";TAB(22
);v
4120 PRINT "   TABELLENPLATZ   :";TAB(22
);pl(h)
4130 PRINT "   DFB-POKAL       :";TAB(23
);dp$(h)
4140 PRINT "   UEFA-CUP        :";TAB(23
);up$(h)
4150 PRINT
4160 PRINT "   GES.-ZUSCHAUER  :";TAB(22
);zu(h)
4170 PRINT "   HEIMSPIELE      :";TAB(22
);az(h)
4180 PRINT "   ZUSCHAUERSCHNITT:";TAB(22
);:IF az(h)=0 THEN PRINT" 0":GOTO 4200
4190 PRINT INT(zu(h)/az(h)+0.5)
4200 PRINT "   GEHAELTER       :";TAB(22
);gh(h)
4210 LOCATE 22,24:PRINT"Taste druecken
[";:PEN 3:PRINT CHR$(24);" ";CHR$(24);:P
EN 1:PRINT"]": CALL &BB06
4220 GOTO 110
4230 PRINT TAB(20-LEN(a$)/2):a$;:RETURN
4240 ' *** Abschlussuebersicht ***
4250 CLS:PRINT CHR$(24):"<<<<ABSCHLUSSTA
BELLE DER  BUNDESLIGA>>>>":CHR$(24);:PRI
NT:GOSUB 2500
4260 FOR h=1 TO sp
4270 CLS:PRINT CHR$(24);:a$="ABSCHLUSSUE
BERSICHT -"+MID$(X$(SP(H)),3):a$=a$+SPAC
E$(40-LEN(a$)):PRINT a$:PRINT CHR$(24):

4280 vv(h)=0:FOR i=1 TO 20:r=INT(ma(h,i,
1)):vv=INT(pr(r)+(RND(1)*ma(h,i,1)*20000
)-(RND(1)*ma(h,i,1)*20000)):vv(h)=vv(h)+
vv
4290 PRINT MID$(STR$(i),2);TAB(4):vv:NEX
T:PRINT TAB(4):"-------":PRINT TAB(3):vv
(h):
4300 LOCATE 12,3:PRINT" MANNSCHAFTSWERT:
":TAB(32):vv(h)
4310 LOCATE 12,4:PRINT " BARKAPITAL
:";TAB(32):dm(h)
4320 LOCATE 12,5:PRINT " ENDVERMOEGEN
:";TAB(32):vv(h)+dm(h)
4330 LOCATE 12,6:PRINT " STARTVERMOEGEN
:";TAB(32):sk(h)
4340 LOCATE 12,7:PRINT " GEWINN/VERLUST
:";TAB(32)::vv(h)=vv(h)+dm(h):PRINT INT(
vv(h)-sk(h))
4350 LOCATE 12,8:PRINT " TABELLENPLATZ
:";TAB(32);pl(h)
4360 LOCATE 12,9:PRINT" PUNKTE        :
";TAB(33);MID$(STR$(pp(sp(h))),2);":";MI
D$(STR$(np(sp(h))),2)
4370 LOCATE 12,10:PRINT" TORE
:";TAB(33):MID$(STR$(pt(sp(h))),2);":";M
ID$(STR$(nt(sp(h))),2)
4380 LOCATE 12,12:PRINT" DFB-POKAL:";LOC
ATE 13,13:PRINT dp$(h)
4390 LOCATE 12,15:PRINT" UEFA-CUP :";LOC
ATE 13,16:PRINT up$(h)
4400 LOCATE 12,18:PRINT" BUNDESLIGAPUNKT
E:";TAB(32):
4410 IF pl(h)<4 THEN q1=60+(4-pl(h))*5:G
OTO 4430
4420 q1=(19-pl(h))*4
4430 PRINT q1
4440 LOCATE 12,19:PRINT" VERMOEGENSPUNKT
E:";TAB(32);
4450 q2=INT((vv(h)-sk(h))/150000+0.5):PR
INT q2
4460 LOCATE 12,20:PRINT" UEFA-CUP-PUNKTE
 :";TAB(32);up(h)
4470 LOCATE 12,21:PRINT" DFB-POKAL-PUNKT
E:";TAB(32);dp(h)
4480 LOCATE 12,22:PRINT" TOR-PUNKTE
 :";TAB(32):
4490 q3=INT(pt(sp(h))/10+0.5):IF pt(sp(h
))>nt(sp(h)) THEN q3=q3+INT((pt(sp(h))-n
t(sp(h)))/10+0.5)
4500 PRINT q3
4510 IF o<>1 THEN LOCATE 12,23:PRINT" --
--------------------"
4520 q5(h)=q1+q2+q3+dp(h)+up(h)
4530 LOCATE 12,24:PRINT" GESAMT
 :";TAB(32):q5(h):
4540 IF o<>1 THEN PRINT TAB(37):"[";:PEN
 3:PRINT CHR$(24):" ";CHR$(24);:PEN 1:PR
INT"]":;CALL &BB06:GOTO 4560
4550 RETURN
4560 NEXT
4570 IF sp=1 THEN 370
4580 CLS:PRINT CHR$(24):TAB(19):"GESAMTU
EBERSICHT      ";CHR$(24);:PRINT
4590 o=1
4600 FOR h=1 TO sp
4610 GOSUB 4400
4620 PRINT MID$(x$(sp(h),3)):TAB(12):"->
"
4630 PRINT:NEXT:PRINT TAB(12):"Taste dru
ecken  [";:PEN 3:PRINT CHR$(24);" ";CHR$
(24);:PEN 1:PRINT "]":;CALL &BB06:GOTO 3
70
4640 '
4650 CLS
4660 PRINT CHR$(24):SPACE$(40):CHR$(24):
PRINT:PRINT
4670 FOR i=5 TO 1 STEP -1:FOR j=1 TO 18
4680 IF VAL(LEFT$(x$(j),2))=i THEN PRINT
 i:TAB(4);MID$(x$(j),3)
4690 NEXT:NEXT
4700 PRINT:PRINT CHR$(24):SPACE$(40):CHR
$(24):
4710 LOCATE 21,3:PRINT"ABSTIEGSKANDIDAT:
":LOCATE 23,4:PRINT"STAERKE : 1":LOCATE
23,5:PRINT"11 - 17 PUNKTE"
4720 LOCATE 21,7:PRINT"UNTERES MITTELFEL
D:":LOCATE 23,8:PRINT"STAERKE : 2":LOCAT
E 23,9:PRINT"18 - 26 PUNKTE"
4730 LOCATE 21,11:PRINT"MITTELFELD:":LOC
ATE 23,12:PRINT"STAERKE : 3":LOCATE 23,1
3:PRINT"27 - 35 PUNKTE"
4740 LOCATE 21,15:PRINT"UEFA-CUP-ANWAERT
ER:":LOCATE 23,16:PRINT"STAERKE : 4":LOC
ATE 23,17:PRINT"36 - 44 PUNKTE"
4750 LOCATE 21,19:PRINT"SPITZENMANNSCHAF
T:":LOCATE 23,20:PRINT"STAERKE : 5":LOCA
TE 23,21:PRINT"45 - 55 PUNKTE"
4760 LOCATE 21,24:PRINT"TASTE DRUECKEN
[";:PEN 3:PRINT CHR$(24);" ";CHR$(24);:P
EN 1:PRINT"]":;CALL &BB06:GOTO 110
4770 ' *** ma-Art ***
4780 ON ma(h,i,4)+1 GOTO 4800,4810,4820,
4830,4840
4790 ON ta(i) GOTO 4810,4820,4830,4840
```

Listing 2. »Manager« (Fortsetzung)

```
4800 q$="      ":RETURN
4810 q$="Torw.":RETURN
4820 q$="Vert.":RETURN
4830 q$="Mitt.":RETURN
4840 q$="Angr.":RETURN
4850 ' *** Mannschaften ***
4860 DATA BAYERN MUENCHEN,HAMBURGER SV,V
FB STUTTGART,1. FC K'LAUTERN,EINTR.FRANK
FURT
4870 DATA BOR.M'GLADBACH,BOR.DORTMUND,1.
FC KOELN,BAY. UERDINGEN,HANNOVER 96
4880 DATA BAYER LEVERKUSEN,SVW MANNHEIM,
FORT.D'DORF
4890 DATA VFL BOCHUM,SAARBRUECKEN,WERDER
 BREMEN
4900 DATA 1.FC NUERNBERG,SCHALKE 04
4910 ' *** Spielpaarungen ***
4920 DATA 011115180616031312100217050408
070914
4930 DATA 140804021712100313061615180111
090705
4940 DATA 011615130610031712040207051411
180908
4950 DATA 140204031706101513011611180908
050712
4960 DATA 151706040307121402081816111309
050110
4970 DATA 140304151701101113181609050208
120706
4980 DATA 010415070614030812051613181011
170902
4990 DATA 141504111718101602120503080607
011309
5000 DATA 011415080605030213101617180411
070912
5010 DATA 141104161713100912030206051508
010718
5020 DATA 010515020612101713041607181411
080903
5030 DATA 141604101709030612150201051108
180713
5040 DATA 011215031704100713141608180511
020906
5050 DATA 141004090615030112110218051608
130717
5060 DATA 010615091714100813051602181211
030704
5070 DATA 140415010611031812160213051008
170907
5080 DATA 010904081705100213121603180611
150714
5090 ' ** Ergebnisse **
5100 DATA 00011222222
5110 DATA 00001122222
5120 DATA 00000112222
5130 DATA 00000111222
5140 DATA 00000011222
5150 DATA 00112222222
5160 DATA 00011222222
5170 DATA 00001122222
5180 DATA 00000112222
5190 DATA 00000111222
5200 DATA 00122222222
5210 DATA 00112222222
5220 DATA 00011222222
5230 DATA 00001122222
5240 DATA 00000112222
5250 DATA 01122222222
5260 DATA 00112222222
5270 DATA 00111222222
5280 DATA 00011122222
5290 DATA 00001122222
5300 DATA 01222222222
5310 DATA 01122222222
5320 DATA 00112222222
5330 DATA 00011222222
5340 DATA 00011122222
5350 '*** Ergebnis ***
5360 IF g5(h)>100 THEN PRINT "AUSGEZEICH
NET !!!!!":PRINT"Sie Schlingel, Herr Net
zer !":PRINT "Sich einfach hier einzusch
leichen !":RETURN
5370 IF g5(h)>80 THEN PRINT "SEHR GUT !!
!!!":PRINT"Es geht aber noch besser !":R
ETURN
5380 IF g5(h)>60 THEN PRINT "SCHON GANZ
GUT !!!!":PRINT "Aus Ihnen koennte noch
was werden !":RETURN
5390 IF g5(h)>50 THEN PRINT "NA JA !!!!"
:PRINT"Probieren Sie mal weiter, Sie ...
....":RETURN
5400 PRINT"Nun, nochmal von ganz vorne:
":PRINT:PRINT"Ein Fussball ist ein ":PRI
NT"ganz aus Leder gefertigter":PRINT"run
der Gegenstand, der mit Luft":PRINT"gefu
ellt ist und .....":RETURN
5410 ' Start
5420 INPUT "Wollen Sie ein Spiel laden <
J/N> ":es$
5430 IF es$="J" OR es$="j" THEN 6070
5440 DIM x$(18),h$(5,5),u(18),s(18),n(18
),pp(18),np(18),pt(18),nt(18),pa(18),va(
18),m$(17)
5450 pr(1)=100000:pr(2)=200000:pr(3)=350
000:pr(4)=550000:pr(5)=800000
5460 FOR i=1 TO 18: READ x$(i):va(i)=i
5470 d=INT(RND(1)*5+1):w(d)=w(d)+1:IF w(
d)>4 OR (d=5 AND w(5)>2) THEN 5470
5480 x$(i)=STR$(d)+x$(i)
5490 NEXT
5500 ' ** Spielsteuerung **
5510 INPUT "Anzahl der Mitspieler (max.3
)":a$:sp=VAL(a$):CLS:IF sp<1 OR sp>3 THE
N 5510
5520 CLS
5530 PRINT"Sie koennen das Management fo
lgender    Mannschaften uebernehmen:":PR
INT
5540 FOR i=1 TO 18:PRINT i:TAB(5):MID$(x
$(i),3,LEN(x$(i))-1):NEXT
5550 FOR i=1 TO sp
5560 LOCATE 9,23:PRINT"                "
5570 LOCATE 1,23:PRINT"Spieler":i;:INPUT
 q$:IF VAL(q$)<1 OR VAL(q$)>18 THEN 5570
5580 sp(i)=VAL(q$):LOCATE 21,sp(i)+3:PEN
 3:PRINT CHR$(24);"<<<<<<";CHR$(24);:PEN
 1:NEXT
5590 FOR i=1 TO 17:READ m$(i):NEXT
5600 FOR i=1 TO 5:FOR j=1 TO 5
5610 READ h$(i,j):NEXT:NEXT
5620 ' ** Staerke MA **
5630 DIM ma(sp,20,4),el(11,sp),v(20,sp),
ju(sp,4)
5640 FOR h=1 TO sp
5650 FOR j=1 TO 2:ma(h,j,1)=2.2+RND(1):m
a(h,j,4)=1:ma(h,j,0)=INT(ma(h,j,1)*6000)
:NEXT
5660 FOR j=3 TO 6:ma(h,j,1)=2.2+RND(1):m
a(h,j,4)=2:ma(h,j,0)=INT(ma(h,j,1)*6000)
:NEXT
5670 FOR j=7 TO 11:ma(h,j,1)=2.2+RND(1):
ma(h,j,4)=3:ma(h,j,0)=INT(ma(h,j,1)*6000
):NEXT
5680 FOR j=12 TO 15:ma(h,j,1)=2.2+RND(1)
:ma(h,j,4)=4:ma(h,j,0)=INT(ma(h,j,1)*600
0):NEXT
5690 dm(h)=2000000:up$(h)="IN DER 1.RUND
E":dp$(h)="IN DER 1.RUNDE"
5700 FOR j=1 TO 15:r=INT(ma(h,j,1)):vv=I
NT(pr(r)+(RND(1)*ma(h,j,1)*20000)-(RND(1
)*ma(h,j,1)*20000)):sk(h)=sk(h)+vv:ma(h,
j,2)=5:NEXT:sk(h)=sk(h)+2000000
5710 GOSUB 1190
5720 NEXT
5730 RETURN
5740 MODE 2:PRINT:PRINT:PRINT:PRINT
5750 POKE &B1C8,&0:POKE &B1CF,&A0:POKE &
B1D0,&A
5760 PRINT "   DFB - POKAL "
5770 POKE &B1C8,&1:POKE &B1CF,&C0:POKE &
B1D0,&30:POKE &B1D1,&C:POKE &B1D2,&3
5780 RETURN
5790 MODE 2:PRINT:PRINT:PRINT:PRINT
5800 POKE &B1C8,&0:POKE &B1CF,&E0:POKE &
B1D0,&E
5810 PRINT "  UEFA - POKAL "
5820 POKE &B1C8,&1:POKE &B1CF,&C0:POKE &
B1D0,&30:POKE &B1D1,&C:POKE &B1D2,&3
5830 RETURN
5840 ' ***  Sichern Spiel ***
5850 CLS: LOCATE 10,10:INPUT "Name der D
atei ":nam$
5860 OPENOUT nam$
5870 PRINT#9,a,sp,t3,v,z
5880 FOR i=1 TO 18
5890 PRINT#9,np(i),nt(i),pp(i),pt(i),s(i
),u(i)
5900 NEXT
5910 FOR i=1 TO sp
5920 PRINT#9,az(i),dm(i),dp(i),ef(i),gh(
i),qu(i),pl(i),sk(i),sp(i),tb(i),up(i),z
u(i)
5930 NEXT
5940 FOR i=1 TO sp:FOR j=1 TO 4:PRINT#9,
ju(i,j):NEXT:NEXT
5950 FOR i=1 TO sp:FOR j=1 TO 11:PRINT#9
,el(j,i):NEXT:NEXT
5960 FOR i=1 TO sp:FOR j=1 TO 20:PRINT#9
,v(j,i):NEXT :NEXT
5970 FOR i=1 TO sp: FOR j=1 TO 20
5980 FOR k=1 TO 4:PRINT#9,ma(i,j,k)
5990 NEXT:NEXT:NEXT
6000 FOR i=1 TO 18:PRINT#9,x$(i):NEXT
6010 FOR i=1 TO sp:PRINT#9,up$(i):PRINT#
9,dp$(i):NEXT
6020 FOR i=1 TO 17:PRINT#9,m$(i):NEXT
6030 FOR i=1 TO 5:FOR j=1 TO 5
6040 PRINT#9,h$(i,j)
6050 NEXT:NEXT
6060 CLOSEOUT:flag1=255:GOTO 110
6070 ' *** Laden Spiel ***
6080 CLS: LOCATE 10,10:INPUT "Name der D
atei ":nam$
6090 OPENIN nam$
6100 INPUT#9,a,sp,t3,v,z
6110 DIM x$(18),h$(5,5),u(18),s(18),n(18
),pp(18),np(18),pt(18),nt(18),pa(18),va(
18),m$(17)
6120 DIM ma(sp,20,4),el(11,sp),v(20,sp),
ju(sp,4)
6130 FOR i=1 TO 18
6140 INPUT#9,np(i),nt(i),pp(i),pt(i),s(i
),u(i)
6150 NEXT
6160 FOR i=1 TO sp
6170 INPUT#9,az(i),dm(i),dp(i),ef(i),gh(
i),qu(i),pl(i),sk(i),sp(i),tb(i),up(i),z
u(i)
6180 NEXT
6190 FOR i=1 TO sp:FOR j=1 TO 4:INPUT#9,
ju(i,j):NEXT:NEXT
6200 FOR i=1 TO sp:FOR j=1 TO 11:INPUT#9
,el(j,i):NEXT:NEXT
6210 FOR i=1 TO sp:FOR j=1 TO 20:INPUT#9
,v(j,i):NEXT :NEXT
6220 FOR i=1 TO sp: FOR j=1 TO 20
6230 FOR k=1 TO 4:INPUT#9,ma(i,j,k)
6240 NEXT:NEXT:NEXT
6250 FOR i=1 TO 18:INPUT#9,x$(i):NEXT
6260 FOR i=1 TO sp:INPUT#9,up$(i):INPUT#
9,dp$(i):NEXT
6270  FOR i=1 TO 17:INPUT#9,m$(i):NEXT

6280 FOR i=1 TO 5:FOR j=1 TO 5
6290 INPUT#9,h$(i,j)
6300 NEXT:NEXT
6310 FOR i=1 TO 18:x$(i)=" "+x$(i):NEXT
6320 CLOSEIN :GOTO 70
```

Listing 2. »Manager« (Schluß)



Retten Sie Ihren Spion!

Als Agent in einem fremden Staat haben Sie viele Aufgaben. Heute ist mal wieder alles schief gegangen und Sie müssen Ihren Spitzeninformanten vor dem Zugriff der gegnerischen Abwehr schleunigst retten.

Ihr bedrohter Geheimagent muß aus dem schwer bewachten Staat »Diktatolos« fliehen. Dazu ist es erst einmal nötig ihn zu wecken, da er von feindlichen Kräften in einen todesähnlichen Tiefschlaf versetzt wurde. Sie haben 30 Sekunden Zeit, mit dem Lidheber sein Auge zu öffnen. Den Lidheber steuern Sie, indem Sie die Taste »8« im Ziffernblock schnell hintereinander drücken. Aber Vorsicht: Es bringt nichts, wenn Sie immer wieder sinnlos auf die Taste hämmern. Sie muß gleichmäßig bewegt werden.

Je nachdem, wie weit Sie das Auge Ihres Agenten öffnen können, haben Sie in den nächsten Szenen mehr oder weniger Zeit, sein Leben zu retten. Die Anweisungen was wo getan werden muß, finden Sie immer am Anfang eines Bildes.

Zwischen den Zeilen 2020 und 2030 können Sie beliebig viele neue Aufgaben programmieren (mit »RENUM« Platz schaffen), sofern diese im Steuerprogramm ab Zeile 2280 eingebunden werden. Das Spielfeld selbst ist als Matrix (40 x 25) definiert, in der die notwendigen Informationen stehen. Alle weiteren Anweisungen gibt das Programm auf dem Bildschirm aus.

(Stefan Langkau)

t,rt,y,y1	: Zeichenvariablen für das Auge
tr%,—tr%	: obere/untere Grenze für das Augenlid
tri%	: Schrittweite für Linien vom Augenlid
zufallszahl%	: Anzahl gesamte Durchläufe (auf/zu)
durchlauf%	: reale Anzahl der Durchläufe
p% bis p7%	: Augenlid-Linie (Shape)
pl1 bis pl12	: Lidheber (Shape)
intens%	: Höhe des Lidhebers
intensity%	: Leistung in Prozent
bild%	: Zeigt aktuelles Bild an
a$ / re$	: Tastaturabfrage (wechselnde Funktionen)
spielfeld (40, 25)	: Matrix für Spielfeld (von Aufgabe 2)
timer	: Anzahl der Zeiteinheiten
timer%	: Zwischenwert zur Rettung des Inhaltes von »timer« über defint-Kommando
score	: Zeitgutschrift
x,y,hx,h1x,hy,h1y,xp,yp auftrieb,rocket,	: Positionsvariable (nach Startbild)
abfrage,abf	: Boolesche Variablen (Werte:o/1)
bild	: wählt Spielfeld für Aufgabe 2 aus
zeit,zahl,a,b,t,t%	: Laufvariable (wechselnde Funktionen)
aufgabe	: wählt Spielfeld in Aufgabe 3
zaehlwerk	: legt Schußintervalle für Barrieren fest
tim,zaehler	: Bewirken regelmäßigen Aufruf der »Timer«-Routine in Zeile
symb,n,l	: Explosion (optisch und akustisch)
aw	: legt Rakete fest
rx	: Höhe der Rakete
t1x,t1y,tx(15), ty(15),z%	: Position und Bewegung der Meteore
zuf	: Zufallszahl
po	: Laufvariable

Alle anderen Strings (zum Beispiel start$) sind im Spielverlauf auftauchende Symbole, bei denen aus dem Namen hervorgeht, was sie bedeuten.

Variablenliste

```
10 REM*********************************
**********
20 REM**********  Written     1985      **
********
30 REM**********            by          **
********
40 REM**********     Stefan  Langkau    **
********
50 REM**********     Kapellnerstr.32    **
********
60 REM********** 4048 Grevenbroich 5    **
********
70 REM**********     Tel.02182/1545     **
********
80 REM*********************************
*******
90 KEY 128,"mode 2:list"+CHR$(13):KEY 12
9,"?chr$(23)chr$(0):run"+CHR$(13)
100 PRINT CHR$(23)CHR$(0)
110 MODE 1
120 ORIGIN 320,220
130 INK 1,0:INK 0,12:INK 2,20:INK 3,3
140 DEF FNf(t,y)=SIN(t)*y:DEF FNg(t,y)=C
OS(t)*y
150 y=60:y1=20
160 FOR t=0 TO 2*PI STEP PI/30:MOVE 0,0:
DRAW FNg(t,y),FNf(t,y),2:MOVE 0,0:DRAW F
Ng(t,y1),FNf(t,y1),1
170 NEXT
180 RAD:t=5.38:RT=2.24:tr%=-60 :tri%=-2
190 y=160
200 IF INT(t*100)/100=INT(200*PI)/100 TH
EN t=0
210 ORIGIN 320,120:PLOT FNf(t,y),FNg(t,y
),1:ORIGIN 320,320:PLOT FNf(rt,y),FNg(rt
,y),1:t=t+0.01:Rt=Rt+0.01:IF INT(t*100)/
100=0.88 THEN 230
220 GOTO 200
230 GOTO 2280
240 PRINT"Versuche den 'Lid-heber' moegl
ichst weitnach oben zu bewegen. Du errei
chst dies durch schnelles druecken der T
aste'8' imTastenfeld. Entscheidend ist-n
ach Ablaufder Zeit(30sek.)- die Hoehe de
s          'Lid-heber''s!!!"
250 a$=INKEY$:IF a$="" THEN 250
260 CLS
270 zufallszahl%=2*INT(RND*4)+1
280 PRINT CHR$(23)CHR$(1):GOSUB 2790
290 p2%=-p%-30:p3%=p1%:p4%=-p%+30:p5%=p1
%:p7%=CINT(SIN(2.24)*160)
300 ORIGIN p%+320,p1%+120
310 p%=0:p1%=0:p7%=2*p7%
320 FOR t%=-tr% TO tr% STEP tri%
330 p3%=t%:p5%=t%:GOSUB 2820
340 NEXT
350 IF A=1 THEN FOR I%=1 TO 1000:NEXT :A
=0:ELSE IF A=0 THEN A=1
```

Listing. »Escape«


```
360 tr%=-tr%:tri%=-tri%:durchlauf%=durch
lauf%+1:IF Durchlauf%=zufallszahl% THEN
380 ELSE GOTO 320
370 END
380 pl1=300:pl2=0
390 PRINT CHR$(23)CHR$(0)
400 tr%= 60
410 AFTER 1500  GOSUB 490
420 pl3=80:pl5=60:pl8=50:pl6=pl5+intens%
:pl9=60:pl10=pl6+10:pl7=10:pl11=20
430 IF abfrage%=0 THEN abfrage%=1:GOSUB
2850
440 GOSUB 2860 :GOSUB 2870
450 IF INKEY(11)=0 AND INKEY$="8" THEN i
ntens%=intens%+2 ELSE intens%=intens%-1
460 IF intens%<=0 THEN intens%=0
470 IF TEST(pl7-9,pl10+2)=1 AND pl10>154
 THEN tr%=tr%-2:IF tr%=58 THEN PRINT CHR
$(23)CHR$(1):GOSUB 2880:GOSUB 2790:GOSUB
 2840:GOSUB 2830:GOSUB 2820 ELSE IF tr%<
58 THEN PRINT CHR$(23)CHR$(1):GOSUB 2880
:GOSUB 2790:GOSUB 2840:GOSUB 2820:GOSUB
2830
480 GOTO 420
490 t%=REMAIN(0)
500 MODE 2
510 tr%=intens%-87
520 IF tr%<0 THEN 540
530 intensity%=INT((tr%/117)*100)
540 LOCATE 1,5
550 IF intensity%>75 THEN PRINT"Gratulie
re Du hast ";intensity%;" Prozent Leistu
ngsfaehigkeit erreicht!"
560 IF intensity%>50 AND intensity%<=75
THEN PRINT"Du hast ein gutes Ergebnis er
reicht. Dein Schuetzling hat ";intensity
%;"Prozent seiner maximalen Leistungsfae
higkeit erreicht."
570 IF intensity%>20 AND intensity%<=50
THEN PRINT"Du hast ein sehr schlechtes E
rgebnis erzielt (";intensity%;"%)"
580 IF intensity%<=20 THEN PRINT"Leider
verloren!!!!!                    Dein Sc
huetzling kann praktisch nicht   schnell
 reagieren und hat somit keine   Chance!
":PRINT"



Noch ein Versuch?(j/n)";:LINE INPUT re$:
IF re$="j" THEN RUN ELSE END
590 IF intensity%>100 THEN intensity%=10
0
600 bild%=1
610 LOCATE 10,15:PRINT">ENTER<";
620 a$=INKEY$:IF a$<>CHR$(13) THEN 620
630 GOTO 2280
640 REM=====================================
=======================================
650 REM 2.Bild
660 REM=====================================
=======================================
670 MODE 1
680 CLS:PRINT"Versuchen Sie das Ziel zu
erreichen!Vermeiden Sie jede Beruehrung
mit den Blauen Steinen("CHR$(127)").Rote
 Steine("CHR$(238)","CHR$(254)")";"bede
uten Zeitgut-schriften.Sie haben nur 400
 Zeiteinheiten um die folgenden Aufgaben
 zu erfuellen."
690 PRINT"Bedenken Sie bei allem das Sie
 tuen,     dass die Schwerkraft Sie unerb
ittlich   nach unten zieht. Sie koennen
dieser nurmit dem 'ANTI-GRAVITATIONSGENE
RATOR'    entgegenwirken, den Sie mit de
r oberen  Pfeiltaste aktivieren koennen.
"
700 PRINT">ENTER<"
710 a$=INKEY$:IF a$<>CHR$(13)THEN 710 EL
SE GOTO 720
720 RESTORE
730 MODE 1
740 DIM spielfeld(40,25) :
750     GOSUB 1010:REM titel+erklaerung

760 timer=400
770 EVERY 50,1 GOSUB 1680
780 DI:LOCATE x,y:PRINT" "
790 IF INKEY(8)=0 OR INKEY(74)=0 THEN x=
x-1 :GOSUB 920
800 IF INKEY(1)=0 OR INKEY(75)=0 THEN x=
x+1 :GOSUB 930
810 IF INKEY(0)=0 OR INKEY(72)=0 THEN IF
 auftrieb=1 THEN y=y-2:GOSUB 980: ELSE I
F auftrieb=0 THEN y=y-1
820 IF INKEY(0)=0 OR INKEY(72)=0 AND auf
trieb=0 THEN auftrieb=1 ELSE IF INKEY(0)
=-1 OR INKEY(72)=-1 THEN auftrieb=0
830 IF x<1 THEN x=1 ELSE IF y<1 THEN y=1
840 IF x>20 AND y<=1 THEN 1670
850 IF y>=24 AND x>24 AND bild=0 THEN 16
70 ELSE IF y>=24 THEN y=24
860 y=y+1 :IF y<=1 THEN y=1
870 IF spielfeld(x,y)<>0 THEN GOSUB 910
880 LOCATE x,y:PRINT hero$:GOSUB 2360 :E
I
890 FOR zeit=1 TO  50  :NEXT zeit
900 GOTO 780
910 IF spielfeld(x,y)=1 THEN y=y-1:RETUR
N
920 IF spielfeld(x,y)=1 THEN x=x+1:RETUR
N
930 IF spielfeld(x,y)=1 THEN x=x-1:RETUR
N
940 IF spielfeld(x,y)=2 THEN GOTO 1660
950 IF spielfeld(x,y)=3 THEN score=score
+5:spielfeld(x,y)=0:GOSUB 1700
960 IF spielfeld(x,y)=4 THEN score=score
+10:spielfeld(x,y)=0:GOSUB 1700:RETURN
970 RETURN
980 IF spielfeld(x,y+1)=1 THEN y=y+1 ELS
E IF spielfeld(x,y+1)=2 THEN 1660
990 IF spielfeld(x,y)=3 THEN score=score
+10:spielfeld(x,y)=0 :GOSUB 1700
1000 RETURN
1010
1020 hero$=CHR$(248)
1030 REM Spielfeld
1040 IF bild=0 THEN RESTORE 1160:bild=1
1050 IF bild=2 THEN RESTORE 1410:bild=0
1060 x=10:y=1:LOCATE 1,1
1070 FOR t=1 TO 25
1080 READ a$:FOR s=1 TO LEN(a$):IF MID$(
a$,s,1)="1" THEN PEN 1:PRINT CHR$(143);:
spielfeld(s,t)=1
1090 IF MID$(a$,s,1)="2" THEN PEN 2:PRIN
T CHR$(127) ; :spielfeld(s,t)=2
1100 IF MID$(a$,s,1)="0" THEN PEN 1:PRIN
T CHR$(32);:spielfeld(s,t)=0
1110 IF MID$(a$,s,1)="3" THEN PEN 3:PRIN
T CHR$(238);:spielfeld(s,t)=3
1120 IF MID$(a$,s,1)="4" THEN PEN 3:PRIN
T CHR$(254);:spielfeld(s,t)=4
1130 IF ASC(MID$(a$,s,1))>57 THEN PEN 1:
PRINT MID$(a$,s,1);
```

```
1140 NEXT:PRINT" ":NEXT
1150 PEN 1:RETURN
1160 DATA 11110000001111000001100000000000
00000001
1170 DATA 111000000000000000011>>>>>>Ziel
<<<<<<<1
1180 DATA 11111111110001100111110000001111
11111201
1190 DATA 11122222200121000011111000000111
11111002
1200 DATA 11111300000030000011112000300222
22220002
1210 DATA 11100110000011000011112000000011
11100011
1220 DATA 11130100000111000011111110000011
11110011
1230 DATA 11110000121020000111111111100002
02220001
1240 DATA 11000000000000001111222222220001
01111001
1250 DATA 11100001111100011111111111100000
00110001
1260 DATA 11110002222200111111111111111100
00000011
1270 DATA 11110001111100211111222222222220
00000011
1280 DATA 11110000222000221111111111111100
00000002
1290 DATA 11110000010000021111111111110000
00000022
1300 DATA 11111300022200011111111111100000
04220031
1310 DATA 11110000000000000000341110000002
02220011
1320 DATA 11110001111110211111111110000002
22220001
1330 DATA 11120000222200111111111110000000
00011001
1340 DATA 11110000021110000000000000002222
22011101
1350 DATA 11111000002200000000000000022222
22000101
1360 DATA 11112000002220022211111001111111
11100001
1370 DATA 11112000000000021111110000111111
11000001
1380 DATA 11111111111111111111220000022220
20000001
1390 DATA 11111111111111111111110000000000
00000001
1400 DATA 11111111111111111111111111111111
11111111
1410 DATA 11111111111111111111111111111111
11111111
1420 DATA 11122222000000000000222222000000
00000111
1430 DATA 11222200000000000000000000000011
11120111
1440 DATA 11111000001112222221111111100220
22100011
1450 DATA 11100000000111000011111111100001
10000111
1460 DATA 11100002000022000002222220000202
22000222
1470 DATA 11100011100010000000111100001111
11100011
1480 DATA 11000111110000021200001100000011
11000111
1490 DATA 11100001100000022200004112000022
22000021
1500 DATA 11200000300011001000022213000000
00000111
1510 DATA 11221000111020001110111111110000
00001111
1520 DATA 11110000000011102200001112000000
00000011
1530 DATA 11122200001001100100000020000000
00002221
1540 DATA 11111000011102200100001111000110
10002211
1550 DATA 11122220002001001110000212000000
00022221
1560 DATA 11110000000011100000000011100000
00000111
1570 DATA 11111000000001000000011111100000
00001111
1580 DATA 11111111100000000000000011100000
00022111
1590 DATA 11130011110000000000000011100000
00033111
1600 DATA 11110011111000000000222111100000
00011111
1610 DATA 11111000000000000000001111100000
00000111
1620 DATA 11111111111000220000000110000000
00022221
1630 DATA 10000000000000000000000011000000
00000001
1640 DATA 10000000000000000000000011000000
00000001
1650 DATA 10000000011000000000000011>>>>Z
iel<<<<1
1660 GOTO 2370
1670 IF bild=1 THEN zeit=REMAIN(1):bild=
2:GOSUB 1010:x=10:y=24:GOTO 770 ELSE IF
bild=0 THEN zeit=REMAIN(1):score=score+t
imer:bild%=2:GOTO 2280
1680 LOCATE 30,1
1690 PRINT USING "###";timer:timer=timer
-1:RETURN
1700 LOCATE 22,1:PRINT USING"####";score
:RETURN
1710 REM------------------------------------
------------------------------------
1720 REM  3.Aufgabe
1730 REM------------------------------------
------------------------------------
1740 timer%=CINT(score)
1750 DEFINT a-z:timer=timer%
1760 bild%=3
1770 aufgabe=0
1780 aufgabe=aufgabe+1:GOSUB 2900
1790 IF aufgabe=1 THEN GOSUB 2480:GOSUB
3040
1800 MOVE hx,hy:PRINT hub$;
1810 h1x=hx:h1y=hy
1820 IF JOY(0)<>0 THEN GOTO 1980
1830 IF INKEY(0)=0 THEN hy=hy+5
1840 IF INKEY(2)=0 THEN hy=hy-5
1850 IF INKEY(8)=0 THEN hx=hx-5
1860 IF INKEY(1)=0 THEN hx=hx+5:IF hx>63
0 THEN GOTO 2280
1870 hy=hy-2
1880 MOVE h1x,h1y:PRINT hub$;:MOVE hx,hy
:PRINT hub$;
1890 IF ABS(hx-440)<5 AND ABS(hy-192)<5
 THEN GOTO 1780
1900 zaelwerk=zaelwerk+INT(RND*3)+1:IF z
aelwerk>=25 AND aufgabe=1 THEN GOSUB 333
0:zaelwerk=0
1910 IF aufgabe=2 THEN z%=INT(RND*15)+1:
GOSUB 3530:IF hy>=398 THEN 1780
1920 IF hy<190 AND hx<240 THEN IF zaelwe
rk>5 AND aufgabe=3 THEN zaelwerk=0:GOSUB
 3710
1930 IF (TEST(hx+16,hy-16)<>0 OR TEST(hx
+16,hy)<>0 OR TEST(hx,hy)<>0 OR TEST(hx,
hy-16)<>0) THEN IF aufgabe<3 AND hx<=304
 GOTO 2370 ELSE IF (TEST(hx+16,hy-16)=3
OR TEST(hx+16,hy)=3 OR TEST(hx,hy)=3 OR
TEST(hx,hy-16)=3) GOTO 2280 ELSE GOTO 23
70
1940 tim=tim+1.5:IF tim>=10 THEN GOSUB 3
790:tim=0:IF timer<0 THEN GOTO 2370
1950 IF rocket=0 AND aufgabe=1 THEN rock
et=1:GOSUB 3240
1960 IF rocket=1 AND aufgabe=1 THEN GOSU
B 3300
1970 GOTO 1810
1980 IF INKEY(72)=0 THEN hy=hy+5
1990 IF INKEY(73)=0 THEN hy=hy-5
2000 IF INKEY(74)=0 THEN hx=hx-5
2010 IF INKEY(75)=0 THEN hx=hx+5:IF hx>6
30 THEN GOTO 2280 ELSE GOTO 1870
2020 GOTO 1870
2030 REM------------------------------------
------------------------------------
2040 REM Letztes Bild
2050 REM------------------------------------
------------------------------------
2060 bild%=4:GOSUB 3800
2070 TAG
2080 hx=448:hy=64
2090 PLOT 0,0,3:MOVE 448,64:PRINT ge$;:P
LOT 0,0,3:MOVE 448,48:PRINT ge1$;
2100 h1x=hx:h1y=hy
2110 zaehler=zaehler+1:IF zaehler>=20 TH
EN zaehler=0:GOSUB 3790:IF timer<0 THEN
2370
2120 hx=hx-1
2130 IF hx<=210 AND hx>150 AND ge$<>lei$
 THEN GOSUB 2200:hy=hy+20:ge$=lei$:ge1$=
lei1$:GOSUB 2200:ELSE IF hx<=210 AND hx>
140 THEN hy=hy+4
2140 IF hx<=360 AND hx> 210 AND ge$<>kr$
 THEN GOSUB 2200:hy=hy-16:ge$=kr$:ge1$="
":GOSUB 2200
2150 IF hx<=110 AND ge$<>spr$ THEN GOSUB
 2200:ge$=spr$:ge1$=spr1$:GOSUB 2200: EL
SE IF hx<=130 THEN hy=hy-3:
2160 GOSUB 2200:GOSUB 2190
2170 IF hy<=16 THEN GOTO 2210
2180 GOTO 2100
2190 PLOT 0,0,3:MOVE hx,hy:PRINT ge$;:PL
OT 0,0,3:MOVE hx,hy-16:PRINT ge1$;:RETUR
N
2200 PLOT 0,0,3:MOVE h1x,h1y:PRINT ge$;:
PLOT 0,0,3:MOVE h1x,h1y-16:PRINT ge1$;:R
ETURN
2210 REM------------------------------------
------------------------------------
2220 REM     Abschied
2230 REM------------------------------------
------------------------------------
2240 MODE 1:INK 0,0:INK 1,26
2250 LOCATE 1 ,10:PRINT"Herzlichen Gluec
kwunsch ";name$;" du hast deine Sache gu
tgemacht.":PRINT:PRINT"Ausserdem haettes
t du noch ";timer;"Zeiteinheiten uebrig
gehabt."
2260 a$=INKEY$:IF a$="" THEN 2260 ELSE E
ND
2270 GOTO 2270
2280 REM Hauptprogramm
2290 REM**************
2300 IF bild%=0 THEN GOSUB 4230:REM Eing
aben
2310 IF bild%=0 THEN GOTO 240:REM intens
ity
2320 IF bild%=1 THEN GOTO 650:REM 2.Bild
2330 IF bild%=2 GOTO 1710
2340 IF bild%=3 THEN 2030
2350 END
2360 FOR t=1 TO 100-intensity%:NEXT:RETU
RN:REM Reaktionsverzoegerung
2370 REM------------------------------------
------------------------------------
2380 REM                              Spielend
e
2390 REM------------------------------------
------------------------------------
2400 t%=REMAIN(1):t%=REMAIN(0)
2410 GOSUB 2660
2420 MODE 1:LOCATE 10,10:PRINT"Leider is
t dein Schuetzling toedlich verungluecke
!!":PRINT:PRINT"
                Noch ein Versuch?     (j/
n)";
2430 a$=INKEY$:IF LOWER$(a$)="j" THEN RU
N ELSE IF a$<>"n" THEN 2430
2440 END
2450 REM------------------------------------
------------------------------------
2460 REM       Titel v.Hoehlen
2470 REM------------------------------------
------------------------------------
2480 MODE 1:PRINT"Dein Schuetzling steig
t jetzt in einen  bereitstehenden Hubsch
rauber ein.       Bewege diesen durch di
e Hoehlen ohne mit weissen oder gelben L
inien und Gegen-  staenden zusammenzutre
ffen."
2490 PRINT"In der ersten Hoehle muss das
 Rote Ziel genau(!) von unten angeflogen
 werden.Diezweite Hoehle muss nach oben
verlassen  werden, waehrend in der letzt
en Hoehle  ein roter Landeplatz zu errei
chen ist!
2500 PRINT"Gefaehrlich sind: ";CHR$(130)
;CHR$(131);CHR$(137);CHR$(133)
2510 PRINT"Der Landeplatz in der dritten
 Hoehle sieht so aus: ";CHR$(134)+CHR$(1
34)+CHR$(134)
2520 PLOT 0,0,1
2530 PRINT CHR$(23)CHR$(1):TAG
2540 DEFREAL y
2550 xp=-16:yp=16:MOVE 500,116:PRINT CHR
$(129);:MOVE 0,0:DRAW 500,100,1
2560 FOR t=0 TO 100 STEP 2:MOVE 500,t:DR
AW 516,t,1:NEXT
2570 man$=CHR$(251):MOVE xp-1,yp-0.2:PRI
NT man$;
2580 IF strin=0 THEN strin=1:MOVE xp-1,y
p-0.2:PRINT CHR$(250);:CALL &BD19:MOVE x
p,yp:PRINT CHR$(251);:CALL &BD19 ELSE IF
 strin=1 THEN strin=0:MOVE xp-1,yp-0.2:P
RINT CHR$(250);:CALL &BD19:MOVE xp,yp:PR
INT CHR$(251);:CALL &BD19
2590 IF INKEY$=CHR$(13)THEN RETURN
2600 IF xp>500 THEN 2630
2610 FOR t=1 TO 50:NEXT
2620 xp=xp+1:yp=yp+0.2:GOTO 2580
2630 MOVE xp-1,yp:PRINT CHR$(250);:TAGOF
F:PRINT:PRINT">ENTER<"
2640 a$=INKEY$:IF a$<>CHR$(13) THEN 2640
2650 RETURN
2660 REM------------------------------------
------------------------------------
2670 REM              Explosion
2680 REM------------------------------------
------------------------------------
```

Listing. »Escape« (Fortsetzung)

```
2690 TAGOFF:LOCATE 1,1:PRINT CHR$(23)CHR
$(1)
2700 symb=136:PLOT 0,0,1
2710 TAG
2720 FOR n=15 TO 1 STEP -1
2730 FOR l=1 TO 5:IF bild%=1 OR bild%=0
THEN MOVE (x-1)*16,400-(16*y):PRINT CHR$
(symb);: ELSE MOVE hx,hy:PRINT CHR$(symb
);
2740 symb=symb+1:NEXT
2750 SOUND 1,426,40,n,,,1
2760 NEXT
2770 TAGOFF
2780 RETURN
2790 REM unterprogramme bildnr.1
2800 REM-----------------------------------
-----------------------------------
2810 p%=CINT(SIN(5.38)*160):p1%=CINT(COS
(5.38)*160) :RETURN
2820 MOVE 0,0:DRAW p2%,p3%,1:DRAW p4%,p5
%:DRAW p7%,0,1:RETURN
2830 MOVE 0,0:LOCATE 1,1:PRINT CHR$(23)C
HR$(0):RETURN
2840 ORIGIN p%+320,p1%+120:p%=0:p1%=0:RE
TURN
2850 ORIGIN p11,p12:MOVE 0,0:DRAW p13,0,
1:DRAW p13,p15:DRAW p18,p15:DRAW p18,p16
:DRAW p19,p16:DRAW p19,p110:DRAW p17,p11
0,2:DRAW p17,p16,1:DRAW p111,p16:DRAW p1
11,p15:DRAW 0,p15:DRAW 0,0:RETURN
2860 ORIGIN p11,p12:MOVE p18,p15:DRAW p1
8,p16,0:DRAW p19,p16:DRAW p19,p110:DRAW
p17,p110:DRAW p17,p16:DRAW p111,p16:DRAW
 p111,p15:RETURN
2870 ORIGIN p11,p12:MOVE p18,p15:DRAW p1
8,p16,1:DRAW p19,p16:DRAW p19,p110:DRAW
p17,p110,2:DRAW p17,p16,1:DRAW p111,p16:
DRAW p111,p15:RETURN
2880 p3%=-tr%:p5%=-tr%:IF tr%<-60 THEN G
OTO 490 ELSE RETURN
2890 REM===================================
=======================================
2900 REM Unterprogramme 3.Aufgabe
2910 REM===================================
=======================================
2920 IF aufgabe>3 THEN GOTO 2280 ELSE ON
 aufgabe GOTO 2930,3380,3570
2930 SYMBOL AFTER 128
2940 SYMBOL 129,0,127,8,140,146,242,12,1
8:REM Hubschrauber
2950 SYMBOL 130,0,24,36,36,60,24,60,66:R
EM Rakete
2960 SYMBOL 131,8,73,85,34,28,8,28,34 :R
EM Barriere
2970 SYMBOL 137,34,28,8,28,34,85,73,8:RE
M Barriere 1
2980 SYMBOL 132,60,90,165,165,153,129,66
,60:REM Meteoriten
2990 SYMBOL 133,1,2,127,15,4,12,18,33:RE
M Sperrkanone
3000 SYMBOL 134,0,255,66,36,24,24,36,66:
REM Landeplatz
3010 SYMBOL 135,90,90,90,66,36,36,24,24:
REM Startfeuer
3020 SYMBOL 136,0,0,1,138,84,34,65,128:R
EM Background
3030 SYMBOL 135,0,24,36,36,36,36,24,24:R
ETURN
3040 MODE 1:GOSUB 3740
3050 PRINT CHR$(22)CHR$(1)
3060 start$=CHR$(135):lande$=CHR$(134):s
perr$=CHR$(133):meteor$=CHR$(132):barrie
r$=CHR$(131):barriere$=CHR$(137):FOR zah
l=1 TO 5:raket$(zahl)=CHR$(130):racket(z
ahl)=1:NEXT:hub$=CHR$(129)
3070 INK 0,0:INK 1,26:INK 2,12:INK 3,3:B
ORDER 0
3080 WINDOW #0,1,30,1,25:WINDOW #1,31,40
,1,25
3090 ORIGIN 0,0:DRAW 0,399,1:DRAW 480,39
9:DRAW 480,0:DRAW 0,0
3100 MOVE 176,399:DRAW 192,150:DRAW 224,
268
3110 RESTORE 4120
3120 READ a,b:IF a=-1 AND b=-1 THEN 3130
 ELSE DRAW a,b:GOTO 3120
3130 MOVE 0,64
3140 READ a,b:IF a=-1 AND b=-1 THEN 3150
 ELSE DRAW a,b:GOTO 3140
3150 PEN 2:LOCATE 26,6:PRINT barriere$:L
OCATE 21,19:PRINT barrier$:PEN 1
3160 PEN 2:FOR t=10 TO 14:LOCATE t,22:PR
INT raket$(t-9):NEXT:PEN 1
3170 LOCATE 3,3:PRINT CHR$(23)CHR$(1)
3180 MOVE 440,176:DRAW 440,192,3:DRAW 45
6,192:DRAW 456,176:PLOT 0,0,1
3190 hx=32:hy=368
3200 WINDOW SWAP 0,1
3210 TAG
3220 RETURN
3230 GOTO 3230
3240 REM rakete starten
3250 REM-----------------------------------
-----------------------------------
3260 RANDOMIZE TIME
3270 aw=INT(RND*5)+10
3280 PLOT 0,0,2:MOVE 16*(aw-1),64:PRINT
raket$(1); :rx=64:PLOT 0,0,1
3290 RETURN
3300 PLOT 0,0,2:MOVE (aw-1)*16,rx:PRINT
start$;:rx=rx+8:MOVE (aw-1)*16,rx:PRINT
start$;:PLOT 0,0,1
3310 IF rx>180 THEN PLOT 0,0,2:MOVE (aw-
1)*16,rx:PRINT start$;:rx=64:rocket=0:PL
OT 0,0,1
3320 RETURN
3330 TAGOFF:MOVE 326,112:DRAW 326,304,1:
MOVE 405,304:DRAW 405,80 :TAG
3340 IF ABS(326-hx-8)<=8 THEN GOTO 2370
ELSE IF ABS(405-hx-8)<=8 GOTO 2370
3350 RETURN
3360 TAGOFF:LOCATE 1,1:GOSUB 1690:TAG:RE
TURN
3370 REM 3. Aufgabe 2.Bild
3380 REM-----------------------------------
-----------------------------------
3390 MODE 1
3400 GOSUB 3740
3410 PRINT CHR$(23)CHR$(0)
3420 RESTORE 4140
3430 READ a,b:MOVE a,b
3440 READ a,b:IF a=-1 AND b=-1 THEN 3450
 ELSE DRAW a,b:GOTO 3440
3450 IF frage=0 THEN RESTORE 4150:frage=
1:GOTO 3430
3460 MOVE 0,0:DRAW 0,399:DRAW 479,399:DR
AW 479,0:DRAW 0,0
3470 hx=16:hy=80
3480 DIM tx(15),t1x(15),ty(15),t1y(15)
3490 FOR t=1 TO 15:tx(t)=112+16*t:IF tx(
t)>368 THEN tx(t)=tx(t)-16*(t/2)
3500 ty(t)=420:NEXT
3510 PRINT CHR$(23)CHR$(1):TAG:MOVE 96,3
99:DRAW 368,399,1
3520 RETURN
3530 t1x=tx(z%):t1y=ty(z%):MOVE t1x,t1y:
PRINT meteor$,:ty(z%)=ty(z%)-INT(RND*19)
-9:MOVE tx(z%),ty(z%):PRINT meteor$;
3540 IF ty(z%)<0 THEN ty(z%)=416:tx(z%)=
INT(RND*272)+112
3550 RETURN
3560 REM        3. Aufgabe  3.Bild
3570 REM-----------------------------------
-----------------------------------
3580 RESTORE 4160
3590 abfrage=0:MODE 1:WINDOW #0,31,40,1,
25 :GOSUB 3740
3600 MOVE 0,0
3610 LOCATE 1,1:PRINT CHR$(23)CHR$(1):TA
G
3620 READ a,b:IF a=-1 AND b=-1 THEN 3630
 ELSE DRAW a,b:GOTO 3620
3630 IF abfrage=0 THEN abfrage=1:MOVE 0,
160:GOTO 3620
3640 SYMBOL 139,0,0,0,24,24
3650 schuss$="  "+CHR$(139)+"  "+CHR$(13
9)+"  "+CHR$(139)+"  "+CHR$(139)
3660 MOVE -10,176:PRINT schuss$;:PLOT 0,
0,2:MOVE 208,176:PRINT sperr$;
3670 landungs$=lande$+lande$+lande$
3680 PLOT 0,0,3:MOVE 384,48:PRINT landun
gs$;:PLOT 0,0,1
3690 hx=80:hy=80 :zuf=-10
3700 RETURN
3710 MOVE zuf,176:PRINT schuss$;:schuss$
=RIGHT$(schuss$,12)+LEFT$(schuss$,1):zuf
=-1*(INT(RND*16)+1):MOVE zuf,176:PRINT s
chuss$;
3720 IF (TEST(hx+16,hy-16)<>0 OR TEST(hx
+16,hy)<>0 OR TEST(hx,hy)<>0 OR TEST(hx,
hy-16)<>0) THEN GOTO 2370
3730 RETURN
3740 PRINT CHR$(23)CHR$(1)
3750 FOR t=0 TO timer STEP 2:MOVE 590,t:
DRAW 630,t,1:NEXT
3760 FOR t=0 TO 400 STEP 2:MOVE 590,t:DR
AW 630,t,3:NEXT
3770 PLOT 0,0,1
3780 RETURN
3790 timer=timer-2:MOVE 590,timer:DRAW 6
30,timer,1:PLOT 0,0,1:RETURN
3800 REM-----------------------------------
-----------------------------------
3810 REM       Letztes Bild
3820 REM-----------------------------------
-----------------------------------
3830 PRINT CHR$(23)CHR$(0)
3840 SYMBOL AFTER 128
3850 SYMBOL 128,28,50,66,34,28,8,60,222
3860 SYMBOL 129,30,28,0,20,18,19,54,0
3870 SYMBOL 131,1,2,2,6,4,12,12,28
3880 SYMBOL 132,28,24,24,16,40,104,80,64
3890 SYMBOL 133,56,69,71,101,43,18,2,0
3900 SYMBOL 134,0,236,220,220,12,26,31,1
3910 SYMBOL 135,28,34,34,34,29,73,127,28
3920 SYMBOL 137,28,0,28,62,98,3
3930 ge$=CHR$(128):ge1$=CHR$(129):spr$=C
HR$(131):spr1$=CHR$(132):kr$=CHR$(133)+C
HR$(134):lei$=CHR$(135):lei1$=CHR$(137)
3940 MODE 0:PRINT CHR$(23)CHR$(0)
3950 abf=0
3960 INK 0,11:INK 1,21:INK 2,14:INK 3,20
:INK 4,18:INK 5,19:INK 6,4:INK 7,13:INK
8,0:INK 9,0
3970 FOR t=0 TO 13*16 STEP 2:MOVE 0,t:DR
AW 480,t,1:NEXT
3980 FOR t=16 TO 256 STEP 2:MOVE 112,t:D
RAW 144,t,3:NEXT
3990 po=112:FOR t=256 TO 320 STEP 2:MOVE
 po,t:DRAW po+32,t,4:po=po+2:NEXT
4000 po=144:FOR t=16 TO 256 STEP 2:MOVE
po,t:DRAW po+64,t+ 64,5:NEXT
4010 po=0:FOR t=16 TO 176 STEP 2:MOVE po
,t:DRAW po+96,t,2:po=po+0.125:NEXT
4020 po=400:FOR t=80 TO 340 STEP 2:MOVE
po,t:DRAW 480,t-80,7:po=po+(po1):po1=po1
+0.001:NEXT
4030 po=430:FOR t=70 TO 160 STEP 2:MOVE
po,t-5:DRAW 480,t-80,8:po=po+(po1):po1=p
o1+0.001:NEXT
4040 RESTORE 4190:MOVE 272,16
4050 READ a,b:IF a=-1 AND b=-1 THEN 4060
 ELSE DRAW a,b,6:GOTO 4050
4060 IF abf=0 THEN abf=1 :MOVE 208,32:GO
TO 4050
4070 WINDOW 16,20,1,25:PAPER 8:CLS
4080 LOCATE 1,1:PRINT CHR$(23)CHR$(1):GO
SUB 3740
4090 RETURN
4100 REM  Data's fuer Unterprogramme
4110 REM-----------------------------------
-----------------------------------
4120 DATA 336,304,400,320,416,320,448,25
6,480,352,-1,-1
4130 DATA 32,96,32,90,112,144,112,112,12
0,112,128,80,144,80,144,48,224,48,224,64
,240,64,288,160,320,144,320,96,336,96,35
2,64,368,80,416,64,480,80,-1,-1
4140 DATA 0,112,48,112,96,136,112,176,96
,224,120,256,96,352,96,400,-1,-1
4150 DATA 352,0,336,64,336,160,368,192,3
52,240,336,272,352,304,368,400,-1,-1
4160 DATA 0,399,480,399,480,0,0,0,-1,-1
4170 DATA 48,160,48,0,160,0,160,160,224,
160,224,176,240,192,240,208,260,256,288,
256,288,272,304,272,304,288,336,320
4180 DATA 448,320,448,304,336,304,336,22
4,432,208,304,208,304,128,432,120,352,12
0,336,128,354,128,368,32,448,32,458,160,
336,160,458,176,458,256,368,256,480,272,
-1,-1
4190 DATA 272,64,288,112,368,64,384,112,
272,64,368,64,368,16,368,64,336,112,288,
112,288,65,288,112,320,64,336,112,384,11
2,384,64,-1,-1
4200 DATA 160,256,176,272,224,64,-1,-1
4210 GOTO 4210
4220 REM-----------------------------------
-----------------------------------
4230 REM Start
4240 REM-----------------------------------
-----------------------------------
4250 WINDOW 1,40,16,25
4260 LOCATE 1,20:INPUT"Name des Spielers
 :";name$
4270 PRINT"Du hast die Aufgabe deinen Sc
huetzling  sicher aus einem schwer bewac
hten feind-lichen Staat zu retten.Zunaec
hst aber   musst du ihn aufwecken. Je we
iter du das Auge oeffnen kannst(durch dr
uecken der Taste '8' im Tastenfeld)"
4280 PRINT"desto schneller kann er im ko
mmenden    Spielverlauf reagieren.Viele
unbekannte Gefahren warten auf deinen Sc
huetzling. Versuche ihn zu retten!!"
4290 a$=INKEY$:IF a$="" THEN 4290
4300 CLS
4310 RETURN
4320 REM===================================
=======================================
```

Listing. »Escape« (Schluß)

Frisch aus Las Vegas — Olli's Crown

Wer liebt nicht das Risiko und den Reiz, der von den Geldspielautomaten ausgeht?

»Olli's Crown« simuliert einen Glückspielautomaten, wie er in fast jeder Kneipe zu finden ist. Mit RUN startet man das Programm, und die mittlere Walze beginnt sich zu drehen. Ein Druck auf die Space-Taste und sie stoppt. Gleichzeitig leuchtet unter der linken Walze die Anweisung »Start« auf, die durch nochmaliges Betätigen der Leer- (für die linke Walze) oder ENTER-Taste (für die mittlere Walze) ausgeführt wird. Startet man keine Walze, so wird die linke nach drei Sekunden automatisch in Bewegung gesetzt. Sie wird — wie auch die rechte — mit der Leertaste angehalten. Der Start der rechten Walze ist automatisch mit dem Stop der linken gekoppelt.

Die Zahlenfolge auf der rechten Seite, die das Wort »Chance« abschließt, steuert die »Superchance«, bei der der nächste Gewinn verfünffacht wird. Bestimmte Kombinationen der drei Walzen setzen die Werte der »Chance« um eins herauf. Ganz unten finden Sie noch die Anzeige der »Kombination«. Stehen die Walzen in der gleichen Reihenfolge, so verdoppelt sich der augenblickliche Kredit und die »Chance« wird wieder um eins heraufgesetzt.

»Risiko« erscheint durch Zufall gesteuert auf dem Bildschirm und bewirkt einen Sondergewinn. Die fünf Zahlen 0 bis 4 werden sehr schnell durchlaufen und mit einem Druck auf die Leertaste wählt man einen Wert aus. Diesen mit 1,5 multipliziert bekommt der Spieler als Kredit gutgeschrieben.

(Oliver Felchner)

```
10 '******************************
20 '*        OLLI'S Crown          *
30 '* Amstrad/Schneider  CPC-464 *
40 '* (c) O.Felchner         1985 *
50 '*Lanferschlade 4  5990 Altena*
60 '******************************
70 REM ****SYMBOLS****
80 SYMBOL AFTER 200
90 SYMBOL 224,3,7,7,15,15,15,15,15
100 SYMBOL 225,255,249,253,252,254,254,2
54,254
110 SYMBOL 226,192,224,224,240,240,240,2
40,240
120 SYMBOL 227,15,15,31,31,63,127,255,0
130 SYMBOL 228,254,254,255,255,255,255,2
55,60
140 SYMBOL 229,240,240,120,56,188,158,25
5,0
150 SYMBOL 231,0,0,0,0,61,126,255,255
160 SYMBOL 232,0,0,56,68,130,2,2,2
170 SYMBOL 233,1,1,0,0,0,0,0,0
180 SYMBOL 234,249,249,254,124,56,0,0,0
190 SYMBOL 235,2,5,1,0,0,0,0,0
200 SYMBOL 230,0,0,0,0,0,0,0,0
210 SYMBOL 200,224,224,64,112,56,60,30,3
1
220 SYMBOL 201,28,28,8,24,60,60,126,255
230 SYMBOL 202,14,14,4,12,28,32,72,208
240 SYMBOL 203,15,15,15,15,0,0,0,0
250 SYMBOL 204,255,255,255,255,0,0,0,0
260 SYMBOL 205,160,96,96,96,0,0,0,0
270 SYMBOL 206,0,127,224,239,224,224,224
,224
280 SYMBOL 207,0,255,0,242,18,34,66,130
290 SYMBOL 208,0,254,2,242,154,138,154,2
42
300 SYMBOL 209,225,226,228,232,239,224,2
55,255
310 SYMBOL 210,2,2,2,2,242,0,255,255
320 SYMBOL 211,194,194,194,194,194,2,254
,252
330 SYMBOL 212,0,0,96,112,120,124,125,12
5
340 SYMBOL 213,0,0,0,0,0,0,255,255
350 SYMBOL 214,0,0,0,3,6,118,254,255
360 SYMBOL 215,125,125,124,120,112,96,0,
0
370 SYMBOL 216,255,255,4,6,2,3,0,0
380 SYMBOL 217,255,251,53,91,213,251,21,
63
390 SYMBOL 218,31,15,15,31,31,31,31,31
400 SYMBOL 219,255,255,255,255,255,231,1
95,129
410 SYMBOL 220,240,240,240,240,240,240,2
40,240
420 SYMBOL 221,31,31,31,31,31,31,31,31
430 SYMBOL 222,129,129,195,231,253,231,2
31,255
440 SYMBOL 223,240,240,240,240,240,240,2
40,240
450 SYMBOL 236,60,66,130,192,192,224,127
,63
460 SYMBOL 237,252,126,71,71,126,124,207
,199
470 SYMBOL 238,60,126,135,131,131,131,19
4,255
480 SYMBOL 239,35,99,97,101,109,109,109,
255
490 SYMBOL 240,60,62,39,35,35,35,227,227
500 SYMBOL 241,240,96,96,96,96,112,241,2
55
510 SYMBOL 242,60,28,24,24,24,28,28,254
520 SYMBOL 243,62,30,12,24,16,0,0,0
530 SYMBOL 244,28,62,115,96,60,3,67,254
540 SYMBOL 245,15,63,109,115,115,109,63,
15
550 SYMBOL 246,240,252,134,190,142,246,1
40,240
560 SPEED INK 5,5
570 INK 0,0:BORDER 0:INK 1,11:INK 2,24:I
NK 3,6:INK 4,18:INK 5,1:INK 6,26:INK 7,1
3:INK 8,18,1:INK 9,8:INK 10,15:INK 11,4:
INK 12,7:INK 13,16
580 DIM ge(3)
590 cred=5:bonus=1
600 ENV 1,1,120,1,120,-1,10
610 co(1)=200:co(2)=200:co(3)=200:sucha=
0
620 REM***Bildschirm***
630 MODE 0
640 DRAWR 638,0,3:DRAWR 0,398:DRAWR -638
,0:DRAWR 0,-398
650 PLOT 0,380,4:DRAWR 638,0:PLOT 0,250,
4:DRAWR 638,0:PLOT 212,250:DRAWR 0,128:P
LOT 428,250:DRAWR 0,128
660 PLOT 200,200,6:DRAWR 240,0:DRAWR 0,3
0:DRAWR -240,0:DRAWR 0,-30
670 PLOT 10,10,10:DRAWR 400,0:DRAWR 0,80
:DRAWR -400,0:DRAWR 0,-80:LOCATE 2,21:PE
N 11:PRINT"COMBINATION"
680 PEN 1:FOR n=1 TO 9:LOCATE 16,n+14:PR
INT n:NEXT:LOCATE 14,24:PRINT"CHANCE"
690 LOCATE 2,15:PEN 13:PRINT CHR$(238)+C
HR$(241)+CHR$(241)+CHR$(242)+CHR$(243)+C
HR$(244)+" ":PEN 7:PRINT CHR$(236)+CHR$
(237)+CHR$(238)+CHR$(239)+CHR$(240)
700 LOCATE 5,17:PEN 2:PRINT"RISIKO":LOCA
TE 3,19:PEN 6:PRINT"0 1 2 3 4"
710 GOTO 1050
720 REM****WALZENSIMULATION****
730 PEN 9:LOCATE 3,5:PRINT " "+CHR$(159)
+" ":LOCATE 3,6:PRINT" "+" "+" ":PEN 6:
LOCATE 16,5:PRINT " "+CHR$(159)+" ":LOCA
TE 16,6:PRINT" "+" "+" "
740 REM
750 a=206:zuf=INT(RND*16)
760 PEN 2
770 LOCATE 10,5:PRINT CHR$(a)+CHR$(a+1)+
CHR$(a+2):LOCATE 10,6:PRINT CHR$(a+3)+CH
R$(a+4)+CHR$(a+5)
780 a$=INKEY$:a$=""
790 IF INKEY(47)=0 THEN ge(1)=a:GOTO 820
800 IF A<230 THEN A=A+6 ELSE a=200
810 GOTO 770
820 kill$=INKEY$:kill$="":IF con=0 THEN
FOR loop=1 TO 80:LOCATE 2,12:PEN 3:PRINT
"START" ELSE 870
830 IF INKEY(18)=0 THEN con=1:LOCATE 2,1
2:PEN 0:PRINT"       ":GOTO 760
840 IF INKEY(47)=0 AND loop>10 THEN 860
850 NEXT
860 LOCATE 2,12:PEN 0:PRINT"       "
870 FOR lo=1 TO 100:NEXT
880 a=212:con=0:PEN 9
890 LOCATE 3,5:PRINT CHR$(a)+CHR$(a+1)+C
HR$(a+2):LOCATE 3,6:PRINT CHR$(a+3)+CHR$
(a+4)+CHR$(a+5)
900 a$=INKEY$:a$=""
910 IF INKEY(47)=0 THEN ge(2)=a:GOTO 940
920 IF a<230 THEN a=a+6 ELSE a=200
930 GOTO 890
940 FOR lo=1 TO 100:NEXT
950 a=224:PEN 6
960 LOCATE 16,5:PRINT CHR$(a)+CHR$(a+1)+
CHR$(a+2):LOCATE 16,6:PRINT CHR$(a+3)+CH
R$(a+4)+CHR$(a+5)
970 a$=INKEY$:a$=""
980 IF INKEY(47)=0 THEN ge(3)=a:GOTO 125
0
990 IF a<230 THEN A=A+6 ELSE a=200
1000 GOTO 960
1010 REM****KREDIT ABFRAGE****
1020 FOR loop =1 TO 500
1030 IF INKEY$="" THEN NEXT
1040 FOR 1=1 TO 50:a$=INKEY$:a$="":NEXT:
cred=cred-1:PEN 12:LOCATE 8,12:PRINT USI
NG "##.##";cred;:PRINT"$":GOTO 730
1050 IF cred<1 THEN LOCATE 6,12:PEN 7:PR
INT"No Credit!":SOUND 128,0,0,0:SOUND 1,
20,100,5,1:SOUND 4,60,100,5,1:SOUND 2,14
0,100,5,1:GOTO 1370
1060 IF cred>=99.9 THEN cred=50
1070  PEN 12:LOCATE 8,12:PRINT USING "##
.##";cred;:PRINT"$"
1080 IF bonus>=10 THEN LOCATE 14,24:PEN
8:PRINT"CHANCE":cha=1:bonus=1:ENT -1,5,-
1,1:SOUND 1,200,200,5,0,1:ELSE FOR t=1 T
O 9: LOCATE 16,t+14:PEN 1:PRINT t:NEXT:L
OCATE 16,bonus+14:PEN 8:PRINT bonus
1090 IF cha=1 THEN BORDER 22,15:ELSE BOR
DER 0
1100 REM****COMBINATION****
1110 FOR j=1 TO 3
1120 zu=INT(RND*7):IF zu=1 THEN co(j)=20
0
1130 IF zu=2 THEN co(j)=206
1140 IF zu=3 THEN co(j)=212
1150 IF zu=4 THEN co(j)=218
1160 IF zu=5 THEN co(j)=224
1170 IF zu=6 THEN co(j)=230
1180 NEXT
1190 PEN 9:LOCATE 2,23:PRINT CHR$(co(2))
+CHR$(co(2)+1)+CHR$(co(2)+2):LOCATE 2,24
:PRINT CHR$(co(2)+3)+CHR$(co(2)+4)+CHR$(
co(2)+5)
1200 PEN 2:LOCATE 6,23:PRINT CHR$(co(1))
+CHR$(co(1)+1)+CHR$(co(1)+2):LOCATE 6,24
:PRINT CHR$(co(1)+3)+CHR$(co(1)+4)+CHR$(
co(1)+5)
1210 PEN 6:LOCATE 10,23:PRINT CHR$(co(3)
)+CHR$(co(3)+1)+CHR$(co(3)+2):LOCATE 10,
24:PRINT CHR$(co(3)+3)+CHR$(co(3)+4)+CHR
$(co(3)+5)
1220 IF cha=1 THEN LOCATE 16,12:PEN 7:PR
INT CHR$(245)+CHR$(246) ELSE LOCATE 16,1
2:PEN 0:PRINT"  "
1230 FOR 1=1 TO 50:a$=INKEY$:a$="":NEXT
1240 GOTO 1010
1250 REM****RISIKO****
1260 IF zuf=5 THEN ENT -1,5,-1,1:SOUND 1
,200,200,5,0,1:FOR x=1 TO 200:NEXT:GOSUB
 1400
1270 REM****AUSWERTUNG****
1280 IF cha=1 THEN kro=50:glei=25:aukro=
12.5:auglei=10:nebglei=2.5:LOCATE 14,24:
PEN 1:PRINT"CHANCE" ELSE kro=10:glei=5:a
ukro=2.5:auglei=2:nebglei=0.5
1290 IF ge(1)=co(1) AND ge(2)=co(2) AND
ge(3)=co(3) THEN cred=(2*cred):cha=0:SOU
ND 1,25,100,5,1:bonus=bonus+1:GOTO 1050
1300 IF ge(1)=200 AND ge(2)=200 AND ge(3
)=200 THEN SOUND 1,25,100,5,1:cha=0:cred
=cred+kro:bonus=bonus+2:GOTO 1050
1310 IF ge(1)=ge(2) AND ge(2)=ge(3) THEN
 SOUND 1,25,100,5,1:cha=0:cred=cred+glei
:bonus=bonus+1:GOTO 1050
1320 IF ge(1)=200 THEN bonus=bonus+1:SOU
ND 1,25,100,5,1
1330 IF ge(2)=200 AND ge(3)=200 THEN SOU
ND 1,25,100,5,1:cred=cred+aukro:cha=0:GO
TO 1050
1340 IF ge(2)=ge(3) THEN SOUND 1,25,100,
5,1:cred=cred+auglei:cha=0:GOTO 1050
1350 IF ge(2)=ge(1) OR ge(1)=ge(3) THEN
SOUND 1,25,100,5,1:cha=0:cred=cred+nebgl
ei:GOTO 1050
1360 GOTO 1050
1370 IF INKEY$="" THEN 1370
1380 RUN
1390 REM****RISIKO****
1400 con=2:LOCATE 5,17:PEN 8:PRINT"RISIK
O"
1410 FOR con=0 TO 8 STEP 2
1420  IF INKEY(47)=0 THEN 1440
1430 NEXT:GOTO 1410
1440 LOCATE con+2,19:PEN 3:PRINT con/2:c
red=cred+(1.5*(con/2)):FOR hi=1 TO 700:N
EXT:LOCATE con+2,19:PEN 6:PRINT con/2:LO
CATE 5,17:PEN 2:PRINT"RISIKO"
1450 RETURN
```

Listing. Ein Glücksspielautomat, der kein Geld schluckt

Wyatt Earp jagt Billy the Kid

Glück und Geschicklichkeit gehören dazu, Billy the Kid im Wilden Westen aufzuspüren. Die Reisekasse ist knapp und so sicher wie heute kommt man nicht ans Ziel. Hat Wyatt Earp überhaupt eine Chance?

Gemütlich sein Pfeifchen rauchend, sitzt Ex-Marshal Wyatt Earp auf der Veranda seines Hauses in Rattle Oak. Inzwischen auf das Altenteil geschoben, verbleibt ihm nur noch die Erinnerung an frühere Heldentaten. Da fällt ihm beim Durchblättern der »Rattle-Oak-Post« eine Notiz auf Seite 3 ins Auge: »Billy the Kid wurde im 560 Meilen entfernten Kempstown gesehen...«. Wyatt beschließt, auch als Pensionär noch einmal zuzuschlagen. Er plündert sein Sparschwein — viel gibt die schmale Rente nicht her — und macht sich auf die Suche nach dem gefürchteten Gangster. Da die Bandscheiben nicht mehr ganz mitmachen, muß er sich dabei auf das Dampfroß schwingen.

Nach einer atemberaubenden Jagd quer über den ganzen Kontinent stellt der Alt-Marshal seinen Kontrahenten im Saloon von Iron City. Billy hat die »Smith & Wesson« schon in der Hand, Wyatt greift zu seinem altgedienten »Peacemaker«.

In dieser Art kann man sich das Spielgeschehen vorstellen. Nachdem das Programm mit dem Titelbild gestartet ist, wird der Schwierigkeitsgrad eingegeben. Hierbei ist Level 3 mehr zum Üben und Level 1 für Experten gedacht. Anschließend beginnt der Reiseteil mit der Karte von Nordamerika. Die Reiseziele werden durch einfachen Tastendruck abgerufen, es sei denn, man entschließt sich zu einer abwartenden Haltung, woraufhin jedoch Billy the Kid weiter und unbekümmert seines Weges geht. Bei der Wahl des Zieles ist zu bedenken, daß Billy während der Reisedauer eine ähnlich große Strecke wie sein Verfolger zurücklegt und nur selten auf seinen Verfolger wartet. Mit höherem Schwierigkeitsgrad wird Billy cleverer und auch — wie später im Duell — schneller. Bei längeren Reisen können die einzelnen Stationen von Billy verfolgt werden.

Nun kann man im Wilden Westen nicht erwarten, immer dort anzukommen, wohin man will. Widrige Umstände können zur Umkehr zwingen — Fahrtkostenerstattung gibt es nicht. Auch ist es möglich, daß Billy untertaucht und sein Aufenthaltsort nicht auszumachen ist. Oder er flieht nach Mexico — eine für den Marshal unerreichbare Domäne der Desperados. Die Kasse sollte immer im Auge behalten werden, denn Reisen kostet Geld und ist das Geld alle, ist man zu untätigem Herumsitzen verurteilt. Aber vielleicht läuft Billy seinem Verfolger ja in die Arme.

Ist dies schließlich der Fall, gilt es, durch ein Saloonfenster hindurch den Gangster (und nicht etwa einen Unschuldigen) zu erledigen. Bald jedoch ist der Überraschungseffekt dahin und Billy schießt zurück. Zum Glück ist nach fünf doppelten Whisky seine Treffsicherheit nicht allzu hoch.

Dieser Programmteil kann übrigens sowohl mit den Cursortasten wie auch mit dem Joystick gespielt werden. Die Aufenthaltsorte der beiden Kontrahenten werden während der Initialisierung per Zufall ausgesucht. Anschließend wird im Unterprogramm ab Zeile 670 die Nordamerika-Karte aufgebaut und ab Zeile 800 die Titelsequenz gespielt. Durch einen Sprung zurück wird die Karte erneut aufgebaut und die notwendigen Informationen ausgegeben. Erreicht während der Tastaturabfrage in den Zeilen 150 bis 170 die Zählvariable »pwa« den Wert 600, so springt das Programm nach Zeile 400 und Billy wandert einen Ort weiter.

Durch Drücken einer entsprechenden Taste werden ab Zeile 180 der Zielort, die Distanz und die Reisekosten ermittelt. Diese gibt das Programm ab Zeile 250 aus und der Spieler akzeptiert oder lehnt ab. Ab Zeile 400 wird die Reisedistanz in Teilstationen von Billys Reiseroute zerlegt.

Die Zugreise beginnt in Zeile 470; bei Reiseunterbrechungen arbeitet das Programm die Zeilen 900 bis 950 ab. Die Teilstationen von Billys Reise werden im Unterprogramm ab Zeile 1030 ausgegeben.

Ab Zeile 1110 beginnt der zweite Programmteil mit der Definition neuer Symbole. Ist dies einmal geschehen, kann das Duell durch »GOTO 1250« separat abgerufen werden (vorher Variable lev (1 bis 3) eingeben). Ab Zeile 1250 wird zuerst das Bild ausgegeben und dann schlagartig sichtbar gemacht. In Zeile 1430 ist die Aufenthalts-, beziehungsweise Abwesenheitsdauer der »Hüte« festgelegt. Die Zeilen 1440 bis 1560 dienen der Joystickabfrage.

Aus diesem Bereich ruft das Programm bei »Feuer« das Unterprogramm ab Zeile 1720, zum »Hutwechsel« das Unterprogramm ab Zeile 1570 auf, und von dort aus gelegentlich zur Feuererwiderung das Unterprogramm ab Zeile 1660. Bei Erfolg wird entsprechend zu der Zeile 1800 oder zu der Zeile 1840 verzweigt.

(Olav Panckow)

Variablenliste

lev	— Schwierigkeitsgrad
n$(i)	— 27 Spielorte
x(i),y(i)	— Koordinaten der Spielorte
w	— Kennziffer des Aufenthaltsortes von Wyatt
ww	— Kennziffer des Reisezieles von Wyatt
zx(n),zy(n)	— Koordinaten der Reisestationen von Wyatt
b(z)	— Kennziffer der Teil-Stationen von Billy
bv	— Billy sichtbar oder verschollen?
do	— Reisekasse
k	— Reisekosten
dw	— Reisestrecke Wyatt
pa	— Pausenvariable
pwa	— Zählvariable
x,y	— Koordinaten des Fadenkreuzes
bx,by	— Koordinaten des »Hutes«
z,ba,bav	— Zeitvariablen
fla	— »Hut« sichtbar oder nicht sichtbar?
flu	— »Hut« von einem Unschuldigen oder von Billy?

```
10  REM   ****************************
20  REM   *      - WYATT EARP -      *
30  REM   *   von Olav Panckow      *
40  REM   *       Vuerfels 57 a     *
50  REM   *  5060 Berg.-Gladbach 1  *
60  REM   *     Tel. 02204/69066    *
70  REM   ****************************
80  REM
90  REM   >> Initialisierung <<
100 ENV 1,1,0,25,5,-1,1:ENV 2,12,1,1,1,0
,166,10,-1,1:ENV 3,3,1,1,1,0,4,3,-1,1:EN
V 4,1,1,1,1,0,16,14,-1,1:ENT 1,1,0,30,10
,1,1,1,0,100,10,-1,1
110 RANDOMIZE TIME:CLS:GOSUB 1190
120 SPEED INK 15,15:ON BREAK GOSUB 2220
130 DIM n$(27):DIM x(27):DIM y(27):DIM z
x(30):DIM zy(30):DIM b(10)
```



```
140 FOR i= 1 TO 27:READ n$(i),x(i),y(i):
NEXT
150 w$="WYATT EARP ":b$="BILLY THE KID "
160 w=INT(RND*23+1):b(1)=INT(RND*27+1):w
1=w
170 IF ABS(w-b(1))<10 GOTO 160
180 REM   >> Reiseteil <<
190 GOSUB 810:IF t=0 GOTO 950
200 LOCATE x(w)+1,y(w):PRINT "<"
210 LOCATE#1,5,1:PRINT#1,w$"ist in ";n$(
w):LOCATE#1,45,1
220 IF bv<1 THEN PRINT#1,b$"ist in "n$(b
(1)):MOVE x(b(1))*8-12,(26-y(b(1)))*16-2
4:DRAWR 16,0,1:DRAWR 0,32:DRAWR -16,0:DR
AWR 0,-32 ELSE PRINT#1,b$"ist verscholle
n ..."
230 IF t>0 THEN LOCATE 70,23:PRINT "LEVE
L";lev:IF sch=1 GOTO 1310
240 LOCATE #2,5,1:PRINT#2,"  Wohin geht'
s ? -  TASTE des Zielortes druecken   (
oder WARTEN ... )"
250 e$=UPPER$(INKEY$):IF e$<>"" GOTO 280
260 pwa=pwa+1:IF pwa=600 THEN PRINT CHR$
(7):CLS#1 :CLS#2 :MOVE x(b(1))*8-12,   (
26-y(b(1)))*16-24: DRAWR 16,0,0:DRAWR 0,
32:DRAWR -16,0:DRAWR 0,-32:ww=0:wa=1:dw=
480:GOTO 520
270 GOTO 250
280 FOR ww=1 TO 27:  IF e$=LEFT$(n$(ww),1
) THEN IF ww<24 AND w<>ww THEN 360 ELSE
 GOTO 300
290 NEXT ww
300 SOUND 4,1000,50,5,1:CLS#2:LOCATE #2,
20,1
310 IF w=ww THEN PRINT#2,"Da ist "w$"doc
h gerade !!":GOTO 340
320 IF ww<28 THEN PRINT#2,"Das geht nich
t! - "n$(ww)" liegt in MEXICO!!"
330 IF ww=28 THEN LOCATE#2,30,1:PRINT#2,
"Wohin soll's gehen ???"
340 FOR i=1 TO 2000:NEXT:GOTO 240
350 REM   >> Reisekosten <<
360 INK 0,24:INK 1,1:INK 2,8,17:INK 3,1,
24:MODE 1:PAPER 0:PEN 1:SOUND 2,239,40,6
,1:SOUND 2,159,80,6,1
370 d=SQR((x(w)-x(ww))^2+(y(w)-y(ww))^2*
4):dw=INT(d*46.6):k=ROUND(dw/12,0)
380 IF bv<1 THEN LOCATE 5,3:PRINT b$"ist
 in "n$(b(1))
390 PEN 2:LOCATE 5,5:PRINT "REISE-KASSE
: ";USING "######";do:LOCATE 27,5:PRINT
"$"
400 PEN 1:LOCATE 5,7:PRINT "REISESTRECKE
: "USING "######";me:LOCATE 27,7:PRINT "
Meilen"
410 LOCATE 5,10:PRINT "Die Reise von ";:
PEN 2:PRINT n$(w):PEN 1:LOCATE 5,12:PRIN
T "nach ";:PEN 2:PRINT n$(ww)
420 PEN 1:LOCATE 5,14:PRINT "wird ueber
weitere"dw"Meilen":LOCATE 5,16:PRINT "ge
hen und";:PEN 2:PRINT k;"$ ";:PEN 1:PRIN
T "kosten."
430 ORIGIN 464,16:v=0.25:GOSUB 870
440 PLOT x(w)*2-1,(26-y(w))*4-2,3:PLOT x
(ww)*2-1,(26-y(ww))*4-2,3:IF bv<1 THEN P
LOT x(b(1))*2-1,(26-y(b(1)))*4-2,2
450 LOCATE 5,21:IF do-k<0 THEN SOUND 2,1
000,200,0,2:PEN 3:PRINT "LEIDER ZU TEUER
 !!": FOR pa=1 TO  4000:NEXT pa:GOTO 190
460 PRINT "Einverstanden ? (J/N)"
470 e$=UPPER$(INKEY$):IF e$="J" THEN PRI
NT CHR$(7):GOTO 490
480 IF e$="N" GOTO 190 ELSE GOTO 470
490 me=me+dw:do=do-k:GOSUB 810
500 LOCATE#2,20,1:PRINT#2,w$"reist nach
"n$(ww)
510 REM   >> Reise Billy <<
520 z=1:bv=0:dwm=dw*(15-lev)/10:FOR i=1
TO 25
530 b(z+1)=INT(RND*27+1):FOR tt=1 TO z:I
F b(z+1)=b(tt) GOTO 570
540 NEXT tt
550 d=SQR((x(b(z))-x(b(z+1)))^2+(y(b(z))
-y(b(z+1)))^2*4):db=INT(d*46.6)
560 IF dwm>=db THEN IF NOT(w=b(z+1) OR w
w=b(z+1))OR RND>(8-lev)/10 THEN dwm=dwm-
db:z=z+1:IF wa>0 GOTO 770
570 NEXT i
580 IF wa>0 GOTO 770
590 REM   >> Zugreise <<
600 EVERY 15,0 GOSUB 1160:fl=0
610 x=x(w)*8-4:y=(26-y(w))*16-8
620 xx=x(ww)*8-4:yy=(26-y(ww))*16-8
630 dx=xx-x:dy=yy-y
640 IF ABS(dx)<ABS(dy) THEN dxx=dx:dx=dy
:dy=dxx:fl=1
650 m=dy/dx:FOR i=0 TO ABS(dx)
660 FOR pa=1 TO 100:NEXT pa
670 IF i MOD 3 <>0 GOTO 710
680 ii=i*SGN(dx)
690 IF fl<1 THEN px=x+ii:py=y+m*ii:PLOT
px,py,1:GOTO 710
700 px=x+m*ii:py=y+ii:PLOT px,py,1
710 FOR j=2 TO z
720 IF i=INT(ABS(dx)*(j-1)/(z-1)) THEN G
OSUB 1210
730 NEXT j
740 IF i=INT(ABS(dx)*0.6) AND RND<0.4 AN
D sp<1 THEN re=REMAIN(0):CLS#2:LOCATE#2,
20,1:sp=INT (RND*6+1):ON sp GOTO 1060,10
70,1080,1090,1100,1110
750 NEXT i:sp=0
760 st=st+1:zx(st)=x(ww)*8-4:zy(st)=(26-
y(ww))*16-8:IF zu<1 THEN w=ww ELSE st=st
-1
770 re=REMAIN(0):zu=0:b(1)=b(z):pwa=0:IF
 w=b(1) THEN bv=0:sch=1
780 IF wa>0 THEN wa=0:GOTO 210
790 GOTO 190
800 REM   >> Nordamerika - Karte <<
810 INK 0,6:INK 1,23:BORDER 1:PEN 1:PAPE
R 0:MODE 2:v=1
820 MOVE x(w1)*8-4,(26-y(w1))*16-8
830 FOR i=1 TO st:DRAW zx(i),zy(i),1:NEX
T
840 PRINT CHR$(24):FOR i=1 TO 27
850 LOCATE x(i),y(i):PRINT LEFT$(n$(i),1
)
860 NEXT i:PRINT CHR$(24):PLOT 0,0,1
870 RESTORE 930:FOR i=1 TO 26:READ a,b,c
:IF a=0 THEN PLOT b*v,c*v ELSE DRAW b*v,
c*v
880 NEXT i:IF v<1 THEN RETURN
890 WINDOW #1,1,80,1,1:PEN #1,0:PAPER #1
,1:CLS#1
900 WINDOW #2,1,80,25,25:PEN #2,0:PAPER
#2,1:CLS#2:RETURN
910 DATA "MIAMI",61,21,"UNDERDOWNS",61,1
8,"ATLANTA",56,15,"NEW ORLEANS",52,20,"H
OUSTON",48,20,"OKLAHOMA",39,16,"@$##%&'@
",45,14,"ST.LOUIS",50,11,"IRON CITY",57,
11,"WASHINGTON",64,12,"YORK",65,10,"DETR
```

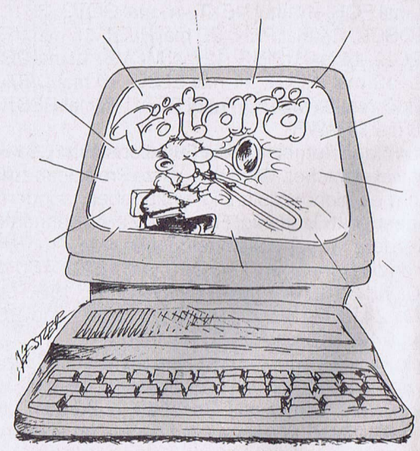
Listing. Kann Wyatt Earp Billy the Kid festsetzen?

```
OIT",58,6,"CHICAGO",54,7,"TOTAL SPRINGS"
,42,6
920 DATA "VALLEY BLUE",37,10,"RATTLE OAK
",30,13,"BIG BALL",24,10,"KEMPSTOWN",29,
7,"GREAT FALLS",21,4,"EASY POINT",15,8,"
FRISCO",10,11,"LOS ANGELES",12,14,"PHOEN
IX",19,15,"ZIJUANA",14,17,"JUAREZ",22,18
,"XANADU",34,19,"QUATL",26,21
930 DATA 0,380,364,1,110,364,1,80,304,1,
70,224,1,100,134,1,110,74,1,120,64,1,117
,134,1,126,143,1,141,47,1,363,47,1,372,7
4,1,459,88,1,468,62,1,480,59,1,492,71,1,
480,98,1,525,176,1,516,203,1,549,350,1,5
28,353,1,474,320,1,381,364,0,97,152,1,22
7,132,1,370,74
940 REM    >> Titel - Sequenz <<
950 LOCATE #1,30,1:PRINT#1,"DER WILDE WE
STEN ......":LOCATE #2,20,1:PRINT#2,"DAS
 GESETZ KAEMPFT GEGEN DIE GESETZLOSEN"
960 TAG:FOR i=0 TO 213 STEP 3:MOVE i,255
:PRINT " "w$;
970 MOVE i+60,222:PRINT " jagd";:MOVE i+
80,191:PRINT " "b$;
980 SOUND 1,500-i,10,6+(i/42.6),3:SOUND
4,600-i,10,6+(i/42.6),3:NEXT:TAGOFF:t=1
990 FOR i=1 TO 20:SOUND 1,486,10,10,3,0,
2:SOUND 4,0,10,10,3,0,3:FOR pa=1 TO 100:
NEXT pa:NEXT i
1000 LOCATE 1,24:PRINT CHR$(24);"
            WIE STARK SOLL BILLY THE K
ID SEIN ?                    ";CHR$(
24)
1010 CLS#2: LOCATE#2,22,1:PRINT#2,">>>>
 1- , 2- oder 3-klassig ? <<<<":EVERY 40
,0 GOSUB 1180
1020 in$=INKEY$:IF in$="" GOTO 1020
1030 IF VAL(in$)<1 OR VAL(in$)>3 GOTO 10
10 ELSE re=REMAIN(0):lev=VAL(in$)
1040 do=800-lev*150:GOSUB 1190:GOTO 190
1050 REM    >> Reiseunterbrechung <<
1060 PRINT #2,"BUEFFEL BLOCKIEREN DIE GL
EISE ...":GOTO 1120
1070 PRINT #2,"BANDITEN UEBERFALLEN DEN
ZUG !!!":GOTO 1120
1080 PRINT #2,"DER DAMPKESSEL DER LOK DR
OHT ZU PLATZEN ...":GOTO 1120
1090 PRINT #2,"DER ZUGFUEHRER IST ZU BES
OFFEN  ( #&@!! )":GOTO 1120
1100 PRINT #2,"INDIANER REITEN IM KRIEGS
SCHMUCK HERAN !":GOTO 1120
1110 PRINT #2,"EIN WIRBELSTURM SAUST HER
AN ...":GOTO 1120
1120 FOR so=1 TO 500:SOUND 4,RND*500,1,4
:NEXT so
1130 CLS#2:LOCATE#2,20,1: IF RND<0.5 THE
N EVERY 15,0 GOSUB 1160:PRINT#2, "GLUECK
 GEHABT - es geht weiter !":FOR pa=1 TO
1000:NEXT pa:zu=0:GOTO 750
1140 PRINT#2,"Pech gehabt - "w$"muss zur
ueck !!":SOUND 2,1000,50,4,1:FOR pa=1 TO
 1500:NEXT pa:zu=1:GOTO 760
1150 REM    >> Div. Sounds <<
1160 SOUND 2,0,30,12,1,0,1:IF RND<0.1 TH
EN SOUND 1,478,80,6
1170 RETURN
1180 SOUND 2,150,5:RETURN
1190 SOUND 1,159,190,1,2,1:SOUND 4,165,1
90,1,2,1:RETURN
1200 REM    >> Stationen Billy <<
1210 IF RND<0.33 AND j=z THEN bv=1:RETUR
N
1220 re=REMAIN(0):PRINT CHR$(7):PLOT x(b
(j))*8-12,(26-y(b(j)))*16-24:DRAWR 16,0,
1:DRAWR 0,32:DRAWR -16,0:DRAWR 0,-32
1230 LOCATE#1,20,1:PRINT#1,b$"wurde in "
n$(b(j))" gesehen"
1240 FOR pa=1 TO 1750+250*lev:NEXT pa:CL
S#1
1250 PLOT x(b(j))*8-12,(26-y(b(j)))*16-2
4:DRAWR 16,0,0:DRAWR 0,32:DRAWR -16,0:DR
AWR 0,-32
1260 EVERY 15,0 GOSUB 1160:RETURN
1270 REM    >> Melodie <<
1280 RESTORE 1290:FOR i=1 TO 6:READ th,t
d:SOUND 1,th,td,0,2:SOUND 4,th+10,td,0,2
:NEXT i:RETURN
1290 DATA 478,100,426,40,402,60,478,40,3
38,160,319,200
1300 REM    >> Uebergang Duell <<
1310 LOCATE#2,4,1:PRINT#2,w$"brauchte ";
USING"###";800-lev*150-do;:PRINT#2," $,
um nach ";USING "####";me;:PRINT#2," Mei
len "b$"zu stellen ..."
1320 GOSUB 1280:FOR pa=1 TO 6000:NEXT pa
1330 BORDER 0:INK 0,0:INK 1,15:INK 2,15,
24:PAPER 0:PEN 1:MODE 1:GOSUB 1190
1340 SYMBOL AFTER 250
1350 SYMBOL 251,&66,&E7,&0,&E7,&0,&E7,&0
,&E7
1360 SYMBOL 252,&81,&5A,&24,&42,&42,&24,
&5A,&81
1370 SYMBOL 253,&0,&18,&3C,&34,&24,&3C,&
7E,&0
1380 SYMBOL 254,&0,&0,&18,&3C,&3C,&3C,&7
E,&0
1390 SYMBOL 255,&0,&18,&18,&3C,&3C,&18,&
18,&18
1400 LOCATE 3,6:PRINT b$"hat sich im":PR
INT"  Saloon von "n$(w)" verschanzt."
1410 PRINT:PRINT "  Schemenhaft taucht e
r":PRINT "  dann und wann hinter den":PR
INT "  staubigen Fenstern des verruchten
":PRINT "  Lokals auf ..."
1420 PRINT:PRINT "  Man erkennt ihn nur
an seinem":PRINT "  durchloecherten Hut
>"CHR$(24)" "CHR$(253)" "CHR$(24)"<."
1430 PEN 2:PRINT:PRINT:PRINT "  "w$"drue
ckt eine TASTE":PRINT "  und zieht den C
olt ...."
1440 WHILE INKEY$="":WEND
1450 REM    >> Duell - Screen <<
1460 MODE 0:BORDER 0:FOR i=0 TO 5:INK i,
0:NEXT i:PAPER 0:CLS:GOSUB 1280
1470 PEN 5:LOCATE 3,17:PRINT CHR$(213):L
OCATE 3,16:PRINT CHR$(143):LOCATE 3,15:P
RINT CHR$(214)
1480 FOR i=0 TO 3:LOCATE 5+i,9-i:PRINT C
HR$(214):NEXT i:LOCATE 6,9:PRINT CHR$(14
3)
1490 FOR i=5 TO 6:FOR n=10 TO 14:LOCATE
i,n:PRINT CHR$(143): NEXT n,i
1500 FOR i=8 TO 17:FOR n=0 TO 1:LOCATE i
+n,7-n:PRINT CHR$(143):NEXT n,i
1510 LOCATE 18,7:PRINT CHR$(212):LOCATE
19,6:PRINT CHR$(212):PEN 2:LOCATE 2,2:PR
INT "Level";lev
1520 PAPER 3:FOR i=7 TO 17:FOR n=8 TO 14
:LOCATE i,n:PRINT " ":NEXT n,i
1530 FOR i=4 TO 17:FOR n=15 TO 17:LOCATE
 i,n:PRINT " ":NEXT n,i
1540 PAPER 4:FOR i=1 TO 20:FOR n=18 TO 2
3:LOCATE i,n:PRINT " ":NEXT n,i
1550 PEN 1:PAPER 2:LOCATE 11,5:PRINT "SA
LOON"
```



```
1560 PAPER 1:FOR i=0 TO 4:FOR n=0 TO 2:L
OCATE 8+2*i,16-3*n:PRINT " ":NEXT n,i
1570 WINDOW#1,1,20,24,25:PAPER#1,1:PEN#1
,2
1580 PLOT 190, 126, 1:DRAW 190, 288:DRAW
 544, 288:DRAW 544, 126:PLOT 638, 126:DR
AW 0, 126:PLOT 188, 176:DRAW 94, 176:DRA
W 62, 160:DRAW 62, 142:DRAW 92, 126:DRAW
 92, 176
1590 PLOT 120, 126:DRAW 120, 158:DRAW 16
4, 126:DRAW 164, 158:DRAW 126, 158:PLOT
126, 176:DRAW 126, 256:DRAW 252, 320:DRA
W 612, 320:DRAW 546, 288:PLOT 512, 320:D
RAW 512, 336:DRAW 318, 336:DRAW 318, 320
1600 PLOT 192, 244:DRAW 256, 320:PLOT 54
4, 244:DRAW 608, 320:PLOT 127, 256:DRAW
127, 264:DRAW 256, 336:DRAW 610, 336:DRA
W 610, 320
1610 FOR i=1 TO 4:PLOT 127+26*i+7*(i-1),
257+13*i+3*(i-1):DRAWR 0,9+i*2:NEXT i
1620 REM    >> Duell - Action <<
1630 INK 0,23:INK 1,0:INK 2,15:INK 3,12:
INK 4,9:INK 5,4
1640 EVERY 300,0 GOSUB 1190:GOSUB 2210:z
=0:flu=0:fla=0:x=12:y=13
1650 ba=INT(8+RND*8):bav=0
1660 xx=x:yy=y
1670 z=z+1:bav=bav+1:IF bav=ba GOTO 1800
1680 IF z MOD 3=0 THEN SOUND 2,100,1
1690 IF INKEY(9)=0 OR INKEY(76)=0 THEN G
OTO 1970
1700 x=x-(INKEY(1)=0)-(INKEY(75)=0)+(INK
EY(8)=0)+(INKEY(74)=0)
1710 y=y-(INKEY(2)=0)-(INKEY(73)=0)+(INK
EY(0)=0)+(INKEY(72)=0)
1720 IF x<7 OR x>17 OR y<8 OR y>17 THEN
x=xx:y=yy
1730 fa=TEST (xx*32-16,(26-yy)*16-1):PEN
 fa
1740 LOCATE xx,yy:PRINT CHR$(22)+CHR$(1)
;CHR$(252)
1750 PEN 0:LOCATE x,y:PRINT CHR$(252):CH
R$(22)+CHR$(0)
1760 PAPER 1:PEN 0:LOCATE 12,17:PRINT CH
R$(251)
1770 IF fla>0 THEN LOCATE xb,yb:IF flu<1
 THEN PRINT CHR$(253) ELSE PRINT CHR$(25
4)
1780 FOR pa=1 TO (lev-1)*20:NEXT pa:GOTO
 1660
1790 REM    >> Hut - Wechsel <<
1800 IF fla>0 GOTO 1880
1810 bx=INT(RND*5):by=INT(RND*3)
1820 xb=bx*2+8:yb=16-(by*3):fla=1
1830 IF xb=x AND yb=y GOTO 1810
1840 IF RND <0.5 THEN flu=1:GOTO 1860 EL
SE flu=0
1850 IF RND <0.1+0.1*lev AND z>125-15*le
v GOTO 1900
1860 PAPER 1:PEN 0:LOCATE xb,yb:IF flu<1
 THEN PRINT CHR$(253) ELSE PRINT CHR$(25
4)
1870 GOTO 1650
1880 PAPER 1:LOCATE xb,yb:PRINT " ":fla=
0:GOTO 1650
1890 REM    >> Schuss Billy <<
1900 CLS#1:PAPER 1:PEN 0
1910 FOR i=1 TO 5:SOUND 2,0,10,10,3,0,2
1920 FOR n=1 TO 2:LOCATE xb,yb:PRINT CHR
$(24);CHR$(253):pa=100^2:NEXT n
1930 FOR pa=1 TO 70:NEXT pa,i
1940 LOCATE#1,2,2:IF RND<0.6+0.05*lev TH
EN PRINT#1," BILLY VERFEHLTE.." ELSE PRI
NT#1,">>> BILLY TRAF <<<":GOTO 2060
1950 FOR pa=1 TO 1000:NEXT pa:GOSUB 2210
:GOTO 1880
1960 REM    >> Schuss Wyatt <<
1970 SOUND 2,250,33,14,4,0,6:CLS#1:LOCAT
E#1,4,2
1980 IF x<>xb OR y<>yb THEN PRINT#1,"AMO
KSCHUETZE!":GOTO 2040
1990 IF fla<1 GOTO 2040
2000 fa=TEST (x*32-16,(26-y)*16-1)
2010 PAPER fa:LOCATE x,y:PRINT " ":LOCAT
E#2,2,2
2020 IF flu>0 THEN PRINT#1,"   MOERDER !
!":PRINT #1,"Das war nicht Billy!";:GOTO
 2040
2030 IF RND<0.5 THEN PRINT#1,"Knapp verf
ehlt!" ELSE GOTO 2110
2040 FOR pa=1 TO 1000:NEXT pa:GOSUB 2210
:GOTO 1800
2050 REM    >> Billy Sieger <<
2060 re=REMAIN(0):PAPER 1
2070 FOR i=1 TO 23:LOCATE 1,i:PRINT "
                "
2080 SOUND 1,500+i*40,70,14,1:SOUND 4,30
0+i*40,70,14,1:FOR pa=1 TO 500
2090 NEXT pa,i:GOTO 2170
2100 REM    >> Wyatt Sieger <<
2110 re=REMAIN(0):PAPER 0:PEN 1
2120 LOCATE x,y:PRINT CHR$(255)
2130 LOCATE 2,20:PRINT " WYATT HAT ALLES
 ":LOCATE 2,21:PRINT "   KLAR GEMACHT  "
2140 LOCATE#1,5,2:PRINT#1,"Glueckwunsch!
"
2150 FOR i=1 TO 6:FOR n=0 TO 26:BORDER n
2160 SOUND 1,400-n*13,5:SOUND 4,500-n*13
,5:NEXT n,i
2170 CLS#1:LOCATE#1,5,2:PRINT#1,"NEU? -
J / N"
2180 e$=UPPER$(INKEY$):IF e$="J" THEN RU
N
2190 IF e$="N" GOTO 2220 ELSE GOTO 2180
2200 REM    >> Unterprogramme <<
2210 CLS#1:LOCATE#1,4,1:PRINT#1,"CURSOR:
 Zielen":LOCATE#1,5,2:PRINT#1,"COPY : Sc
huss":RETURN
2220 INK 0,1:INK 1,20:BORDER 1:PAPER 0:P
EN 1:MODE 1
2230 LOCATE 12,12:PRINT "GOOD BYE, WYATT
 !!":FOR pa=1 TO 2000:NEXT pa:MODE 2:END
```

Listing. »Wyatt Earp jagt Billy the Kid« (Schluß)



Kopfüber in die Aktien

Wer hat nicht schon einmal davon geträumt, an der Börse zu spekulieren? Mit Ihrem Schneider riskieren Sie dabei kein Geld — aber spüren trotzdem die Atmosphäre.

Mit »Börse 79« kommt das Geschehen an der Börse zu Ihnen nach Hause. Bis zu vier Personen können gleichzeitig spielen. Ein Partner kann vom Computer übernommen werden.

Das Programm ist menügesteuert. Mit »w« für weiter und »z« für zurück werden die einzelnen Optionen ausgewählt, während man mit »h« für höher und »n« für niedriger die Firma, das Limit und die Stückzahl verändert. Ein Limit ist nötig, weil man immer für die nächste Runde bietet. Man muß sich also vor dem Kauf über die Option »Kurve« die besten Aktien aussuchen. Man kann beliebig viele Aktionen ausführen, solange Zeit und Geld vorhanden sind. Voreingestellt sind als Zeitlimit 30 Sekunden und als Startkapital 2 000 Dollar. Ist man zahlungsunfähig, dann kann man einen Kredit aufnehmen. Doch hierbei ist Vorsicht geboten: Innerhalb von zehn Runden muß der Kredit mit zehn Prozent Zinsen zurückgezahlt sein, sonst wird das Bargeld gepfändet und falls dies nicht reicht, werden die Aktien des Spielers zu 80 Prozent des Wertes verkauft, bis der Kredit zurückgezahlt ist. Damit der Kredit auch abgesichert ist, darf man höchstens 50 Prozent des eigenen Aktien- und Geldwertes beleihen. Aber auch diese 50 Prozent können am Anfang geändert werden.

Tips zum Spiel: Man sollte das Zeitlimit nicht zu knapp wählen, denn beim Computer wird die Zeit nicht gemessen, was auch witzlos wäre, denn er braucht nur wenige Sekunden für seine Aktionen. Außerdem sollte man in der ersten und in den letzten beiden Runden weder kaufen noch verkaufen, es lohnt sich nicht. Vor jeder vierten Runde sollten möglichst keine Aktien verkauft werden, weil in der vierten Runde die Dividendenauszahlung erfolgt. Einen Kredit sollte man nur dann aufnehmen, wenn man nicht mehr genug Geld hat, um eine Aktie zu kaufen, die mit einem Male wie verrückt steigt.

Noch ein paar Tips zum Eingeben des Programms: Ein großer Vorzug des CPC ist, daß man ziemlich viele Funktionstasten zur Verfügung hat. Deshalb sind hier alle Befehle mit der Anzahl, in der sie vorhanden sind, aufgelistet:
101 mal IF... THEN, 98 mal PRINT, 63 mal LOCATE, 64 mal REM, 44 mal FOR, 44 mal NEXT, 40 mal GOTO, 38 mal CLS, 30 mal GOSUB, 24 mal PEN, 20 mal PLOT, 15 mal RETURN, 14 mal SPC...), 14 mal USING, 12 mal MOVE, 12 mal DRAW, 12 mal CALL, 12 mal INPUT, 11 mal ELSE, 10 mal DATA, 8 mal LOWER$(, 7 mal SYMBOL, 7 mal READ, 7 mal RESTORE, 7 mal INT, 6 mal DRAWR.

Legt man diese Befehle auf die Funktionstasten, so wird das Eingeben viel einfacher. Wem das ganze Programm zu lang ist, der braucht es nicht vollständig einzugeben, sondern es ist auch »abgespeckt« lauffähig. Folgende Zeilen können weggelassen werden, wobei zusammengehörende Programmteile nebeneinanderstehen: 300 bis 320 und 1820, 400 bis 420 und 1520 und 3280 bis 3580, 590 bis 780, 1580 bis 1810 und 1830, 2410 bis 2590, 2690 bis 2770 und 2810 bis 2970 und 940, 910 und 2630 bis 2650.

Noch ein Hinweis zum Schluß: Das Spiel ist für vier Spieler gedacht und sollte auch mit vier Spielern gespielt werden, denn Konkurrenz belebt das Geschäft, außerdem bringt es meistens mehr Spaß, als immer nur alleine zu spielen.

(Henning Roes)

firma$()	= Firmenname
s$()	= Spielernamen
rendite()	= Rendite der Firma
p$,e,f	= wichtig beim Bildschirmaufbau
scroll	= ab hier steht das Maschinenprogramm im Speicher
a,b,f,k,w	= Laufvariablen n,u,s,t,i
sp	= Spieleranzahl
za	= Anzahl der zu spielenden Runden
comp	= ist 1, wenn CPC mitspielt
ak(,)	= Aktienanzahl des Spielers
bonus(,)	= ist >0, wenn der Spieler schon den Bonus hatte (Zeile 1950)
divi()	= Dividende der Aktie
entw(,)	= hier sind alle Kurse enthalten
fi(),fir	= Hilfsvariablen für CPC
k()	= Kursfaktor über längere Zeit
kurs()	= Kurs der Aktie
li(,)	= Limits des Spielers
ma(),mi()	= höchster und niedrigster Kurs
st(,)	= (Ver)Kaufsaufträge des Spielers
we(,1)	= Bargeld des Spielers
we(,2)	= Aktienwert des Spielers
we(3,)	= we(,1)+we(,2)
zeit	= Die Uhr läuft nur, wenn Zeit = 0
ag	= wichtig bei Zeigerdarstellung
zlimit	= Zeitlimit
krepro	= Prozentwert bei Kreditaufnahme
n(),b(),h()	= wichtig bei Änderungen
a$	= INKEY$
num	= Firmennummer bei Änderungen
nam$	= neuer Firmenname
runde,r	= aktuelle Rundennummer
runde(t)	= Rundenzahl bei Kreditaufnahme
g()	= Ausgaben des Spielers
aus	= 1 wenn zlimit überschritten
hoch	= Hürde für bestimmte Variablen
sieg()	= Spielernamen und Vermögen bei
sieg$()	= Siegerehrung
na	= Nachfrage
k1,k2,k3	= Kursfaktoren
mi,ma	= niedrigster und höchster Wert bei Kurvendarstellung
x()	= Firmennummer bei Darstellung
y,v	= wichtig bei Kurvendarstellung
div	= Dividendenauszahlung
kre%	= höchstmöglicher Kredit
kredit%	= Kredithöheneingabe
kre()	= Kredithöhe

Variablenliste

```
10 REM ******************************
20 REM *     DAS B\RSENSPIEL        *
30 REM *   COPYRIGHT 1985   by       *
40 REM *      Henning  Roes          *
50 REM *      Starnberger Str 92     *
60 REM *      2300 Kiel 14           *
70 REM *  TEL : 04 31 / 78 17 19     *
80 REM ******************************
90 REM Zeichensatz
100 REM
110 SYMBOL AFTER 90
120 SYMBOL 91,90,60,102,102,126,102,102,
0
130 SYMBOL 92,186,108,198,198,198,108,56
,0
140 SYMBOL 93,102,0,102,102,102,102,60,0
150 SYMBOL 123,72,0,120,12,124,204,118,0
160 SYMBOL 124,36,0,60,102,102,102,60,0
```



```
170 SYMBOL 125,68,0,102,102,102,102,62,0
180 KEY DEF 22,1,124,92:KEY DEF 19,1,125
,93:KEY DEF 17,1,123,91
190 REM
200 REM SCROLL-PROGRAMM
210 REM
220 CLEAR:scroll=42011:MEMORY scroll-100
0
230 DATA &06,&00,&3e,&0,&21,1,0,&11,12,3
9,&cd,&50,&bc,&c9,0
240 FOR n=scroll TO scroll+14
250 READ u:POKE n,u:NEXT
260 REM
270 REM start
280 REM
290 MODE 1:BORDER 1:PAPER 0:PAPER#1,0:PA
PER#2,0:INK 0,1:INK 1,24:INK 2,14:INK 3,
6:WINDOW#1,3,37,8,23:WINDOW#2,3,37,18,23
:PEN#3,2:PEN#2,2
300 EVERY 10,0 GOSUB 1770:RESTORE 1810:s
oun=1
310 TAG:FOR n=1 TO 430 STEP 10:MOVE 0,12
6+n/1.5:DRAW 275-n/1.5,400,1:MOVE 366+n/
1.5,0:DRAW 640,275-n/1.5,2
320 PLOT n-125,n,2:PRINT"B!rse";:PLOT n-
25,n,2:PRINT"79";:PLOT n+32,n,1:PRINT"19
85";:PLOT n+112,n,2:PRINT"by";:PLOT n+16
0,n,1:PRINT"Henning";:PLOT n+288,n:PRINT
"Roes";:NEXT:TAGOFF:CLS:PEN 0:PRINT REMA
IN(0):soun=0
330 PEN 2:p$="$#":FOR n=0 TO 1:LOCATE 1,
n*22+2:FOR u=1 TO 19:PRINT p$;:NEXT:PRIN
T"$":NEXT
340 FOR n=0 TO 1:FOR u=3 TO 23 STEP 2:LO
CATE n*38+1,u:PRINT"#":LOCATE n*38+1,u+1
:PRINT"$":NEXT u,n
350 PEN 3:LOCATE 17,4:PRINT"B\RSE 79":PE
N 2:LOCATE 7,6:PRINT CHR$(164)"  1985  b
y Henning Roes  "CHR$(164)
360 PEN#1,1:LOCATE#1,1,3:INPUT#1,"Wievie
le Spieler nehmen teil";sp:IF sp>4 OR sp
<1 THEN sp=4
370 LOCATE#1,1,6:PRINT#1,"Bitte geben Si
e Ihren Namen ein":FOR n=1 TO sp:INPUT#1
,"1-5 Buchstaben";s$(n):s$(n)=LEFT$(s$(n
),5):NEXT:CLS#1
380 LOCATE#1,1,3:INPUT#1,"Jber  wieviele
  Runden  wollen Sie spielen (1-100)";za
390 IF za<1 OR za>100 THEN 380
400 LOCATE#1,1,7:IF sp=4 THEN 460
410 INPUT#1,"Wollen Sie auch gegen den C
omputer spielen";a$
420 IF LOWER$(a$)="j" THEN comp=1:sp=sp+
1:s$(sp)="CPC":ELSE 460
430 REM
440 REM initialisierung
450 REM
460 DEFINT e,l,b,n
470 DIM ak(4,20),bonus(4,20),divi(20),en
tw(za+1,20),fi(20),firma$(20),k(20),kurs
(20),li(4,20),ma(20),mi(20),rendite(20),
st(4,20),we(4,3)
480 RESTORE 490:FOR n=1 TO 20:READ firma
$(n):NEXT
490 DATA Acorn   1,Amstrad,Apple,Atari,A
T&T    5,BASF,Commodore,Fujitsu,IBM,Oliv
etti,Phillips,RCA    12,Sanyo,Schneider,
Sharp,Siemens,Sinclair,Tandy  18,Texas I
n.,Xerox
500 zeit=1:EVERY 50,1 GOSUB 3210:ag=264:
r=1:zlimit=30:krepro=50:b(1)=2000
510 FOR n=1 TO 20:RANDOMIZE TIME:k(n)=RN
D*0.135+0.935
520 kurs(n)=CINT((RND+0.1)*250+10):divi(
n)=ROUND(kurs(n)/22+RND*3)
530 rendite(n)=divi(n)/kurs(n):entw(1,n)
=kurs(n):mi(n)=kurs(n):ma(n)=mi(n):NEXT
540 FOR u=1 TO 3 STEP 2:FOR n=1 TO sp:we
(n,u)=2000:runde(n)=1000:NEXT n,u
550 ENT -1,10,-1,1
560 REM
570 REM Änderungen
580 REM
590 LOCATE#1,1,11:INPUT#1,"Wollen Sie ir
gendwelche Änderungen vornehmen";a$
600 IF LOWER$(a$)="n" THEN CLS:GOTO 820
610 h(1)=10000:n(1)=200:n(2)=1:h(2)=100:
b(2)=50:n(3)=10:h(3)=600:b(3)=30
620 CLS#1:PRINT#1:PRINT#1,"Es können geä
ndert werden:":PRINT#1
630 PRINT#1,"1) Startkapital in $";"2) H
öchster Kredit in % des Geldes";"3) Zeit
limit in sek."SPC(9)"4) Firmennamen"SPC(
15)"5) WEITER"
640 INPUT#2,"Ihre Wahl";n:IF n<1 OR n>5
THEN 640
650 ON n GOTO 660,660,660,710,770
660 CLS#2:PRINT#2,"Höchstmöglicher Wert
ist"SPC(10)"Kleinstmöglicher Wert ist"SP
C(9)"Der momentane Wert liegt bei"
670 PEN#2,1:LOCATE#2,30,1:PRINT#2,USING"
#####";h(n):LOCATE#2,30,2:PRINT#2,USING"
#####";n(n):LOCATE#2,30,3:PRINT#2,USING"
#####";b(n):PEN#2,2
680 LOCATE#2,1,5:INPUT#2,"Ihre Änderung"
;b(n)
690 IF b(n)<n(n) OR b(n)>h(n) THEN 680
700 CLS#2:GOTO 640
710 CLS#1:FOR u=0 TO 1:FOR n=1 TO 10:LOC
ATE#1,1+16*u,n:PRINT#1,n+u*10") "firma$(
n+u*10):NEXT n,u
720 LOCATE#2,1,1:INPUT#2,"Bitte geben Si
e die Nummer ein";num
730 IF num<1 OR num>20 THEN 720
740 PRINT#2,"Alter Name:";:PEN#2,1:PRINT
#2,firma$(num):PEN#2,2:PRINT#2,"Bitte ge
ben Sie den neuen Namen ein";:PEN#2,1:IN
PUT#2,nam$
750 firma$(num)=LEFT$(nam$,9):PEN#2,2:IN
PUT#2,"Weitere Änderungen";ein$:IF ein$=
"j" THEN CLS#2:GOTO 710
760 CLS#2:GOTO 620
770 FOR u=1 TO 3 STEP 2:FOR n=1 TO sp:we
(n,u)=b(1):NEXT n,u
780 zlimit=b(3):krepro=b(2)/100:ERASE n,
h:CLS
790 REM
800 REM Spielfeldaufbau
810 REM
820 WINDOW#1,1,40,14,25:WINDOW#2,1,33,18
,23:WINDOW#3,1,40,2,13
830 PAPER#2,3:PEN#2,1:PEN 2:PRINT"FIRMA
   KURS DI"
840 FOR n=1 TO sp:LOCATE 12+n*6,1:PRINT
s$(n):NEXT
850 MOVE 0,384:DRAW 640,384,2:CLS#2:GOSU
B 3110
860 REM
870 REM Hauptteil
880 REM
890 GOSUB 2020
900 FOR runde=1 TO za
910 IF r/4=ROUND(r/4) THEN GOSUB 2640
920 FOR t=1 TO sp-comp
930 IF control(t)>0 THEN zeit=1:CLS#2:PR
INT#2,"Herzlichen Glückwunsch,Sie haben
 es geschaft den Bonus in Höhe von"bonus(
```

Listing. »Börse 79« ►

```
t,bon(t))"zu kassieren.Der Bonus ist ihr
em Bargeld bereits zugez{hlt worden":CAL
L &BB18:zeit=0:control=0
940 IF r-runde(t)=2 THEN GOSUB 2820
950 GOSUB 2370:zeit=0:GOTO 1040
960 IF i>20 THEN i=1
970 IF i<=9 THEN w=i+11 ELSE w=i-9
980 LOCATE#3,1,13:PRINT#3,firma$(w):LOCA
TE 10,13:PRINT USING"####";kurs(w);:PRIN
T USING"###";divi(w):FOR n=1 TO 4:LOCATE
 12+n*6,13:IF ak(n,w)=0 THEN PRINT "====
=" ELSE PRINT USING"#####";ak(n,w)
990 NEXT:GOTO 1040
1000 IF i<1 THEN i=20
1010 CALL scroll
1020 LOCATE#3,1,1:PRINT#3,firma$(i):LOCA
TE 10,2:PRINT USING"####";kurs(i);:PRINT
 USING"###";divi(i):FOR n=1 TO 4:LOCATE
12+n*6,2:IF ak(n,i)=0 THEN PRINT"====="
ELSE PRINT USING"#####";ak(n,i)
1030 NEXT
1040 LOCATE 1,25:PRINT"            ":LOCATE
 1,25:PRINT firma$(i)
1050 IF aus=1 THEN 1500
1060 a$=LOWER$(INKEY$):IF a$="h" THEN i=
i+1:GOTO 960
1070 IF a$="n" THEN i=i-1:GOTO 1000
1080 IF a$="w" THEN g(t)=g(t)-st(t,i)*li
(t,i)*((st(t,i)<=0)+1):GOTO 1100
1090 GOTO 1050
1100 IF li(t,i)=0 THEN li(t,i)=kurs(i)
1110 IF li(t,i)>kurs(i)*1.4 THEN li(t,i)
=li(t,i)-1
1120 IF g(t)+li(t,i)*st(t,i)>we(t,1) THE
N li(t,i)=INT((we(t,1)-g(t))/st(t,i))
1130 IF li(t,i)<kurs(i)*0.6 THEN li(t,i)
=li(t,i)+1
1140 LOCATE 10,25:PRINT"       ":LOCATE 1
0,25:PRINT li(t,i)
1150 IF aus=1 THEN 1500
1160 a$=LOWER$(INKEY$):IF a$="h" THEN li
(t,i)=li(t,i)+1:GOTO 1110
1170 IF a$="n" THEN li(t,i)=li(t,i)-1:GO
TO 1110
1180 IF a$="w" THEN 1210
1190 IF a$="z" THEN g(t)=g(t)+st(t,i)*li
(t,i)*((st(t,i)<=0)+1):GOTO 1040
1200 GOTO 1150
1210 IF st(t,i)<-ak(t,i) THEN st(t,i)=-a
k(t,i)
1220 IF g(t)+li(t,i)*st(t,i)>we(t,1) THE
N st(t,i)=INT((we(t,1)-g(t))/li(t,i))
1230 LOCATE 16,25:PRINT"     ":LOCATE 16,
25:PRINT st(t,i)
1240 IF aus=1 THEN 1500
1250 a$=LOWER$(INKEY$):IF a$="h" THEN st
(t,i)=st(t,i)+1:GOTO 1220
1260 IF a$="n" THEN st(t,i)=st(t,i)-1:GO
TO 1210
1270 IF a$="w" THEN g(t)=g(t)+st(t,i)*li
(t,i)*((st(t,i)<=0)+1):CLS#2:GOTO 1300
1280 IF a$="z" THEN 1140
1290 GOTO 1240
1300 LOCATE#2,1,2:PRINT#2,"a) GRAFIK ein
er Firma"SPC(12)"b) GRAFIK zweier Firmen
"SPC(10)"c) GRAFIK dreier Firmen"SPC(10)
"d) ZAHLENWERTE"
1310 IF aus=1 THEN 1500
1320 a$=LOWER$(INKEY$):IF a$="a" OR a$="
b" OR a$="c" THEN GOSUB 2420
1330 IF a$="d" THEN CLS#2:zeit=1:s=1:GOS
UB 2430:CLS#2:LOCATE#2,1,1:FOR n=1 TO r:
PRINT#2,entw(n,x(s));:NEXT:CALL &BB18:CL
S#2:zeit=0
1340 IF a$="w" THEN CLS#2:GOTO 1370
1350 IF a$="z" THEN g(t)=g(t)-st(t,i)*li
(t,i)*((st(t,i)<=0)+1):CLS#2:GOTO 1210
1360 GOTO 1300
1370 LOCATE#2,1,2:PRINT#2,"a) eine AUSGA
BENUBERSICHT"SPC(8)"b) KREDIT aufnehmen"
SPC(14)"c) KREDIT zur}ckzahlen"
1380 IF aus=1 THEN 1500
1390 a$=LOWER$(INKEY$):IF a$="a" THEN LO
CATE#2,5,5:PRINT#2,"$"g(t)
1400 IF a$="b" THEN GOSUB 2700
1410 IF a$="c" THEN GOSUB 3020
1420 IF a$="w" THEN CLS#2:GOTO 1450
1430 IF a$="z" THEN CLS#2:GOTO 1300
1440 GOTO 1370
1450 LOCATE#2,1,3:PRINT#2,"Willst Du wir
klich aufh!ren ?"
1460 IF aus=1 THEN 1500
1470 a$=LOWER$(INKEY$):IF a$="j" THEN CL
S#2:GOTO 1510
1480 IF a$="n" THEN CLS#2:GOTO 1040
1490 GOTO 1460
1500 SOUND 1,200,200,15:aus=0:zeit=1:CLS
#2:LOCATE#2,1,3:PRINT#2,"Ihre Zeit ist u
m!"SPC(10)"Taste dr}cken":CALL &BB18:CLS
#2
1510 PLOT 584,80:DRAW 584+30*COS(ag),80-
30*SIN(ag),0:ag=264:zeit=0:NEXT
1520 IF comp=1 THEN GOSUB 3290
1530 GOSUB 2180:GOSUB 1880:GOSUB 2020
1540 NEXT:zeit=1:CALL &BB18
1550 REM
1560 REM Siegerehrung
1570 REM
1580 MODE 0:BORDER 1:PEN 2:LOCATE 4,1:PR
INT"Siegerehrung":CALL &BB18:MODE 1
1590 RESTORE 1600:MOVE 112,32:FOR n=1 TO
 26:READ a,b:DRAWR a,b,1:NEXT
1600 DATA 384,0,0,64,-128,0,0,64,-128,0,
0,-48,-128,0,0,-80
1610 DATA 0,80,32,16,96,0,0,32,32,16,128
,0,0,-64,-32,-16,128,0,32,16,0
1620 DATA -64,-32,-16,0,64,32,16,-128,0,
-32,-16,0,64,32,16
1630 MOVE 472,240:FOR n=1 TO 36:READ a,b
:DRAWR a,b,1:NEXT
1640 RESTORE 1660:MOVE 344,304:FOR n=1 T
O 36:READ a,b:DRAWR a,b,1:NEXT
1650 RESTORE 1660:MOVE 216,256:FOR n=1 T
O 36:READ a,b:DRAWR a,b,1:NEXT
1660 DATA 0,-128,8,0,8,-8,-32,0,0,72,0,-
72,-32,0,8,8,8,0,0,128,0,-80,-16,0,0,16
1670 DATA 16,0,-16,0,0,72,16,8,8,0,0,11,
-8,5,0,16,8,4,16,0,8,-4,0,-16,-8,-5,0
1680 DATA -11,9,0,15,-8,0,-88,-16,0,0,16
,16,0,-16,0,0,16,-32,0
1690 TAG:PLOT 174,80,3:PRINT"2";:PLOT 29
7,128,3:PRINT"1";:PLOT 430,76,3:PRINT"3"
;:TAGOFF:a=0:k=0
1700 a=a+1:hoch=0:we(k,3)=0:FOR t=1 TO s
p
1710 IF we(t,3)>hoch THEN hoch=we(t,3):k
=t:sieg(a)=we(k,3):sieg$(a)=s$(k)
1720 NEXT
1730 IF a<sp THEN 1700
1740 FOR n=1 TO sp:IF n<3 THEN LOCATE 1,
1+n:ELSE LOCATE 27,n-1
1750 PRINT n")";:PRINT sieg$(n);:PRINT S
PC(5-LEN(sieg$(n)));:PRINT USING"#####";
sieg(n):NEXT
1760 EVERY 10,1 GOSUB 3630
1770 ENT -1,1,-127,1,5,24,1:ENV 1,1,15,1
,5,-3,2:ENV 2,15,-1,10
1780 READ a
1790 IF a=-1 THEN RESTORE 1810:GOTO 1780
```



```
1800 SOUND 4,a,20,15
1810 DATA 239,179,179,179,142,159,179,15
9,142,179,179,142,119,106,106,106,119,14
2,142,179,159,179,159,142,179,213,213,23
9
1820 DATA 179,179,106,119,142,142,179,15
9,179,159,106,119,142,142,119,106,106,10
6,89,119,142,142,179,159,179,159,142,179
,213,213,239,179,179,179,-1
1830 IF soun=1 THEN RETURN
1840 GOTO 1780
1850 REM
1860 REM Entscheidung
1870 REM
1880 zeit=1:FOR t=1 TO sp:FOR i=1 TO 20
1890 ON SGN(st(t,i))+2 GOTO 1900,1930,19
10
1900 IF li(t,i)>kurs(i) THEN 1930 ELSE 1
920
1910 IF li(t,i)<kurs(i) THEN 1930
1920 ak(t,i)=ak(t,i)+st(t,i):we(t,1)=we(
t,1)-li(t,i)*st(t,i)
1930 st(t,i)=0:li(t,i)=0:g(t)=0:NEXT i,t
1940 FOR t=1 TO sp:we(t,2)=0
1950 FOR i=1 TO 20:we(t,2)=we(t,2)+ak(t,
i)*kurs(i)
1960 IF ak(t,i)*kurs(i)>=za*250*(b(1)/20
00) AND za>=15 AND bonus(t,i)<1 THEN bon
(t)=i:bonus(t,bon(t))=za*50:we(t,1)=we(t
,1)+bonus(t,bon(t)):control(t)=1
1970 NEXT:we(t,3)=we(t,1)+we(t,2):NEXT
1980 zeit=0:RETURN
1990 REM
2000 REM reinschreiben
2010 REM
2020 e=2:zeit=1:SOUND 1,300,300,4,0,1:FO
R n=1 TO 12
2030 PEN 2:LOCATE 1,e:PRINT firma$(n);SP
C(9-LEN(firma$(n)))
2040 PEN 1:LOCATE 10,e:PRINT USING "####
";kurs(n);:PRINT USING "###";divi(n)
2050 FOR u=1 TO 4:LOCATE 12+u*6,e:IF ak(
u,n)=0 THEN PRINT "=====" ELSE PRINT USI
NG"#####";ak(u,n)
2060 NEXT:e=e+1:NEXT
2070 f=13:FOR u=1 TO 3:f=f+1
2080 FOR n=1 TO 4:LOCATE 12+n*6,f:PRINT
USING"#####";we(n,u):NEXT n,u
2090 PEN 2:LOCATE 1,14:PRINT"Bargeld":PR
INT"Aktienwert":PRINT"Insgesamt"
2100 MOVE 0,191:DRAW 640,191,2
2110 MOVE 0,144:DRAW 640,144,2
2120 LOCATE 1,24:PRINT"FIRMA     LIMIT ST
JCK  KURVE HELP WEITER"
2130 LOCATE 25,17:PRINT"RUNDE";:PEN 1:PR
INT r:IF control>0 THEN RETURN ELSE i=1
2140 zeit=0:RETURN
2150 REM
2160 REM kurse
2170 REM
2180 zeit=1:r=r+1
2190 FOR i=1 TO 20:RANDOMIZE TIME
2200 k1=RND*0.135+0.935
2210 na=0:FOR n=1 TO sp:na=na+st(n,i):NE
XT:na=na*kurs(i)/100
2220 k2=-19.5*(rendite(i)^2)+4.5*rendite
(i)+0.82
2230 k3=0.000025*na^2+0.00675*na+1
2240 k(i)=k(i)+(RND*0.02-0.01)
2250 IF k(i)<0.935 THEN k(i)=0.935
2260 IF k(i)>1.07 THEN k(i)=1.07
2270 IF k2<0.96 THEN divi(i)=divi(i)+1
2280 IF k2>1.04 THEN divi(i)=divi(i)-1
2290 IF k3<0.935 THEN k3=0.935
2300 IF k3>1.07 THEN k3=1.07
2310 kurs(i)=CINT(kurs(i)*k1*k2*k3*k(i))
2320 IF kurs(i)<15 THEN divi(i)=0:kurs(i
)=15
2330 rendite(i)=divi(i)/kurs(i):entw(r,i
)=kurs(i)
2340 IF kurs(i)<mi(i) THEN mi(i)=kurs(i)
2350 IF kurs(i)>ma(i) THEN ma(i)=kurs(i)
2360 NEXT:zeit=0:RETURN
2370 LOCATE 1,17:PRINT"Du bist dran,";:P
EN 2:PRINT s$(t);:PRINT SPC(5-LEN(s$(t))
):PEN 1
2380 RETURN
2390 REM
2400 REM Kurve
2410 REM
2420 zeit=1:mi=999:ma=0:CLS#2:f=(ASC(a$)
-96):FOR s=1 TO f
2430 INPUT#2,"Geben Sie bitte die  Numme
r  der darzustellenden Firma ein";x(s):I
F x(s)<1 OR x(s)>20 THEN x(s)=20
2440 IF mi(x(s))<mi THEN mi=mi(x(s))
2450 IF ma(x(s))>ma THEN ma=ma(x(s))
2460 IF a$="d" THEN RETURN
2470 NEXT:CLS#1
2480 ORIGIN 135,0:MOVE 0,191:DRAW 0,0,1:
DRAW 504,0:DRAWR 0,191:DRAWR -650,0
2490 LOCATE 2,15:PRINT ma:LOCATE 2,25:PR
INT mi
2500 IF r=1 THEN LOCATE 12,20:PRINT"Dars
tellung nicht m!glich":GOTO 2600
2510 FOR n=1 TO f:LOCATE 1,16+n:PEN n:PR
INT firma$(x(n)):NEXT
2520 y=ma-mi:v=1/y
2530 FOR u=1 TO f
2540 MOVE 0,(entw(1,x(u))-mi)*v*191
2550 FOR n=2 TO r
2560 DRAW (n-1)*505/(r-1),(entw(n,x(u))-
mi)*v*191,u
2570 DRAW (n-1)*505/(r-1),0,u
2580 MOVE (n-1)*505/(r-1),(entw(n,x(u))-
mi)*v*191
2590 NEXT n,u
2600 ORIGIN 0,0:CALL &BB18:CLS#1:PEN 1:c
ontrol=1:GOSUB 2070:control=0:GOSUB 2370
:CLS#2:GOSUB 3110:zeit=0:RETURN
2610 REM
2620 REM Dividende
2630 REM
2640 zeit=1:CLS#2:LOCATE#2,1,2:PRINT#2,"
Dividende f}r"
2650 FOR u=1 TO sp:FOR n=1 TO 20:div=div
+ak(u,n)*divi(n):NEXT
2660 we(u,1)=we(u,1)+div:LOCATE#2,16,u+1
:PRINT#2,s$(u)"="div"$":div=0:we(u,3)=we
(u,1)+we(u,2):NEXT:GOSUB 2070:zeit=0:RET
URN
2670 REM
2680 REM Kredit
2690 REM
2700 zeit=1:CLS#1:LOCATE#1,1,2:kre%=we(t
,3)*krepro:IF kre(t)>0 THEN PRINT#1,"Sie
 haben schon getankt!":CALL &BB18:GOTO 2
780
2710 IF we(t,3)<1 THEN PRINT#1,"Tut uns
leid,aber Sie sind Pleite!":CALL &BB18:G
OTO 2780
2720 IF za-r<9 THEN PRINT#1," Tut uns le
id, aber es sind weniger als  10 Runden
zu spielen.Kredit unm!glich":CALL &BB18:
GOTO 2780
2730 PRINT #1,"Sie  k!nnen  einen  Kredi
t  in  H!he  von"kre%"$ aufnehmen"
2740 LOCATE#1,1,5:INPUT#1,"Auf  wieviele
 $ soll  sich  ihr  Kredit  belaufen ";k
```

Listing. »Börse 79« (Fortsetzung) ►

```
redit%
2750 IF kredit%<1 OR kredit%>kre% THEN 2
740
2760 LOCATE#1,1,8:PRINT#1,"Wollen  Sie
den Kredit wirklich aufneh-men? Bedenken
 Sie das Sie den Kredit+10% innerhalb vo
n  10  Runden zur}ckzahlen m}ssen!":LOCA
TE#1,18,11:INPUT#1,q$
2770 IF q$="j" THEN kre(t)=kredit%*1.1:w
e(t,1)=we(t,1)+kredit%:we(t,3)=we(t,1)+w
e(t,2):runde(t)=r ELSE kredit%=0
2780 CLS#1:control=1:GOSUB 2070:control=
0:GOSUB 2370:CLS#2:GOSUB 3110:zeit=0:RET
URN
2790 REM
2800 REM Kreditr}ckzahlung
2810 REM
2820 zeit=1:CLS#1:LOCATE #1,1,3:PRINT #1
,"Es ist wieder  mal soweit. Die Schulde
n werden eingetrieben.Ihr Bargeld in H!h
e von"we(t,1)"$ wird konfisziert"
2830 LOCATE#1,1,8:PRINT#1,"Ihre Schulden
 belaufen sich auf"kre(t)"$"
2840 kre(t)=kre(t)-we(t,1):we(t,1)=0
2850 IF SGN(kre(t))=-1 THEN we(t,1)=we(t
,1)-kre(t):PRINT#1,"R}ckgeld:"kre(t)*-1"
$":kre(t)=0
2860 ON SGN(kre(t))+2 GOTO 2970,2970,287
0
2870 LOCATE #1,1,10:PRINT#1,"Taste dr}ck
en zur Pf{ndung":IF INKEY$="" THEN 2870
2880 CLS#1:n=1:LOCATE#1,1,2:PRINT#1,"Es
wurden verkauft:":LOCATE#1,1,4
2890 WHILE kre(t)>0
2900 IF ak(t,n)>0 THEN ak(t,n)=ak(t,n)-1
:kre(t)=kre(t)-INT(kurs(n)*0.8):we(t,2)=
we(t,2)-kurs(n):b=b+1
2910 IF kre(t)>0 AND ak(t,n)>0 THEN 2900
2920 IF b>0 THEN PRINT#1,b"* "firma$(n);
SPC(9-LEN(firma$(n)))" je"INT(kurs(n)*0.
8)"$":b=0
2930 IF we(t,2)=0 AND kre(t)>0 THEN PRIN
T"Sie sind Pleite":we(t,1)=-kre(t):kre(t
)=0
2940 n=n+1:WEND
2950 IF kre(t)<0 THEN we(t,1)=we(t,1)-kr
e(t)
2960 LOCATE#1,1,11:PRINT#1,"R}ckgeld:"kr
e(t)*-1"$":kre(t)=0
2970 we(t,3)=we(t,1)+we(t,2):CALL &BB18:
CLS#1:IF n=0 THEN 2980 ELSE GOSUB 2020
2980 control=1:GOSUB 2070:control=0:GOSU
B 2370:CLS#2:GOSUB 3110:zeit=0:RETURN
2990 REM
3000 REM Kredit zur}ckzahlen
3010 REM
3020 zeit=1:CLS#1:IF we(t,1)<kre(t) THEN
 PRINT#1,"Ihr Geld reicht nicht.":GOTO 3
070
3030 LOCATE#1,1,2:PRINT#1,"Ihre Schulden
 belaufen sich auf"kre(t)"$"
3040 LOCATE#1,1,5:PRINT#1,"Sie zahlen"we
(t,1)"$"
3050 LOCATE#1,1,8:PRINT#1,"R}ckgeld:"we(
t,1)-kre(t)"$"
3060 we(t,1)=we(t,1)-kre(t):kre(t)=0:run
de(t)=1000:we(t,3)=we(t,1)+we(t,2)
3070 CALL &BB18:CLS#1:control=1:GOSUB 20
70:control=0:GOSUB 2370:CLS#2:GOSUB 3110
:zeit=0:RETURN
3080 REM
3090 REM Kreis
3100 REM
3110 zeit=1:DEG:FOR n%=1 TO 48 STEP 6
3120 a1%=COS(n%)*45:a2%=SIN(n%)*45
3130 PLOT 584+a1%,80+a2%,1:PLOT 584+a2%,
80+a1%,1
3140 PLOT 584+a1%,80-a2%,1:PLOT 584+a2%,
80-a1%,1
3150 PLOT 584-a1%,80+a2%,1:PLOT 584-a2%,
80+a1%,1
3160 PLOT 584-a1%,80-a2%,1:PLOT 584-a2%,
80-a1%,1
3170 NEXT:RETURN
3180 REM
3190 REM Uhr
3200 REM
3210 IF zeit=1 THEN RETURN
3220 PLOT 584,80:DRAW 584+30*COS(aq),80-
30*SIN(aq),0
3230 aq=aq+6:PLOT 584,80:DRAW 584+30*COS
(aq),80-30*SIN(aq),1
3240 IF aq>=z1imit*6+270 THEN aus=1
3250 RETURN
3260 REM
3270 REM Comou-gambler
3280 REM
3290 zeit=1:t=sp:IF r=1 OR r>za-2 THEN 3
550
3300 IF r>3 THEN 3390
3310 fir=0:FOR n=1 TO 20
3320 IF kurs(n)/entw(r-1,n)<1.07 THEN 33
40
3330 fir=fir+1:fi(fir)=n
3340 NEXT
3350 IF fir=0 THEN 3550 ELSE qeld=INT(we
(t,1)/fir):FOR n=1 TO fir
3360 li(t,fi(n))=kurs(fi(n))+(kurs(fi(n)
)-entw(r-1,fi(n)))*1.6
3370 st(t,fi(n))=INT(geld/li(t,fi(n)))
3380 NEXT:GOTO 3550
3390 k=0:hoch=1.1:FOR n=1 TO 20
3400 IF kurs(n)/entw(r-3,n)>hoch AND we(
t,1)>kurs(n)*1.1 THEN hoch=kurs(n)/entw(
r-3,n):k=n
3410 NEXT
3420 IF k=0 THEN 3450
3430 li(t,k)=kurs(k)+((kurs(k)-entw(r-3,
k))/3)*1.6
3440 st(t,k)=INT(we(t,1)/li(t,k))
3450 IF (r+1)/4=ROUND((r+1)/4) THEN 3550
3460 FOR n=1 TO 20
3470 IF ak(t,n)=0 THEN 3540
3480 IF kurs(n)/entw(r-3,n)<1.03 AND kur
s(n)/entw(r-1,n)<1.02 THEN 3500
3490 GOTO 3540
3500 IF kurs(n)-entw(r-3,n)>0 THEN 3520
3510 li(t,n)=kurs(n)+((kurs(n)-entw(r-3,
n))/3)*1.5:GOTO 3530
3520 li(t,n)=kurs(n)-((kurs(n)-entw(r-3,
n))/3)*1.5
3530 st(t,n)=-ak(t,n)
3540 NEXT
3550 CLS#2:LOCATE#2,1,1:PRINT#2,"Aktione
n des Computers:":FOR n=1 TO 20
3560 IF st(t,n)=0 THEN 3580
3570 PRINT#2,firma$(n)""SPC(9-LEN(firma$
(n)))"Limit:"li(t,n)"St}ck:"st(t,n)
3580 NEXT
3590 zeit=0:RETURN
3600 REM
3610 REM Schlagzeug
3620 REM
3630 k=k+1:IF k=5 THEN SOUND 2,0,10,14,2
,0,2:k=1:GOTO 3650
3640 SOUND 1,k*150,6,14,1,1,2
3650 RETURN
```

Listing. »Börse 79« (Schluß)

Daten gut aufbereitet

Adressen, Telefonnummern, Geburtstage — es gibt viele Daten, an die man sich erinnern muß. Eine Datenverwaltung ist dabei eine nützliche Hilfe.

Jeder Mensch muß in seinem Leben unterschiedlichste Informationen aufnehmen und im richtigen Moment zur Hand haben. Solange es Computer gibt, gehören Datenverwaltungsprogramme zu der am meisten benutzten Software. Unser Programm »Data 464« verwandelt auch Ihren Schneider in eine »Datenverwaltungsmaschine«. Damit Sie es individuellen Wünschen leicht anpassen können, ist es vollständig in Basic geschrieben. Routinen, die in der abgedruckten Version nicht eingebaut sind, lassen sich dadurch auch von einem weniger erfahrenen Programmierer leicht einbauen.

Rein theoretisch können beliebig viele Datenfelder angelegt werden. Der Platz ist nur durch den Speicher des CPC begrenzt. Die Auswahl einzelner Funktionen erfolgt über ein Menü, zu dem man aus jeder Unterroutine durch den »!« wieder zurückkommt. Es stehen Such-, Sortier- und Eingabe-Routinen zur Verfügung, aber auch eine zum Laden und eine zum Speichern der Daten. Alle Routinen erklären sich von selbst. Das Programm ist dadurch sehr benutzerfreundlich.

(Michael Straßer)

```
10 REM**********   DATA 464   **********
     * (c) 7.1985 by Michael Strasser *
     * Rottalstr. 5, 8000 Muenchen 80 *
     ***********************************
20 MODE 1:WINDOW #1,2,39,2,2:WINDOW #2,2
,39,3,3:WINDOW #3,2,39,23,24:WINDOW 2,39
,6,20
30 WINDOW #4,1,40,1,4:WINDOW #5,1,40,5,2
1:WINDOW #6,1,40,22,25:FOR n=4 TO 6:PEN
#n,3:NEXT
40 DEFINT a-z:DIM o(20),u(20)
50 GOSUB 1100
60 BORDER 1:INK 0,1:INK 1,24:INK 2,7:INK
 3,14:PAPER 0
70 PAPER #3,0:FOR n=0 TO 3:CLS #n:NEXT:P
EN #1,3:PEN #2,2:PEN #3,2:PAPER #3,1:PRI
NT#1,TAB(8)"A U S W A H L M E N U E";
80 PEN 1:PRINT:PRINT:PRINT TAB(8);"1 - N
eue Datei anlegen":PEN 2:PRINT TAB(8);"2
 - Weitere Eingaben":PEN 1:PRINT TAB(8);
"3 - Datei korrigieren"
90 PEN 2:PRINT TAB(8);"4 - In Datei such
en":PEN 1:PRINT TAB(8);"5 - Datei ansehe
n":PRINT TAB(12)"- ausdrucken"
100 PEN 2:PRINT TAB(8);"6 - Datei einles
en":PEN 1:PRINT TAB(8);"7 - Datei absave
n"
110 PEN 2:PRINT TAB(8);"8 - Daten sortie
ren":PEN 1:PRINT TAB(8);"9 - Ende des Pr
ogrammes":PRINT:PEN 3:PRINT TAB(8)"Bitte
 waehlen"
120 in$="":WHILE in$="":in$=INKEY$:WEND
130 in=VAL(in$):IF in<1 OR in>9 THEN SOU
ND 2,2000,5,7:IN$="":GOTO 120
140 IF in=2 AND das>0 THEN 150 ELSE IF i
n>1 AND ((in<>6 AND in<>9) AND (das=0 OR
 daz=1)) THEN LOCATE 1,1:PRINT"":GOTO 12
0
150 ON IN GOTO 190,270,330,400,510,810,8
60,920,160
160 CLS #1:CLS:LOCATE 1,12:PRINT"Wollen
Sie das Programm wirklich":PEN 2:PRINT:P
RINT"beenden ?";:PEN 1:PRINT" (j/n)"
170 in$=UPPER$(INKEY$):IF in$="J" THEN M
ODE 1:END ELSE IF in$="N" THEN CLS #1:CL
S:GOTO 70 ELSE 170
180 ' ********************
      * Datei erstellen *
      ********************
190 CLS:CLS #1:PRINT#1,"     D A T E I
 E R S T E L L E N";
200 CLEAR:PRINT:INPUT "Wieviele Datensae
tze soll die Datei   haben";das:IF das<1
 THEN CLS:GOTO 200
210 PRINT:INPUT "Wieviele Felder soll ei
n Datensatz    haben";fe
220 daz=1:DIM be$(fe),in$(fe,das)
230 CLS:FOR n=1 TO fe:PRINT"Name des"n!
en Feldes":PRINT:INPUT be$(n):PRINT:NEXT
240 PRINT#3,"Alles richtig ?  (j/n)"
250 in$=UPPER$(INKEY$):IF in$="N" THEN C
LS#3:GOTO 230 ELSE IF in$<>"J" THEN 250
260 CLS:CLS #3
270 CLS #1:PRINT#1,"     D A T E N   E I
 N G E B E N";:PRINT#3,"Beenden mit '^'"
280 PRINT#2,USING "Datensatz Nr.:###";da
z;:PRINT#2,TAB(30)USING "### frei";das-d
az
290 CLS:FOR n=1 TO fe:PRINT be$(n)+":";I
NPUT in$(n,daz):IF in$(n,daz)="^" THEN :
in$(n,daz)="":GOTO 70 ELSE NEXT:PRINT#3,
 j/n"
300 IN$=UPPER$(INKEY$):IF IN$="N" THEN C
LS #3:GOTO 290 ELSE IF IN$<>"J" THEN 300
 ELSE CLS #3:PRINT#3,"Beenden mit '^'"
310 daz=daz+1:IF daz>das THEN DAZ=DAZ-1:
CLS:CLS #2:PRINT:PRINT:PRINT"Datei voll"
:CALL &BB06:GOTO 70
320 GOTO 280
330 CLS:CLS #1:PRINT#1,"  D A T E I   K
 O R R I G I E R E N";
340 PRINT:INPUT "Nr. des zu korrigierend
en Datensatzes   ";kdas
350 CLS:PRINT#2,"Datensatz Nr.";kdas:FOR
 n=1 TO fe:PRINT be$(n):PRINT"Alt:";in$(
n,kdas):INPUT "Neu:";in$:IF IN$="#" THEN
 IN$(N,KDAS)="" ELSE IF in$<>"" THEN in$
(n,kdas)=in$
360 CLS:NEXT
370 PRINT:PRINT"Wollen Sie einen weitere
n Datensatz":PRINT:PRINT"korrigieren? j/
n"
380 in$=UPPER$(INKEY$):IF in$="J" THEN C
LS:GOTO 340 ELSE IF in$<>"N" THEN 380
390 GOTO 70
400 CLS:CLS#1:PRINT#1,"     D A T E I
  S U C H E N";
```

Listing. »Datei 464«

```
410 PRINT#2,"Suchkriterium:":PRINT:FOR n
 - ";be$(n):PRINT:NEXT
420 in$=INKEY$:in=VAL(in$):IF in<1 OR in
>fe THEN 420
430 CLS#2:CLS:PRINT#2,"Zu suchender Ausd
ruck:":PRINT:INPUT su$
440 CLS:CLS#2:PRINT#2,"Bitte warten":PRI
NT
450 FOR n=1 TO daz:IF INSTR(in$(in,n),su
$)=0 THEN 500
460 PRINT"Datensatznummer"n:FOR fe1=1 TO
 fe:PRINT be$(fe1):PRINT in$(fe1,n):NEXT
470 PRINT:PRINT#2,"Suche fortsetzen? j/n
"
480 in$=UPPER$(INKEY$):IF in$="N" THEN 7
0 ELSE IF in$<>"J" THEN 480
490 in$=""
500 NEXT:PRINT#3,"Ende der Suche":CALL &
BB06:GOTO 70
510 CLS:CLS #1:PRINT:PRINT"Ausgabe auf B
ildschirm oder Drucker?"
520 in$=UPPER$(INKEY$):IF in$="D" THEN 6
80 ELSE IF in$<>"B" THEN 520
530 CLS:PRINT#1,"     D A T E I   A N S
E H E N";
540 PRINT:PRINT"ALLE DATENSAETZE DURCHBL
AETTERN ODER  EINZELNE DATENSAETZE ANSEH
EN?"
550 in$=UPPER$(INKEY$):IF in$="A" THEN 6
10 ELSE IF in$<>"E" THEN 550 ELSE CLS
560 INPUT #2,"Nr. des Datensatzes";adas:
IF adas<1 OR adas>daz THEN 560 ELSE PRIN
T#2,"Datensatz Nr.";adas
570 FOR n=1 TO fe:PRINT be$(n):PRINT in$
(n,adas):PRINT:NEXT
580 PRINT#3,"Weiteren Datensatz ansehen?
 j/n"
590 in$=UPPER$(INKEY$):IF in$="J" THEN C
LS:CLS #3:GOTO 560 ELSE IF in$<>"N" THEN
 590
600 GOTO 70
610 CLS:PRINT#3,"1 - VORWAERTS
2 - RUECKWAERTS3 - NR. EINGABE          4
- ENDE       ";
620 ADAS=1
630 CLS:PRINT#2,"Datensatz Nr.";ADAS:FOR
 N=1 TO FE:PRINT BE$(N):PRINT IN$(N,ADAS
):NEXT
640 IN$=INKEY$:IN=VAL(IN$):IF IN<1 OR IN
>4 THEN 640
650 IF IN=1 AND ADAS<DAZ-1 THEN ADAS=ADA
S+1 ELSE IF IN=2 AND ADAS>1 THEN ADAS=AD
AS-1 ELSE IF IN=3 THEN 670 ELSE IF in=4
THEN 70
660 GOTO 630
670 CLS:INPUT #2,"Nummer:";adas:IF adas<
1 OR adas>daz THEN PRINT CHR$(7):GOTO 67
0 ELSE 630
680 CLS:PRINT#1,"     D A T E I   A U S D
 R U C K E N";
690 PRINT:PRINT"ALLE DATENSAETZE ODER EI
NZELNE        DATENSAETZE AUSDRUCKEN?"
700 in$=UPPER$(INKEY$):IF in$="A" THEN 7
50 ELSE IF in$<>"E" THEN 700 ELSE CLS
710 CLS:INPUT #2,"Datensatznummer";adas
720 FOR n=1 TO fe:PRINT#8,in$(n,adas):NE
XT
730 PRINT#3,"Weiter? j/n"
740 in$=UPPER$(INKEY$):IF in$="J" THEN C
LS #3:GOTO 710 ELSE IF in$<>"J" THEN 740
 ELSE 70
750 PRINT#8,CHR$(27)+"A"+CHR$(10);CHR$(2
7)+"2";:PRINT#8,CHR$(27)+"8";
760 adas=1:be=80:PRINT#8,CHR$(13).
770 IF in$(1,adas)<>"" THEN PRINT#8,"Dat
ensatznummer"adas:FOR n=1 TO fe:PRINT#8,
be$(n)+":";TAB(20);in$(n,adas):NEXT:PRIN
T#8
780 IF adas/be=INT(adas/be) AND adas>0 T
HEN GOSUB 800
790 adas=adas+1:IF adas>daz THEN 70 ELSE
 770
800 CLS:CLS #2:INPUT "Neues Papier einle
";E$:CLS:RETURN
810 CLS:PRINT#1,"     D A T E I   L A
D E N";:CLEAR
820 INPUT "Dateiname";ou$:PRINT"

Bitte warten, Datei wird geladen"
830 OPENIN ou$
840 INPUT #9,das,fe,daz:DIM be$(fe),in$(
fe,das):FOR n=1 TO fe:INPUT #9,be$(n):NE
XT:FOR n=1 TO daz:FOR nn=1 TO fe:INPUT #
9,in$(nn,n):NEXT NN,N
850 CLOSEIN:CLS:PRINT"Datei geladen":FOR
 n=1 TO 1000:NEXT:GOTO 70
860 CLS:PRINT#1,"     D A T E I   A B S
A V E N";
870 INPUT "Dateiname";OU$:CLS
880 PRINT"Bitte warten, Datei wird abges
aved"
890 SPEED WRITE 1:OPENOUT OU$
900 PRINT#9,das,fe,daz:FOR n=1 TO fe:PRI
NT#9,be$(n):NEXT:FOR n=1 TO daz:FOR nn=1
 TO fe:PRINT#9,in$(nn,n):NEXT nn,n
910 CLOSEOUT:CLS:PRINT"Datei gesaved":FO
R n=1 TO 1000:NEXT:GOTO 70
920 CLS:CLS #1:PRINT#1,"     D A T E N
S O R T I E R E N";
930 PRINT"Sortieren nach:":PRINT:FOR n=1
  ";be$(n):NEXT
940 in$=INKEY$:in=VAL(in$):IF in<1 OR in
>fe THEN 940
950 CLS:PRINT#2,"Datei wird sortiert"
960 s=1:o(1)=1:u(1)=daz-1
970 l=o(s):r=u(s):s=s-1
980 i=l:j=r:h$=in$(in,(l+r)/2)
990 WHILE in$(in,i)<h$ AND i<r:i=i+1:WEN
D
1000 WHILE in$(in,j)>h$ AND j>1:j=j-1:WE
ND
1010 IF i<=j THEN FOR n=1 TO fe:in$=in$(
n,i):in$(n,i)=in$(n,j):in$(n,j)=in$:NEXT
:i=i+1:j=j-1
1020 IF i<=j THEN 990
1030 IF r-i<=j-l THEN 1060
1040 IF l<j THEN s=s+1:o(s)=l:u(s)=j
1050 l=i:GOTO 1080
1060 IF i<r THEN s=s+1:o(s)=i:u(s)=r
1070 r=j
1080 IF r>l THEN 980
1090 IF s>0 THEN 970 ELSE 70
1100 o$=CHR$(150)+STRING$(38,CHR$(154))+
CHR$(156):u$=CHR$(147)+STRING$(38,CHR$(1
54))+CHR$(153):m$=CHR$(149)+SPACE$(38)+C
HR$(149)
1110 FOR n=4 TO 6:PRINT#n,o$;:NEXT
1120 FOR n=1 TO 2:PRINT#4,m$;:NEXT:PRINT
#4,u$;
1130 FOR n=1 TO 15:PRINT#5,m$;:NEXT:PRIN
T#5,u$;
1140 FOR n=1 TO 2:PRINT#6,m$;:NEXT:PRINT
#6,u$;
1150 RETURN
```

Listing. »Datei 464« (Schluß)

Funktionen anschaulich gemacht

Mathematische Gleichungen lassen sich auf dem Computer leicht veranschaulichen. Ein Trick umgeht die Schwierigkeiten, die Formel in das Programm einzubauen.

Jeder Schüler der Oberstufe, aber auch jeder, der sich für Mathematik oder Naturwissenschaften interessiert, muß sich mit mathematischen Funktionen beschäftigen. Einer Formel einen bestimmten Verlauf anzusehen ist nur bei einfachen Funktionen möglich. Alle anderen Graphen benötigen intensive Berechnungen bevor man ihren Verlauf erkennt. Hier hilft das Programm »Funktion«, das die Gleichung als durchgängige Kurve zeichnet.

Ein Trick bei der Programmierung umgeht das Problem, die Funktionsgleichung als DFN-Befehl vor Programmstart eingeben zu müssen. Bekanntlich ist es bei den Schneider-Computern nicht einfach, mathematische Funktionen mit dem Befehl INPUT oder im Direkt-Modus einzugeben. Unser Programm nutzt deshalb die Editierfunktion und die Möglichkeit, die Tasten des Schneiders softwaremäßig umzudefinieren. Dadurch wird die zu bearbeitende Funktion in die Zeile 180 eingefügt.

(Olav Panckow)

```
10 '(c)
 Olav Panckow
20 '     Vuerfels 57a
30 '     5060 Berg.-Gladbach 1
40 '     Tel.: 02204/69066
50 BORDER 0:INK 0,0:INK 1,23:PAPER 0:PEN
 1:MODE 2:RAD
60 KEY 128,"180 DEF FN f (x) = ":KEY 129
, CHR$(13)+"print chr$(7);chr$(24);chr$(
22)+chr$(0):goto 170"+CHR$(13)
70 KEY 138,CHR$(13)+"run"+CHR$(13):KEY 1
39,CHR$(13)+"goto 180"+CHR$(13)
80 GOSUB 540:LOCATE 1,3:PRINT "Die Funkt
ion kann unter Verwendung folgender Verk
nuepfungen erstellt werden :"
90 LOCATE 1,5:PRINT "   +      -      *
  /              Addition, Subtraktion,
Multipl. u. Division"
100 LOCATE 1,7:PRINT "       SQR (x)
^                    Quadratwurzel von x, P
otenzen"
110 LOCATE 1,9:PRINT "SIN (x) COS (x) TA
N (x) ATN (x)      Trigonometrische Funkt
ionen von x in RAD"
120 LOCATE 1,11:PRINT "EXP (x)   LOG (x)
   LOG10 (x)        e hoch x, natuerliche
r bzw. dekadischer Log."
130 LOCATE 1,13:PRINT "    ABS (x)   SGN
 (x)                 Absolutbetrag bzw. Si
gnum von x"
140 LOCATE 1,15:PRINT "Achten Sie auf ei
ne computergerechte Funktions-Eingabe wi
e z.B. y= 2*(x^2)/(x-1)"
150 LOCATE 1,17:PRINT "Diese Eingabe ist
 jetzt mit der >0<-Taste auf dem Ziffern
block abzurufen und     mit der kleinen E
NTER - Taste auf dem Ziffernblock zu bee
nden."
160 LOCATE 1,19:PRINT "Bei Schreibfehler
n ist die Eingabe durch Druecken der >1<
-Taste auf dem          Ziffernblock ruec
kgaengig zu machen."
170 LOCATE 11,23:PRINT CHR$(24);"

                ";CHR$(22)+CHR$(1);CHR$(7
):LOCATE 1,22:END
180 DEF FN f (x) =
190 PRINT CHR$(24);CHR$(22)+CHR$(0);CHR$
(7):CLS:GOSUB 540
200 LOCATE 1,10:PRINT "Geben Sie nun die
 Eckpunkte des Koordinatenkreuzes mit de
r                       grossen (!)  ENTE
R - Taste ein ..."
210 LOCATE 1,12:PRINT "(Um graphische Ve
rzerrungen zu vermeiden, empfiehlt sich
ein Laengenverhaeltnis
    von x- zu y-Achse gleich 1.8 zu 1)"
220 LOCATE 22,16:INPUT "X-Achsenabschnit
t ? (z.B.-5,100) ",x1,x2
230 LOCATE 22,18:INPUT "Y-Achsenabschnit
t ? (z.B. -10,0) ",y1,y2
240 yy1=y1:yy2=y2:IF y2 < y1 THEN y1=yy2
:y2=yy1
250 xx1=x1:xx2=x2:IF x2 < x1 THEN x1=xx2
:x2=xx1
260 fx=607/(ABS(x1)+ABS(x2)):fy=335/(ABS
(y1)+ABS(y2)):PRINT CHR$(7)
270 ORIGIN 0,16,0,639,383,16:PAPER 1:PEN
 0:CLS
280 WINDOW #1,1,80,25,25:PAPER #1,0:PEN
#1,1:CLS #1
290 PLOT 0,351-y2*fy,0:DRAWR 4,4:PLOTR -
4,-4:DRAWR 4,-4:PLOTR -4,4:DRAWR 639,0:D
RAWR -4,4:PLOTR 4,-4:DRAWR -4,-4
300 PLOT 623-x2*fx,367:DRAWR -4,-4:PLOTR
 4,4:DRAWR 4,-4:PLOTR -4,4:DRAWR 0,-367:
DRAWR -4,4:PLOTR 4,-4:DRAWR 4,4
310 m=0.1:IF ABS(x1)>10 OR x2>10 OR ABS(
y1)>4 OR y2>4 THEN m=1
320 IF ABS(x1)>100 OR x2>100 OR ABS(y1)>
40 OR y2>40 THEN m=10
330 n= MAX(ABS(x1),x2,ABS(y1),y2)/m
340 FOR i=1 TO n
350 l=4:IF i MOD 5 = 0 THEN l=6
360 IF i MOD 10 =  0 THEN l=10
370 PLOT 623-x2*fx-i*fx*m,351-y2*fy+1/2:
DRAWR 0,-1:PLOT 623-x2*fx+i*fx*m,351-y2*
fy+1/2:DRAWR 0,-1
380 PLOT 623-x2*fx-1/2,351-y2*fy-i*fy*m:
DRAWR 1,0:PLOT 623-x2*fx-1/2,351-y2*fy+i
*fy*m:DRAWR 1,0
390 NEXT
400 m=fx*(ABS(x1)+x2):nx=x1*fx
410 sx=16:sy=351-y2*fy
420 LOCATE #1,21,1:PRINT #1,"Die Funktio
n ";
430 ON ERROR GOTO 510
440 FOR i= 1 TO m-1
450 y=FNf((nx+i)/fx):yy=FNf((nx+i+1)/fx)
460 IF ABS (y-yy)>100 AND SGN(y)<>SGN(yy
) THEN 500
470 LOCATE #1,34,1:IF yy>y THEN PRINT #1
```

Listing. »Funktion«

x1, x2	x-Achsenabschnitt
y1, y2	y-Achsenabschnitt
fx, fy	Punkt pro x(y)-Achseneinheit
y,yy	aufeinanderfolgende Funktionswerte von f(x)

Variablenliste

```
,"steigt: f(x) =";USING "#######.#";yy EL
SE IF y>yy THEN PRINT #1,"faellt: f(x)=
";USING "#######.#";yy
480 PLOT sx+i,sy+y*fy
490 DRAW sx+i+1,sy+yy*fy
500 SOUND 4,650-i,10,3:NEXT:GOTO 520
```

```
510 RESUME 500
520 PRINT#1,"                Neu ?  -  Die Pu
nkt-Taste auf dem Ziffernblock druecken
!";
530 LOCATE 10,1:PRINT CHR$(24);"

                ";:LOCATE 80,1:PRINT " ";:CH
R$(22)+CHR$(1);CHR$(7):LOCATE 1,1:EDIT 1
80
540 LOCATE 1,1:PRINT "******************
************ FUNKTIONS-PLOTTER *********
**********************":RETURN
```

Listing. »Funktion« (Schluß)

Assembler-Disassembler für den CPC 464

Nur für CPC 464 ohne Diskettenlaufwerk

Mit diesem Entwicklungssystem steht dem Besitzer des CPC 464 ein leistungsfähiges Werkzeug zur Entwicklung von Maschinencode-Programmen zur Verfügung.

Das hier vorgestellte Programm vereinigt einen Assembler und einen Disassembler zu einem fast professionell zu nennenden Entwicklungssystem. Routinen in Maschinensprache sorgen für eine hohe Verarbeitungsgeschwindigkeit. Da diese Routinen am Ende des RAM-Speichers liegen, darf keine Diskettenstation angeschlossen sein. Deren Controller benutzt nämlich die gleichen Adressen. Auch auf dem CPC 664 (und dem 6128), bei dem das Laufwerk bekanntlich fest eingebaut ist, kann das Programm nicht benutzt werden.

Der Basic-Teil unseres Entwicklungssystems übernimmt die Ein- und Ausgaben, sowie die Programmsteuerung. Unterroutinen in Maschinencode führen das eigentliche Assemblieren, Disassemblieren und die Textverwaltung durch.

Beim Abtippen des Programms sollte man darauf achten, daß die Zeile 110 vollständig mit Leerzeichen (insgesamt genau 241) ausgefüllt wird. Zeilen, die nur Programmkommentierungen enthalten (REM-Zeilen), müssen nicht abgetippt werden, da diese nicht durch GOSUB, GOTO und so weiter angesprungen werden.

Das Basic-Programm (Listing 1) sollte direkt nach der Eingabe auf Kassette gespeichert werden, bevor man versucht, diesen Teil des Programms mit RUN zu starten. Mit Hilfe des Eingabeprogramms (Listing 2) kann man nun das Hex-Listing (Listing 3) abtippen. Dieser Programmteil in Maschinensprache muß anschließend mit »SAVE"DIS+AS",B,&988E,4236« auf Kassette gspeichert werden. Erst jetzt können beide Programmteile mit »RUN« zusammen geladen werden, das heißt das Laden des Maschinenprogramms wird durch das Basic-Programm gesteuert. Tabelle 1 gibt eine Übersicht über die Bedeutung der wichtigsten Variablen im Basic-Programm.

Wenn soweit alles klar ist, erscheint nach dem Programmstart das Hauptmenü auf dem Bildschirm:

(A) ssembler
(D) isassembler
(E) ditor
(P) rogrammende

Die in Klammern stehenden Buchstaben geben jeweils die Taste an, mit der man in den entsprechenden Modus gelangt. Durch Drücken der »P«-Taste kann man aus der Bearbeitung aussteigen, ohne das Programm im Speicher zu löschen. Ein im Speicher befindliches Assemblerprogramm wird dabei jedoch zerstört. Ein Neustart ist mit der Funktionstaste »0« möglich. Zum Eingeben eines Programms in Assembler muß man den Editor-Modus wählen. In diesem Modus findet man, wie in jedem anderen auch, eine Zeile am oberen Bildschirmrand, die laufend alle zur Verfügung stehenden Befehle zeigt.

(Q)uit ermöglicht den Rücksprung in das Hauptmenü.

Man kann immer zwischen Bildschirm- und Druckerausgabe wählen. Dies erfolgt durch den Befehl (P)rinter, der praktisch als Schalter dient. Der aktuelle Druckerstatus wird laufend in einer zweiten Zeile angegeben (on/off).

(A) uto

Der »Auto«-Befehl wird wie im normalen Basic verwendet.

(D) elete

Mit »Delete« kann man ebenfalls wie im Basic wahlweise einzelne Zeilen oder ganze Zeilenblöcke löschen.

(L) ist

Mit »List« wird auf der rechten Bildschirmhälfte das Programmlisting des Assemblerprogramms ausgegeben. Der Eingabeteil auf der linken Bildschirmhälfte bleibt dabei unverändert. Durch Drücken der Space-Taste kann man den List-Vorgang unterbrechen und auch wieder fortsetzen. Mit der »S«-Taste wird das Listen ganz beendet.

(R) enum

Der Befehl »Renum« wird wie im Basic verwendet, um die erste Zeilennummer zu ändern, sowie die Schrittweite zwischen den nachfolgenden Zeilen festzulegen. (Eingabeformat: Nummer der ersten Zeile/Schrittweite).

(S) ave Text

Mit »Save Text« wird das Programm gespeichert.

FE$ (n)	Fehlermeldungen
A$	Variable für Tastaturabfrage mit INKEY$
TMFL	Flag für Entscheidung ob original oder modifiziert assembliert werden soll
RF	Flag für Entscheidung ob im ROM- oder RAM-Bereich disassembliert werden soll
PF	Druckerflag
FE	Fehlercode
T	Variable für Schleifen
F$,LFIL$	Filename
ZN	Zeilennummer im Auto-Modus
ST$	Eingabevariable für Auto, Renum und so weiter

Tabelle 1. Bedeutung der wichtigsten Variablen

(T) ext laden

Bevor man mit »Text laden« ein Programm von der Kassette laden kann, erfolgt Abfrage nach dem Filenamen.

Wenn eine falsche Befehlstaste gedrückt wurde, so kann man dies bei den Befehlen »Delete« »List« »Renum« »Save Text« und »Text laden« durch Drücken der »\«-Taste rückgängig machen. Die Meldung Ready beziehungsweise das Zeichen »>« signalisiert die Bereitschaft für den Empfang eines neuen Befehls. Zur Übersetzung des eingegebenen Programms wählt man den Assembler-Modus.

Im Editor-Modus werden Programmzeilen mit Zeilennummer am Anfang direkt eingegeben. Hier gibt es zwei Möglichkeiten:

1. Zeilennummer, label, Space(s), Befehl
2. Zeilennummer, Space(s), Befehl

Fall also hinter einer Zeilennummer ein Leerzeichen (Space) eingegeben wird, muß als nächstes ein gültiger Befehl folgen. Ein Label wird immer ohne Leerzeichen direkt hinter die Zeilennummer geschrieben. Argumente können wahlweise als Dezimalzahlen, Hexzahlen oder als Labels angegeben werden. Das Bilden von Summen und Differenzen ist dabei möglich. Beispiel:

10START LD HL,1000+500H-START

Die Syntax der Assembler-Anweisungen ist sehr eng an den Z80-Standard angelehnt. Eingegebene Zahlen müssen immer mit einer Ziffer von 0 bis 9 beginnen. Am Anfang eines Labels muß immer ein Großbuchstabe stehen. Hexadezimalzahlen sind am Ende mit einem H zu kennzeichnen. Bei Dezimalzahlen kann man am Ende ein D zufügen, um Dezimalzahlen als solche zu kennzeichnen. Dies ist aber nicht unbedingt notwendig. Bei Kommentarzeilen muß zu Beginn ein Semikolon stehen. Zusätzlich zu den normalen Z80-Befehlen können folgende Assembler-Anweisungen von diesem Programm bearbeitet beziehungsweise übersetzt werden: ORG, DEFB, DEFW, DEFS, DEFM, EQU und END. Die folgende Liste gibt einen kurzen Überblick über die Bedeutung dieser Anweisungen.

ORG nn	gibt Startadresse des Programms an (gilt nur für »original assemblieren«)
DEFB n	legt das Byte n (8-Bit-Wert) an der momentanen Speicherstelle ab
DEFW nn	wie DEFB; es wird jedoch ein Wort im Speicher abgelegt (16-Bit-Wert)
DEFS nn	hält die angegebene Zahl an Speicherplätzen frei
DEFM »ABCD«	legt den Text in Anführungszeichen an der momentanen Speicherstelle ab
EQU nn	weist einem Label (das vor diesem Befehl stehen muß) den Wert nn zu
END nn	gibt die Einsprungadresse des Programms an (muß in jedem Programm stehen)

Das folgende Listing zeigt ein Beispielprogramm, das den Bildschirm mit der aktuellen Schriftfarbe füllt:

```
10SCRANF    EQU 0C000H
20SCREND    EQU 0FFFFH
30SCRLAE    EQU SCREND-SCRANF
40START     LD HL,SCRANF
50          LD DE,SCRANF+1
60          LD BC,SCRLAE
70          LD(HL), 255
80          LDIR
90          RET
100         END START
```

(O) riginal as. (zum Speichern)

Mit »Original as« kann man ein Programm in assemblierter Form (als Binärfile) auf Kassette speichern. Ein solches Programm ist dann jedoch erst nach erneutem Laden lauffähig.

(M) odifiziert as. (zum Testen)

Der Befehl »Modifiziert as.« startet die eigentliche Programmausführung. Hier wird das Programm lauffähig im Speicher abgelegt.

Im Assembler-Modus ist beim Speichern von Programmen auf Kassette zu beachten, daß ein Filename eine maximale Länge von 16 Zeichen haben darf.

Direkt nach dem Einstieg in diesen Modus wird die Eingabe einer vierstelligen Hex-Startadresse erwartet, von der ab disassembliert werden soll.

(R) OM/RAM

Die Taste **»R«** dient hier als Schalter für wahlweises Disassemblieren im ROM oder RAM-Bereich. In der Zeile unterhalb der Befehlsübersicht wird neben dem Druckerstatus auch laufend angezeigt, in welchem Bereich momentan disassembliert wird. Im Anfangszustand ist dies der ROM-Bereich.

(Space)

Mit der Space-Taste wird das Disassemblieren unterbrochen.

(S) tartadresse

Der Befehl Startadresse beendet das Disassemblieren und fragt nach einer neuen Startadresse, von der ab nun disassembliert werden soll.

Damit wären alle Funktionen des Assembler/Disassemblers besprochen. Wer sich noch nie mit Assembler-Programmierung beschäftigt hat, der sollte sich allerdings vor dem Ausprobieren dieses Programms zunächst einmal durch geeignete Lektüre mit der Z80-Maschinensprache und mit der internen Organisation seines Schneider-Computers vertraut machen.

(Jens Wulf)

```
100 GOTO 120
110 PRINT#0,"




  "
120 MODE 1:IF PEEK(&A4D9)=79 THEN 290
130 '*********************************
140 '*    ASSEMBLER-DISASSEMBLER     *
150 '*         VERSION 1.2           *
160 '*       (C) BY  JENS  WULF      *
170 '*         APRIL/MAI 1985        *
180 '*          HEIDEWEG 17          *
190 '*         3508  MELSUNGEN       *
200 '*       TEL. : 05661/3237       *
210 '*********************************
220 LOCATE 5,1:PRINT"Das Maschinenunterp
rogramm wird geladen,"
230 LOCATE 8,3:PRINT"bitte einen Moment
warten !"
240 OPENOUT F$:MEMORY &27FC:CLOSEOUT
250 LOAD"!
260 '
270 '******** INITIALISIERUNG ********
280 '
290 KEY 128,"RUN "+CHR$(13):DIM FE$(15)
300 SYMBOL 240,108,108,108,0,0,0,0,0
310 FOR T=1 TO 11:READ FE$(T):NEXT
320 FIRSZ=&97F5:NAESZ=&97F7:FEHFL=&97FD:
RF=1:AKTZEI=&9812:READYF=&988C
330 IOBUF=&96E9:SOUANF=&97EF:SOUEN=&97F9
:PASS=&9820:ZEIFL=&988D
340 SYMTAB=&9814:SYMANF=&980A:APOINT=&98
10:STARAD=&9816:PF=0
350 POKE SOUANF,0:POKE SOUANF+1,&38:CALL
 &A0D0
360 '
370 ' ******** MENUE ********
380 '
390 MODE 1:LOCATE 3,1:PRINT"2-PASS ASSEM
BLER,DISASSEMBLER,EDITOR"
400 LOCATE 5,3:PRINT"COPYRIGHT (C) BY JE
```

Listing 1. Assembler-Disassembler (Basic-Teil)

```
NS WULF 1985"
410 LOCATE 15,8:PRINT"M E N U E :"
420 LOCATE 14,10:PRINT"(A)SSEMBLER"
430 LOCATE 14,12:PRINT"(D)ISASSEMBLER"
440 LOCATE 14,14:PRINT"(E)DITOR"
450 LOCATE 14,16:PRINT"(P)ROGRAMMENDE"
460 LOCATE 15,18:PRINT"Waehle :"
470 GOSUB 1930
480 IF A$="A"THEN GOSUB 540:GOTO 390
490 IF A$="D"THEN GOSUB 1020:GOTO 390
500 IF A$="E"THEN GOSUB 1180:GOTO 390
510 IF A$="P"THEN 1900
520 GOTO 470
530 '
540 ' ******** ASSEMBLER ********
550 '
560 MODE 2:WINDOW#1,1,80,1,3
570 POKE FEHFL,0:WINDOW#0,1,80,4,25
580 LOCATE#1,32,1
590 PRINT#1,"Z-80 ASSEMBLER"
600 PRINT#1,"(O)riginal as. [zum Saven]
  (M)odifiziert as. [zum Testen]   (P)ri
nter   (Q)uit":GOSUB 2330
610 GOSUB 1930
620 IF A$="O"THEN TMFL=0:GOTO 670
630 IF A$="M"THEN TMFL=1:GOTO 670
640 IF A$="P"THEN GOSUB 2350:GOTO 610
650 IF A$="Q"THEN RETURN
660 GOTO 610
670 LOCATE#1,1,2:PRINT#1,SPACE$(80):LOCA
TE 1,5:PRINT"Assemblierung laeuft !"
680 IF TMFL THEN CALL &99ED ELSE CALL &9
9F7
690 FE=PEEK(FEHFL):IF FE THEN 720
700 IF PEEK(READYF)=0 THEN 760
710 PRINT:PRINT"Durchgang Nr. "PEEK(PASS
)+1:CALL &9A6F:GOTO 690
720 PRINT
730 IF FE=12 THEN PRINT"END-Befehl nicht
 gefunden":GOTO 750
740 PRINT FE$(FE);" in Zeile";PEEK(AKTZE
I)+256*PEEK(AKTZEI+1)
750 GOSUB 2040:GOTO 390
760 PRINT:PRINT"Assemblierung beendet"
770 ZONE 9:PRINT TAB(10)," Anfang"," End
e"," Laenge","Anfang","Ende","Laenge"
780 PRINT TAB(10)," Dez."," Dez."," Dez.
","Hex.","Hex.","Hex."
790 QA=PEEK(SOUANF)+256*PEEK(SOUANF+1):Q
E=PEEK(SOUEN)+256*PEEK(SOUEN+1)
800 TA=PEEK(SYMANF)+256*PEEK(SYMANF+1):T
E=PEEK(SYMTAB)+256*PEEK(SYMTAB+1)
810 PA=TE+1:PE=PEEK(APOINT)+256*PEEK(APO
INT+1)
820 PRINT"Quelltext      :",QA,QE-1,QE-QA
,HEX$(QA,4),HEX$(QE-1,4),HEX$(QE-QA,4)
830 PRINT"Symboltabelle :",TA,TE-1,TE-TA
,HEX$(TA,4),HEX$(TE-1,4),HEX$(TE-TA,4)
840 PRINT"Code           :",PA,PE-1,PE-PA
,HEX$(PA,4),HEX$(PE-1,4),HEX$(PE-PA,4)
850 PRINT:PRINT"Symboltabelle ausgeben ?
  (J/N) ";:GOSUB 2060:IF A$="N"THEN 900
860 CLS:Z=1
870 CALL &98A4:IF PEEK(READYF)=0 THEN GO
SUB 110:GOTO 900
880 IF Z<21 THEN LOCATE 1,Z:GOSUB 110:Z=
Z+1:GOTO 870
890 GOSUB 2040:GOTO 860
900 PRINT:IF TMFL=0 THEN 950
910 PRINT"Programm starten ? (J/N) ";
920 GOSUB 2060:IF A$="N"THEN 390
930 CALL PEEK(STARAD)+256*PEEK(STARAD+1)
940 GOSUB 2040:GOTO 390
950 PRINT"Programm saven ? (J/N) ";
960 GOSUB 2060:IF A$="N"THEN 390
970 INPUT "NAME (Max. 16 Zeichen)";F$
980 IF LEN(F$)>16 THEN 970
990 F$=F$+STRING$(16-LEN(F$),0)
1000 FOR T=1 TO 16:POKE IOBUF-1+T,ASC(MI
D$(F$,T,1)):NEXT:CALL &98FC:GOTO 390
1010 '
1020 ' ******** DISASSEMBLER ********
1030 '
1040 MODE 2:WINDOW#0,1,45,4,25:WINDOW#1,
1,80,1,3:LOCATE#1,30,1
1050 PRINT#1,"Z-80 DISASSEMBLER"
1060 GOSUB 2220:GOSUB 2330:LOCATE#1,6,2
1070 PRINT#1,"(Q)uit (P)rinter (S)tartad
resse (R)om/Ram <SPACE> halten/weiter"
1080 GOSUB 2180
1090 A$=UPPER$(INKEY$)
1100 IF A$=" "THEN GOSUB 1930
1110 IF A$="R"THEN RF=1-RF:GOSUB 2180
1120 IF A$="P"THEN GOSUB 2350
1130 IF A$="S"THEN GOSUB 2220
1140 IF A$="Q"THEN RETURN
1150 IF RF THEN CALL &A2A2 ELSE CALL &A2
A9
1160 GOSUB 110:GOTO 1090
1170 '
1180 ' ******** EDITOR ********
1190 '
1200 MODE 2:WINDOW#0,1,39,4,25:WINDOW#2,
41,80,4,25:WINDOW#1,1,80,1,3
1210 LOCATE#1,31,1:PRINT#1,"***** EDITOR
 *****"
1220 PRINT#1,"   (A)uto (D)elete (L)ist
(P)rinter (Q)uit (R)enum (S)ave Text (T)
ext laden":GOSUB 2330
1230 GOTO 1250
1240 PRINT
1250 PRINT">";:CALL &BB81
1260 GOSUB 1930
1270 IF A$>="0" AND A$<="9"THEN 1370
1280 IF A$="A"THEN 1440
1290 IF A$="D"THEN 1540
1300 IF A$="L"THEN 1580
1310 IF A$="P"THEN GOSUB 2350:GOTO 1260
1320 IF A$="Q"THEN RETURN
1330 IF A$="R"THEN 1730
1340 IF A$="S"THEN 1770
1350 IF A$="T"THEN 1840
1360 GOTO 1260
1370 ZAHL$=A$:Z=1:PRINT A$;
1380 GOSUB 1930
1390 IF A$>="0" AND A$<="9"THEN Z=Z+1:IF
 Z=6 THEN Z=Z-1:GOTO 1380 ELSE ZAHL$=ZAH
L$+A$:PRINT A$;:GOTO 1380
1400 IF A$=CHR$(127)THEN Z=Z-1:PRINT CHR
$(8);CHR$(16);:IF Z=0 THEN 1260 ELSE ZAH
L$=LEFT$(ZAHL$,LEN(ZAHL$)-1):GOTO 1380
1410 ZN=VAL(ZAHL$):IF ZN>65535 THEN PRIN
T"Zahl zu gross":GOTO 1240
1420 IF A$=CHR$(13)THEN SP=0:ST=ZN:PRINT
:GOTO 1560
1430 Z=0 :GOSUB 2120:GOTO 1240
1440 TEX$="AUTO: [1.Zeilennummer][;Schri
ttweite]":GOSUB 1950
1450 IF ST$="\"THEN 1890
1460 CALL &BB81
1470 PRINT ST;CHR$(8);:ZN=ST:MZN=INT(ZN/
256):LZN=ZN-MZN*256:POKE IOBUF+1,LZN:POK
E IOBUF+2,MZN:CALL &A0C0:IF PEEK(ZEIFL)T
HEN PRINT" "
;ELSE PRINT"*";
1480 GOSUB 1930
1490 IF A$=CHR$(127)THEN 1480
1500 IF A$=CHR$(13)THEN 1240
```



```
1510 Z=0:GOSUB 2120:PRINT
1520 IF Z=0 THEN 1240
1530 ST=INT(ST+SW):IF ST>65535 THEN 1240
 ELSE 1470
1540 TEX$="DELETE: [1.Zeilennummer][;bis
]":GOSUB 1950
1550 IF ST$=""OR ST$="\"THEN 1890
1560 IF SP=0 THEN SW=ST:GOSUB 2000
1570 CALL &A20D:GOTO 1890
1580 PRINT "LIST: [1.Zeilennummer][;bis]
":INPUT ST$:IF ST$="\"THEN 1890
1590 IF LEN(ST$)>11 THEN 1580 ELSE SP=IN
STR(ST$,";")
1600 IF ST$=""THEN ST=0:SW=65535:GOTO 16
50
1610 IF SP=1 THEN SW=VAL(RIGHT$(ST$,LEN(
ST$)-1)):ST=0
1620 IF SP=0 THEN ST=VAL(ST$):SW=ST
1630 IF SP>1 THEN ST=VAL(ST$):SW=VAL(RIG
HT$(ST$,LEN(ST$)-SP)):IF SW=0 THEN SW=65
535
1640 IF ST<0 OR ST>65535 OR SW<0 OR SW>6
5535 THEN 1580
1650 GOSUB 2000:CLS#2
1660 CALL &A1A8:IF PEEK(&988B)=0 THEN 18
90 ELSE WINDOW SWAP 2,0
1670 A$=UPPER$(INKEY$)
1680 IF A$=" "THEN GOSUB 1930
1690 IF A$="S"THEN 1710
1700 CALL &A1BD:IF PEEK(&988B) THEN GOSU
B 110:GOTO 1670
1710 WINDOW SWAP 0,2
1720 GOTO 1890
1730 TEX$="RENUM: [1.Zeilennummer][;Schr
ittweite]":GOSUB 1950
1740 IF ST$="\"THEN 1890
1750 IF SW<1 THEN 1730
1760 CALL &A12E:GOTO 1890
1770 INPUT "SAVE :Filename ";F$
1780 IF F$="\"THEN 1890
1790 LND=PEEK(SOUEN):MND=PEEK(SOUEN+1)
1800 POKE &37FE,LND:POKE &37FF,MND
1810 ANF=PEEK(SOUANF)+256*PEEK(SOUANF+1)
-2:SPEED WRITE 1
1820 SAVE F$,B,ANF,LND+256*MND-ANF
1830 GOTO 1890
1840 INPUT"LOAD :Filename ";LFIL$
1850 IF LFIL$="\"THEN 1890
1860 POKE &BC77,195:POKE &BC78,&14:POKE
&BC79,&A9:LOAD LFIL$:CALL &BD37
1870 POKE SOUEN,PEEK(&37FE)
1880 POKE SOUEN+1,PEEK(&37FF)
1890 PRINT"Ready";:GOTO 1240
1900 MODE 1:PRINT"Ok, das war's bis zum
naechsten Mal !"
1910 PRINT"Neustart mit Funktionstaste '
0'":END
1920 ' ********* UNTERPROGRAMM: AUF TAST
E WARTEN
1930 A$=UPPER$(INKEY$):IF A$=""THEN 1930
 ELSE RETURN
1940 ' ********* UNTERPROGRAMM: ARGUMENT
EINGABE (FUER DELETE, AUTO USW.)
1950 PRINT TEX$:INPUT ST$
1960 IF LEN(ST$)>11 THEN 1950 ELSE ST=VA
L(ST$):SP=INSTR(ST$,";")
1970 IF SP=0 THEN SW=10 ELSE SW=VAL(RIGH
T$(ST$,LEN(ST$)-SP))
1980 IF LEN(ST$)=0 THEN ST=10
1990 IF ST<0 OR ST>65535 OR SW<1 OR SW>6
5535 THEN 1950
2000 MST=INT(ST/256):LST=ST-MST*256:MSW=
INT(SW/256):LSW=SW-MSW*256
2010 POKE FIRSZ,LST:POKE FIRSZ+1,MST:POK
E NAESZ,LSW:POKE NAESZ+1,MSW
2020 RETURN
2030 ' ********* UNTERPROGRAMM: AUF TAST
E WARTEN
2040 LOCATE 60,20:PRINT"[Taste]":GOTO 19
30
2050 ' ********* UNTERPROGRAMM: JA/NEIN
ABFRAGE
2060 GOSUB 1930:IF A$<>"J"AND A$<>"N"THE
N 2060 ELSE PRINT A$:RETURN
2070 ' ********* UNTERPROGRAMM: ZEILENEI
NGABE FUER EDITOR
2080 Z=0:POKE FEHFL,0
2090 A$=INKEY$:IF A$=""THEN 2090
2100 IF A$=CHR$(13)THEN 2140
2110 IF A$=CHR$(127)THEN Z=Z-1:IF Z<0 TH
EN Z=0:GOTO 2090 ELSE PRINT CHR$(8);CHR$
(16);:GOTO 2090
2120 PRINT A$;:Z=Z+1:POKE IOBUF+Z+2,ASC(
A$):IF Z>229 THEN PRINT CHR$(7);:A$=CHR$
(127):GOTO 2110
2130 GOTO 2090
2140 IF Z=0 THEN ST=ZN:SW=ST:GOSUB 2000:
CALL &A20D:RETURN
2150 POKE IOBUF,Z+3:MZN=INT(ZN/256):LZN=
ZN-MZN*256:POKE IOBUF+1,LZN:POKE IOBUF+2
,MZN:CALL &A0DC:IF PEEK(FEHFL)=8 THEN PR
INT FE$(8)
2160 RETURN
2170 ' ********* UNTERPROGRAMM: STATUS I
M DISASSEMBLER ANZEIGEN
2180 LOCATE#1,20,3:PRINT#1,"Status : ";
2190 RR$="R"+CHR$(65+RF*14)+"M"
2200 PRINT#1,RR$:PRINT RR$:RETURN
2210 ' ********* UNTERPROGRAMM: EINGABE
EINER 4-STELLIGEN HEXZAHL
2220 INPUT"Startadresse (Hex)";HE$
2230 IF LEN(HE$)<>4 THEN 2220
2240 Z=1:HE$=UPPER$(HE$)
2250 TE=ASC(MID$(HE$,Z,1))
2260 IF (TE>47 AND TE<58) OR (TE>64 AND
TE<71) THEN Z=Z+1 ELSE 2220
2270 IF Z<4 THEN 2250
2280 DE=VAL("&"+HE$):DE=DE-(DE<0)*65536
2290 MSB=INT (DE/256):LSB=DE-256*MSB
2300 POKE &A2E0,LSB:POKE &A2E1,MSB
2310 RETURN
2320 ' ********* UNTERPROGRAMM: DRUCKER
AUSSCHALTEN
2330 PF=8
2340 ' ********* UNTERPROGRAMM: DRUCKERZ
USTAND INVERTIEREN UND ANZEIGEN
2350 PF=8-PF:POKE 384,14+PF
2360 LOCATE#1,1,3:PRINT#1,"Printer : ";
2370 IF PF THEN PRINT#1,"On ";ELSE PRINT
#1,"Off";
2380 RETURN
2390 DATA "Syntax Fehler"
2400 DATA "Nicht definierter Opcode"
2410 DATA "Nicht erlaubter Operand"
2420 DATA "Zahl fehlerhaft"
2430 DATA ""
2440 DATA "Nicht definiertes Label"
2450 DATA "Doppelt definiertes Label"
2460 DATA "Zu wenig Speicherplatz"
2470 DATA "Zahl zu gross"
2480 DATA "Operand fehlt"
2490 DATA "Sprung zu weit"
```

Listing 1. Assembler-Disassembler (Basic-Teil) (Schluß)

```
10 MODE 1:MEMORY &9800
20 DEFINT A-Z
30 WINDOW#0,1,40,2,25:WINDOW#1,1,40,1,1
40 INPUT"STARTADRESSE (DEZ.)";S
50 LOCATE#1,1,1:PRINT#1,"(Q)uit (S)tarta
dresse (R)eihe wiederh."
60 PRINT
70 PRINT HEX$(S,4)" ";:LINE INPUT ;H$
80 IF INSTR(H$,"Q")THEN END
90 IF INSTR(H$,"S")THEN 40
100 IF INSTR(H$,"R")THEN S=S-8:GOTO 60
110 Z=0:CS=0:H$=UPPER$(H$)
120 FOR T=1 TO LEN(H$):HZ=0
130 T$=MID$(H$,T+HZ,1):IF (T$>="0"AND T$
<="9")OR (T$>="A"AND T$<="F")THEN HZ=HZ+
2 ELSE 160
140 IF HZ<2 THEN 130
150 V=VAL("&"+MID$(H$,T,2)):POKE S+Z,V:C
S=CS+V:Z=Z+1:T=T+1
160 NEXT:IF Z<>8 THEN 60
170 PRINT TAB(35);HEX$(CS,4)
180 S=S+8:GOTO 70
```

Listing 2. Hex-Eingaberoutine für Listing 3

```
9888 00 00 00 00 00 00 22 01 0023
9890 C9 01 CF 4F AA CF 73 A6 047A
9898 CF 95 A6 CF 88 A6 CF 3F 0515
98A0 A8 CF 13 A7 2A FB 97 06 03F3
98A8 05 11 83 01 7E 3C 32 8C 0212
98B0 98 28 27 C5 3D 4F 06 00 023E
98B8 23 ED B0 4F 3E 07 91 47 032C
98C0 CD F5 98 7E F5 23 7E CD 053B
98C8 DF 98 23 F1 CD DF 98 06 04D5
98D0 05 CD F5 98 C1 10 D5 22 0427
98D8 FB 97 06 00 C3 05 A2 F5 03F7
98E0 07 07 07 07 CD E8 98 F1 035A
98E8 E6 0F FE 0A 38 02 C6 07 0304
98F0 C6 30 12 13 C9 3E 20 12 0254
98F8 13 10 FC C9 DD 21 1F 98 039D
9900 11 04 00 DD 19 2A 16 98 01E3
9908 22 66 B8 3E 01 32 5C B8 02C5
9910 AF 32 00 B8 3D 32 63 B8 0323
9918 21 00 02 22 5D B8 21 E9 0264
9920 96 11 4C B8 01 10 00 ED 02A9
9928 B0 DD 5E 00 DD 56 01 DD 03FC
9930 6E 04 DD 66 05 7C B5 C8 03B3
9938 A7 ED 52 7C B5 28 C1 4D 044D
9940 44 22 64 B8 EB DD 5E 02 03AA
9948 DD 56 03 ED 53 61 B8 22 03B1
9950 4B B8 78 D6 0B 38 1E 47 02F3
9958 C5 01 00 0B ED 43 5F B8 0315
9960 C5 DD E5 CD 91 99 DD E1 063C
9968 C1 2A 61 B8 09 EB 2A 48 036A
9970 B8 09 C1 18 D6 ED 43 5F 03FF
9978 B8 3E FF 32 5D B8 DD E5 04FE
9980 CD 91 99 DD E1 C3 00 99 0511
9988 36 04 9F F5 CD 92 98 F1 04B6
9990 C9 01 02 84 CD 95 98 30 037A
9998 47 06 8A 11 4C B8 CD 98 0351
99A0 98 21 63 B8 CD 9B 98 30 0404
99A8 37 2A 48 B8 22 4A B8 E5 036A
99B0 21 4C B8 11 40 00 3E 2C 01E0
99B8 CD 9E 98 E1 30 22 ED 5B 047E
99C0 5F B8 3E 16 CD 9E 98 21 038F
99C8 5D B8 DC 9B 98 30 11 21 0386
99D0 00 00 22 5F B8 21 5C B8 026E
99D8 34 AF 32 63 B8 37 18 AA 0329
99E0 B7 21 47 B8 28 A2 06 86 032D
99E8 CD A1 98 18 BC 21 89 9B 041C
99F0 36 01 23 36 00 18 0B 21 00D1
99F8 89 9B 36 00 23 36 01 21 01D2
9A00 00 9B 06 2B 36 00 23 10 012F
9A08 FB ED 73 FE 97 3E FF 32 055F
9A10 8C 9B 2A F9 97 23 22 14 0337
9A18 9B 36 FF 22 0A 98 22 FB 03AE
9A20 97 3A 89 9B B7 28 3B DD 03E9
9A28 2A EF 97 DD 7E 00 B7 28 03EA
9A30 2E 32 EE 97 DD 7E 03 FE 0441
9A38 20 28 1A E5 DD E5 E1 E5 04CF
9A40 CD A8 9E DD E1 E1 3A 09 04F5
9A48 98 B7 20 09 3A 1F 9B C6 032F
9A50 03 5F 16 00 19 3A EE 97 0250
9A58 5F 16 00 DD 19 18 CC 23 0272
9A60 18 03 21 00 00 22 10 9B 0106
9A68 22 1A 98 32 20 9B C9 DD 0364
9A70 2A EF 97 DD 7E 00 B7 32 03F4
9A78 EE 97 20 23 3A 20 9B B7 0371
9A80 C2 2E 9D 3C 2A 14 9B 36 02D5
9A88 FF 23 22 10 9B EB 21 23 031B
9A90 9B CD A6 A0 11 00 00 CD 0389
9A98 A6 A0 22 18 9B 18 CC DD 03D9
9AA0 6E 01 DD 66 02 22 12 9B 02B0
9AA8 DD E5 E1 E5 CD AB 9E 3A 05D5
9AB0 04 9B B7 20 54 3A 09 9B 02A2
9AB8 B7 C2 5C 9B 3A 20 9B B7 0419
9AC0 C2 BA 9B 3A 1F 9B B7 CA 04B9
9AC8 34 9B 2A 0A 9B ED 5B 14 02F7
9AD0 9B CD B3 A0 20 0C CD 91 0442
9AD8 A0 22 14 9B CD 00 9B C3 0399
9AE0 34 9B 11 E9 96 E5 D5 CD 04E6
9AE8 7C A0 D1 E1 3E 07 28 19 0354
9AF0 3B 1D CD AB A0 ED 5B 14 03C9
9AF8 9B CD B3 A0 20 E4 18 D6 04AA
9B00 11 E9 95 CD B3 A0 DB 3E 04C5
9B08 0B 32 FD 97 C3 CB A0 EB 04E7
9B10 2A 14 9B D5 E5 A7 ED 52 0476
9B18 44 4D E1 E5 3A 1F 9B C6 040E
9B20 03 16 00 5F 19 22 14 9B 015F
9B28 CD 00 9B EB E1 03 ED B8 04DC
9B30 D1 CD 91 A0 DD 7E 01 2A 0455
9B38 10 9B 4F E6 07 FE 07 28 0311
9B40 28 23 B7 28 09 23 FE 05 0259
9B48 38 04 23 23 18 0B 79 E6 0204
9B50 18 28 06 23 FE 10 20 01 0198
9B58 23 22 10 9B DD E1 3A EE 03D3
9B60 97 5F 16 00 DD 19 C3 73 0338
9B68 9A 3A 00 9B FE 03 28 DE 0373
9B70 FE 04 28 DA FE 07 28 E1 0412
9B78 FE 06 20 0B DD 5E 03 16 02B0
9B80 00 19 18 D5 CD 4F 9D 3A 02F9
9B88 00 9B FE 01 20 18 3A B9 0292
9B90 9B B7 20 C5 ED 53 10 9B 041C
9B98 3A 1F 9B B7 28 BE 2A 0E 02C6
9BA0 9B 73 23 72 18 B6 FE 02 036E
9BA8 20 0A 3A 1F 9B B7 20 EE 02E0
9BB0 3C C3 09 9B 2A 10 9B 19 028E
9BB8 1B 9F DD 7E 01 2A 10 9B 02E5
9BC0 22 1C 9B 06 00 4F E6 07 0218
9BC8 28 3B FE 07 CA C5 9C FE 0491
9BD0 01 20 06 16 DD 72 23 18 01C7
9BD8 2C FE 02 20 04 16 FD 18 027B
9BE0 F4 FE 03 20 04 16 ED 18 0334
9BE8 EC FE 04 20 04 16 CB 18 030B
9BF0 E4 FE 05 20 0C 1E DD 73 0381
9BF8 23 36 CB 23 23 06 01 18 0189
9C00 04 1E FD 18 F2 3A 00 9B 02FB
9C08 77 23 57 79 E6 18 28 24 02B4
9C10 FE 10 20 05 7A FE 36 28 0309
9C18 72 79 E6 18 F5 C5 CD 4F 04BF
9C20 9D C1 7B B7 20 5D F1 FE 04F9
9C28 08 20 21 73 23 4F 7A B7 025F
9C30 3E 09 20 4C 22 10 9B CD 024A
9C38 00 9B ED 5B 1C 9B A7 ED 042B
9C40 52 ED 5B 1A 9B 19 22 1A 02A1
9C48 9B C3 5C 9B FE 10 20 06 03B6
9C50 73 23 72 23 18 DE E5 3A 0340
9C58 89 9B B7 20 04 2A 1A 9B 02D8
9C60 23 23 EB A7 ED 52 55 7C 03E8
9C68 E1 3B 09 B7 20 10 CB 7A 034E
9C70 20 0C 18 DE FE FF 20 06 0345
9C78 CB 7A 28 02 18 D4 3E 0B 02A4
9C80 C3 09 9B F1 2B 2B 73 23 0344
9C88 23 18 A3 E5 21 E9 96 DD 0440
9C90 5E 02 16 00 19 7E 2B FE 0236
9C98 29 20 FA E5 2B 7E FE 2B 03FA
9CA0 20 FA 23 44 4D E1 22 0C 02DD
9CA8 9B 21 B1 9C E3 E5 C3 66 04F7
9CB0 9D 7A B7 3E 09 20 C9 D5 03D3
9CB8 CD 4F 9D 7A B7 20 F4 7B 0479
9CC0 D1 57 C3 50 9C 3A 00 9B 03A9
9CC8 FE 01 20 1A E5 2A 18 9B 02F8
9CD0 D1 D5 CD A6 A0 CD 4F 9D 0572
9CD8 ED 53 1A 9B CD A6 A0 22 0427
9CE0 18 9B E1 C3 34 9C FE 02 0424
9CE8 CA 34 9C FE 03 CA 19 9C 041A
9CF0 FE 04 CA 19 9C FE 05 20 03A4
9CF8 07 CD 4F 9D 19 C3 34 9C 036C
9D00 FE 06 20 1A E5 21 E9 96 03C3
9D08 DD 5E 02 1D 16 00 19 DD 0266
9D10 4E 03 42 D1 7B B1 28 02 02B7
9D18 ED B0 EB C3 34 9C E5 3E 053E
9D20 01 32 1E 9B CD 4F 9D ED 038F
9D28 53 16 9B E1 18 ED 3A 1E 033F
9D30 9B B7 3E 0C CA 09 9B 2A 0331
9D38 18 9B ED 5B 10 9B CD A6 0413
9D40 A0 11 00 00 CD A6 A0 CD 0391
9D48 A6 A0 AF 32 BC 9B C9 3A 044E
9D50 02 9B 3D 5F 16 00 E5 21 0252
9D58 E9 96 19 E5 3A 03 9B 3D 038F
9D60 5F 19 22 0C 9B C1 21 00 0220
9D68 00 22 B7 9B 0A FE 2B 20 0294
9D70 08 03 E5 CD A9 9D E1 18 03FC
9D78 14 FE 2D 20 0B 03 E5 CD 031F
9D80 A9 9D E1 A7 ED 52 18 06 042B
9D88 E5 CD A9 9D E1 19 CD 9A 0559
9D90 9D ED 4B B7 9B 30 D5 EB 04E4
9D98 E1 C9 E5 C5 ED 5B B7 9B 05BB
9DA0 2A 0C 9B CD B3 A0 C1 E1 0490
9DA8 C9 AF 32 21 9B 67 6F 3D 0376
9DB0 32 22 9B 0A CD B9 A0 30 034C
9DB8 1A FE 24 20 4C CD B0 9E 0393
9DC0 3A B9 9B B7 ED 5B 10 9B 0402
9DC8 C0 3A 20 9B B7 C8 ED 5B 0479
9DD0 1A 9B C9 E5 C5 CD B0 9E 0510
9DD8 60 69 D1 A7 ED 52 7D 32 042F
9DE0 1F 9B 2A 0A 9B E5 D5 ED 042A
9DE8 5B 14 9B CD B3 A0 3E 06 036B
9DF0 CA 09 9B D1 D5 CD 7C A0 04FD
9DF8 D1 28 06 E1 CD AB A0 18 0410
9E00 E4 42 4B 5E 23 56 E1 E1 040A
9E08 C9 CD 66 9E C5 30 05 3E 03D2
9E10 04 C3 09 9B CD B0 9E 0B 0361
9E18 0A FE 48 20 06 32 21 9B 0261
9E20 0B 18 04 FE 44 28 F9 0A 0294
9E28 CD 66 9E 30 10 3A 21 9B 0304
9E30 B7 20 02 18 DA 0A CD 6D 030F
9E38 9E 3B D4 C6 0A E5 C5 26 044A
9E40 00 6F 3A 22 9B 3C 32 22 01F3
9E48 9B 28 0D 47 3A 21 9B B7 02BE
9E50 C4 7B 9E CC 74 9E 10 F4 04BF
9E58 EB C1 E1 19 D1 D5 79 0B 04D0
9E60 BB 20 C4 EB C1 C9 D6 30 051A
9E68 D8 FE 0B 3F C9 D6 41 D8 04DB
9E70 FE 06 3F C9 54 5D 29 29 030F
9E78 19 29 C9 29 29 29 29 C9 0278
9E80 0A FE 2B C8 FE 2D C8 CD 04BB
9E88 9A 9D 03 ED 43 87 9B 30 03B9
9E90 EF 0B C9 23 7E FE 20 28 03AA
9E98 FA A7 C9 DD CB 05 7E 06 049B
9EA0 2C 48 C8 06 3B 0E 00 C9 0254
9EA8 23 23 23 11 E9 96 3A EE 0321
9EB0 97 D6 03 4F 3E 01 CA 09 02D1
9EB8 9B 06 00 D5 ED B0 EB 36 0434
9EC0 20 23 71 E1 AF DD 21 00 0342
9EC8 9B 32 00 9B 32 01 9B 32 025F
9ED0 04 9B 32 09 9B 32 1F 9B 0258
9ED8 22 06 9B 7E FE 3B 20 05 029C
9EE0 DD 36 09 01 C9 FE 20 28 032C
9EE8 24 CD B9 A0 30 05 DD 36 0392
9EF0 04 01 C9 06 01 23 7E B7 022D
9EF8 28 F4 FE 3B 28 F0 FE 20 04BB
9F00 28 03 04 18 F0 78 FE 07 02B4
9F08 30 E4 32 1F 9B CD 94 9E 03FC
9F10 28 DC FE 3B 28 CA 2B 11 036B
9F18 D8 A4 E5 23 13 1A 96 CB 0412
9F20 27 28 10 1A 3C 13 20 FB 01E3
9F28 1A 3C E1 1B 20 EC DD 36 0371
9F30 04 02 C9 30 E6 23 7E FE 0384
9F38 20 28 06 FE 3B 28 02 18 01C9
9F40 E2 C1 13 DD 36 01 00 E5 03AF
9F48 1A 07 38 18 DD CB 05 BE 02DC
9F50 CD A4 9F 13 20 0B 13 13 0274
9F58 1A 3C E1 20 E6 DD 71 04 038F
9F60 C9 DD 77 00 DD CB 05 FE 04C8
9F68 CD A4 9F 28 E9 C1 DD 86 0545
9F70 00 47 13 1A B0 DD 77 00 0248
9F78 DD 7E 01 E6 03 FE 03 20 0366
9F80 04 DD 77 04 C9 1B 1A E6 0340
9F88 60 FE 40 06 04 28 0D FE 02DB
9F90 20 06 03 28 07 A7 06 07 010C
9F98 20 02 06 00 3A 01 9B B0 01AB
9FA0 32 01 9B C9 D5 1A E6 1F 0388
9FA8 47 20 13 CD 94 9E 20 03 029C
9FB0 04 D1 C9 D6 3B 28 F9 0E 03DE
9FB8 01 06 01 05 D1 C9 11 FF 02B7
9FC0 A6 13 1A 3C 20 FB 10 F9 0333
9FC8 CD 94 9E 2B 0E 0A 28 E9 0353
9FD0 E5 13 23 1A E6 7F FE 23 03BB
9FD8 20 0A 3A 01 9B C6 0B 32 01FD
9FE0 01 9B 18 10 FE 24 28 0B 0213
9FE8 FE 25 20 31 DD CB 01 DE 03FB
9FF0 DD CB 01 E6 E5 AF ED 4B 055B
9FF8 06 9B ED 42 2C DD 75 02 034D
```

Listing 3. Maschinenspracheroutinen zum Assembler-Disassembler

Noch mehr interessante Informationen und jede Menge Tips & Tricks rund um die Schneider-Computer

... finden Sie im großen Schneider-Sonderteil von Happy Computer:

★ Ausführlicher Testbericht über den neuen Schneider 6128
★ Selbstbau einer PIO – auch für den Empfang von Wettersatelliten-Bilddaten – Teil 1 ★ Deutscher Zeichensatz unter CP/M
★ Einführung in die RSX-Befehle – Teil 1 ★ Tasword mit deutscher Tastatur ★ und nützliche Tips & Tricks zum Selber-Schneidern.

NEU: jetzt regelmäßig jeden Monat im großen Schneider-Sonderteil von Happy Computer

Die neue Ausgabe von Happy Computer erhalten Sie immer zur Mitte des Monats bei Ihrem Zeitschriftenhändler ..... oder – mit diesem Gutschein – durch uns.

Gutschein

FÜR EIN KOSTENLOSES PROBEEXEMPLAR VON HAPPY COMPUTER

JA, ich möchte »Happy-Computer« kennenlernen.
Senden Sie mir bitte die aktuellste Ausgabe kostenlos als Probeexemplar. Wenn mir »Happy-Computer« gefällt und ich es regelmäßig weiterbeziehen möchte, brauche ich nichts zu tun: Ich erhalte »Happy-Computer« dann regelmäßig frei Haus per Post und bezahle pro Jahr nur DM 66,— statt DM 72,— Einzelverkaufspreis (Ausland auf Anfrage).

Vorname, Name

Straße PLZ, Ort

Datum 1. Unterschrift

Mir ist bekannt, daß ich diese Bestellung innerhalb von 8 Tagen bei der Bestelladresse widerrufen kann und bestätige dies durch meine zweite Unterschrift. Zur Wahrung der Frist genügt die rechtzeitige Absendung des Widerrufs.

Datum 2. Unterschrift

Gutschein ausfüllen, ausschneiden, in ein Kuvert stecken und absenden an: Markt&Technik Verlag Aktiengesellschaft, Vertrieb, Postfach 1304, 8013 Haar

CS5865

```
A000 E1 E5 1A 07 38 5B 13 23 02B0
A008 7E B7 2B 66 1A E6 7F BE 0400
A010 20 F5 C1 AF E5 ED 42 DD 0576
A018 75 03 E1 18 2D FE 5B 20 0314
A020 11 BE 20 06 DD CB 01 C6 0364
A028 18 08 3C BE 20 07 DD CB 02E9
A030 01 CE BE 2B 11 1A 07 13 01FA
A038 30 FB 13 1A E1 3C 1B 20 02B0
A040 8F 0E 03 C3 B9 9F 1A 07 02DC
A048 30 87 23 C1 CD 94 9E CD 0467
A050 9B 9E 7E 90 28 05 7E 91 0383
A058 C2 B7 9F 3C 13 23 1A D1 0375
A060 C9 CD 9B 9E 7E FE 20 28 0493
A068 0E B8 28 0B B9 28 08 23 0205
A070 18 F2 C1 C1 C3 B7 9F 2B 04D0
A078 C1 0B 18 97 3A 1F 98 47 02B3
A080 96 4F 3B 01 46 23 1A BE 025F
A088 23 13 C0 10 F9 79 B7 07 0336
A090 C9 21 E9 96 3A 1F 98 2B 0385
A098 77 3C 4F 06 00 ED B0 2A 02CF
A0A0 10 98 EB 22 0E 98 73 23 02F1
A0A8 72 23 C9 4E 0C 0C 0C 06 01D6
A0B0 00 09 C9 7C BA C0 7D BB 0400
A0B8 C9 FE 41 D8 FE 5B 3F C9 0541
A0C0 CD 85 A1 2B 01 3E AF 32 033B
A0C8 BD 98 C9 ED 7B FE 97 C9 05B4
A0D0 AF 32 04 98 2A EF 97 22 034F
A0D8 F9 97 77 C9 ED 73 FE 97 05C5
A0E0 CD B5 A1 28 16 30 17 21 0299
A0E8 E9 96 ED 5B F9 97 4E 06 04AB
A0F0 00 ED B0 EB 71 22 F9 97 04AB
A0F8 C3 00 9B CD 65 A1 CD 85 0483
A100 A1 3B E4 2A F9 97 06 00 037D
A108 3A E9 96 4F E5 09 CD 00 03C3
A110 9B 22 F9 97 70 EB E1 E5 056E
A118 DD E5 C1 AF ED 42 44 4D 04F2
A120 E1 2B 1B ED B8 23 11 E9 03E9
A128 96 EB 4E ED B0 C9 DD 2A 053C
A130 EF 97 2A F5 97 ED 5B F7 057B
A138 97 DD 7E 00 B7 28 08 4F 0328
A140 06 00 DD 09 19 30 F2 32 0259
A148 FD 97 C0 DD 2A EF 97 2A 050B
A150 F5 97 DD 7E 00 B7 C8 DD 0543
A158 75 01 DD 74 02 4F 06 00 021E
A160 DD 09 19 18 ED DD E5 D1 0497
A168 62 6B 06 00 09 D5 EB 2A 02C6
A170 F9 97 AF ED 52 44 4D EB 04FA
A178 D1 78 B1 28 02 ED B0 EB 04AC
A180 71 22 F9 97 C9 2A EA 96 0496
A188 DD 2A EF 97 DD 7E 00 4F 0437
A190 B7 20 03 3C 37 C9 DD 7E 0371
A198 02 BC 38 06 C0 DD 7E 01 031B
A1A0 BD D0 06 00 DD 09 18 E4 0375
A1A8 2A F5 97 3E 55 32 BB 98 039E
A1B0 CD BB A1 DD 22 F1 97 D0 054D
A1B8 AF 32 BB 98 C9 FD 2A F1 04E5
A1C0 97 2A F7 97 7C FD BE 02 04BB
A1C8 38 EE 20 06 7D FD BE 01 0385
A1D0 38 E6 21 BB 01 36 20 22 0240
A1D8 F3 97 2B CD 28 A2 ED 5B 0494
A1E0 F3 97 FD E5 E1 7E B7 28 05AA
A1E8 CF D6 03 47 23 23 23 7E 02D6
A1F0 FE 22 20 02 3E F0 23 13 02A6
A1F8 12 10 F4 FD 4E 00 FD 09 0367
A200 FD 22 F1 97 13 21 BE 98 0401
A208 0E 04 ED B0 C9 2A F5 97 042E
A210 ED 5B F7 97 CD B3 A0 28 051E
A218 01 D0 CD BB A1 DB 20 05 03C4
A220 E5 CD 65 A1 E1 23 18 EB 04BC
A228 CD 78 A2 21 ED 97 E5 CD 053E
A230 78 A2 E1 36 01 06 10 FD 0345
A238 6E 01 FD 66 02 CB 1C CB 0386
A240 1D E5 C5 DC 5A A2 CD 5F 04CB
A248 A2 C1 E1 10 F0 21 83 01 03E9
A250 06 05 CB EE CB E6 23 10 03A8
A258 F9 C9 11 87 01 18 03 11 0287
A260 ED 97 21 ED 97 0E 05 A7 03E3
A268 1A BE FE 0A 3B 02 D6 0A 02CA
A270 12 3F 1B 2B 0D 20 F1 C9 027E
A278 06 05 36 00 2B 10 FB C9 0240
A280 00 FF FF F8 C7 C7 CF CF 0622
A288 CF CF FF FF FF FF FF FF 0798
A290 FF 00 00 00 FF FF FF FF 04FB
A298 FF C7 C7 C7 E7 00 00 00 043B
A2A0 00 4C DF A6 A2 C9 A9 A2 0487
A2A8 FC CD EC A2 22 E2 A2 2A 0527
A2B0 E0 A2 E5 E5 21 FA A2 E3 05EC
A2B8 E5 C3 A1 A3 D1 A7 ED 52 05A3
A2C0 45 20 02 06 02 21 83 01 0114
A2C8 22 E2 A2 EB CD 0B A3 C5 04D1
A2D0 E5 7E CD 10 A3 23 10 F9 040F
A2D8 22 E0 A2 E1 C1 C3 00 A9 04B2
A2E0 A2 A5 8B 01 00 0E A5 08 028E
A2E8 2A 07 CD 5B 22 83 01 06 0201
A2F0 1A 36 20 23 10 FB 21 9D 025C
A2F8 01 C9 E5 ED 5B E2 A2 21 049C
A300 C9 A4 01 04 00 ED B0 E1 03F0
A308 1B B2 C9 CD 20 A3 18 03 033E
A310 CD 25 A3 3E 20 E5 2A E2 03E4
A318 A2 77 23 22 E2 A2 E1 C9 048C
A320 7C CD 25 A3 7D F5 07 07 0391
A328 07 07 CD 2E A3 F1 E6 0F 0392
A330 FE 0A 38 02 C6 07 C6 30 0305
A338 C3 15 A3 1A E6 1F 01 B1 031C
A340 A2 B1 4F 0A 47 C9 1A E6 038C
A348 1F 06 01 CB D5 CD 3B A3 036E
A350 2F A6 DD 77 03 1A E6 1F 034B
A358 47 11 00 A7 1A 3C 13 20 0188
A360 FB 10 F9 D5 1A 07 13 30 033D
A368 FB 1A DD BE 03 C1 28 08 03A4
A370 13 1A 3C 20 EE 47 D1 C9 035B
A378 0A FE 49 28 07 03 0A 0B 0198
A380 FE 49 20 04 DD CB 00 D6 03E9
A388 DD 7E 00 E6 03 28 0C 0A 0282
A390 FE 48 28 DC 03 0A FE 4B 039D
A398 0B 28 D5 D1 C9 E5 CD 0B 045F
A3A0 A3 DD 21 E4 A2 0E 00 7E 03B3
A3A8 FE DD 20 0B CB C1 23 7E 0433
A3B0 FE CB 20 03 23 18 13 FE 033B
A3B8 FD 20 04 CB C9 18 EF FE 04BA
A3C0 ED 20 03 CB E9 23 FE CB 04B0
A3C8 20 03 CB F1 23 DD 71 00 0350
A3D0 11 07 A5 ED 53 E5 A2 1A 039E
A3D8 07 13 30 FB DD CB 00 FE 03EB
A3E0 DD CB 00 96 CD 3B A3 A6 048F
A3E8 F5 1A 07 06 FF 3B 08 DD 0338
A3F0 CB 00 BE 13 CD 3B A3 F1 0438
A3F8 A0 47 13 1A B8 20 5E 1B 0265
A400 1A DD AE 00 E6 60 20 54 035F
A408 CD 46 A3 78 B7 28 4D C5 041F
A410 1B DD CB 00 7E 06 01 20 0268
A418 06 CD 46 A3 13 78 A7 FD 03EB
A420 E1 28 39 DD 7E 00 E6 03 0386
A428 28 06 DD CB 00 56 28 39 028D
A430 FD E5 C5 23 ED 5B E5 A2 0599
A438 06 07 1A E6 7F CD 15 A3 0311
A440 1A 07 05 13 30 F4 CD CF 02F9
A448 A4 D1 CD 6B A4 7A 3D 28 0430
A450 05 3E 2C CD 15 A3 D1 CD 0392
A458 6B A4 C1 C9 13 13 1A 3C 0315
A460 C2 DC A3 13 1A 3C C2 D3 043F
A468 A3 E1 C9 7A 3D C8 1A E6 04CC
A470 7F FE 58 20 07 DD CB 00 03A4
A478 4E 28 01 3C FE 23 20 17 020B
A480 DD CB 00 76 28 0A 2B 2B 02A6
A488 7E CD 25 A3 23 23 18 33 02A4
A490 7E CD 25 A3 23 18 2C FE 037B
A498 24 20 0D D5 5E 23 56 EB 02E8
A4A0 CD 20 A3 EB D1 23 18 1B 03A2
A4A8 FE 25 20 14 7E E5 FE 80 0438
A4B0 06 00 38 02 06 FF 4F 09 019D
A4B8 23 CD 20 A3 E1 23 18 03 02D2
A4C0 CD 15 A3 1A 07 13 30 A6 028F
A4C8 C9 22 01 C9 01 C7 CF C5 0411
A4D0 CD 13 A3 10 FB C1 C9 FF 0517
A4D8 FF 4F 52 C7 F4 01 FF 45 04A0
A4E0 51 D5 F4 02 FF 44 45 46 03EA
A4E8 C2 F3 03 FF 44 45 46 D7 045D
A4F0 F4 04 FF 44 45 46 D3 F4 048D
A4F8 05 FF 44 45 46 CD FD 06 03A3
A500 FF 45 4E C4 F4 07 FF 48 0498
A508 41 4C D4 80 76 FF 4C C4 0466
A510 92 00 B0 00 03 82 40 0D 0214
A518 89 F9 09 97 2A 17 B9 22 030E
A520 01 97 3A 17 81 32 07 B7 025A
A528 4B 17 A7 43 03 93 06 05 01ED
A530 94 01 FF 41 44 C4 01 82 0360
A538 B0 0B 87 09 0C 88 09 01 01B9
A540 93 C6 FF 41 44 C3 01 82 0423
A548 B8 0B A7 4A 01 93 CE FF 03E5
A550 53 55 C2 82 90 93 D6 FF 04E4
A558 53 42 C3 01 82 9B 0B A7 0325
A560 42 01 93 DE FF 41 4E C4 0406
A568 82 A0 93 E6 FF 5B 4F D2 0513
A570 82 A8 93 EE FF 4F D2 82 054D
A578 B0 93 F6 FF 43 D0 82 B8 0585
A580 93 FE FF 49 4E C3 83 04 0471
A588 85 03 FF 44 45 C3 83 05 035B
A590 85 0B FF 50 55 53 C8 86 03D5
A598 C5 FF 50 4F D0 86 C1 FF 0579
A5A0 42 49 D4 18 C2 40 FF 53 03CB
A5A8 45 D4 18 C2 C0 FF 52 45 0449
A5B0 D3 18 C2 80 FF 4A D0 1A 0460
A5B8 94 C2 8A E9 94 C3 FF 43 0562
A5C0 41 4C CC 1A 94 C4 94 CD 042C
A5C8 FF 4A D2 1B 95 20 95 1B 0398
A5D0 FF 44 4A 4E DA 95 10 FF 0459
A5D8 52 45 D4 80 C9 9A C0 FF 050D
A5E0 45 D8 91 00 0E 89 E3 FF 0427
A5E8 49 CE 04 AF 40 01 96 DB 037C
A5F0 FF 4F 55 D4 0F A4 41 16 0381
A5F8 81 D3 FF 49 CD BC 00 FF 0524
A600 52 4C C3 C2 00 FF 52 CC 0440
A608 C2 10 FF 52 52 C3 C2 08 0402
A610 FF 52 D2 C2 18 FF 53 4C 049B
A618 C1 C2 20 FF 53 52 C1 C2 04CA
A620 28 FF 53 52 CC C2 38 FF 0491
A628 52 53 D4 99 C7 FF 43 43 045E
A630 C6 80 3F FF 43 50 C4 A0 047B
A638 A9 FF 43 50 44 D2 A0 B9 04AA
A640 FF 43 50 C9 A0 A1 FF 43 04DE
A648 50 49 D2 A0 B1 FF 43 50 044E
A650 CC 80 2F FF 44 41 C1 80 0440
A658 27 FF 44 C9 80 F3 FF 45 04EA
A660 C9 80 FB FF 45 5B D8 80 0538
A668 D9 FF 49 4E C4 A0 AA FF 057C
A670 49 4E 44 D2 A0 BA FF 49 044F
A678 4E C9 A0 A2 FF 49 4E 49 0438
A680 D2 A0 B2 FF 4C 44 C4 A0 0517
A688 A8 FF 4C 44 44 D2 A0 B8 04A5
A690 FF 4C 44 C9 A0 A0 FF 4C 04E3
A698 44 49 D2 A0 B0 FF 4E 45 0441
A6A0 C7 A0 44 FF 4E 4F D0 80 0497
A6A8 00 FF 4F 54 44 D2 A0 BB 0413
A6B0 FF 4F 54 49 D2 A0 B3 FF 050F
A6B8 4F 55 54 C4 A0 AB FF 4F 0455
A6C0 55 54 C9 A0 A3 FF 52 45 044B
A6C8 54 C9 A0 4D FF 52 45 54 03F4
A6D0 CE A0 45 FF 52 4C C1 80 0491
A6D8 17 FF 52 4C 43 C1 80 07 033F
A6E0 FF 52 4C C4 A0 6F FF 52 04C1
A6E8 52 C1 80 1F FF 52 52 43 0398
A6F0 C1 80 0F FF 52 52 C4 A0 0457
A6F8 67 FF 53 43 C6 80 37 FF 047B
A700 FF C1 00 FF C2 00 C3 01 0445
A708 C4 02 C5 03 C8 04 CC 05 032B
A710 28 48 4C A9 06 28 49 58 0234
A718 2B 23 A9 06 C1 07 FF C2 0386
A720 00 C3 08 C4 10 C5 18 C8 0344
A728 20 CC 28 28 48 4C A9 30 02A9
A730 28 49 58 2B 23 A9 30 C1 02B1
A738 38 FF C2 00 C3 08 C4 10 0398
A740 C5 18 C8 20 CC 28 C1 38 03B2
A748 FF 42 C3 00 44 C5 10 48 0365
A750 CC 20 49 D8 20 53 D0 30 0380
A758 FF 42 C3 00 44 C5 10 48 0365
A760 CC 20 49 D8 20 41 C6 30 0364
A768 FF 42 C3 00 44 C5 10 48 0365
A770 CC 20 53 D0 30 FF 42 C3 0443
A778 00 44 C5 10 49 D8 20 53 02AD
A780 D0 30 FF 48 CC 00 49 D8 0434
A788 00 FF 28 48 4C A9 00 28 028C
A790 49 58 A9 00 FF 48 CC 00 035D
A798 FF 49 D8 00 FF 53 D0 00 0442
A7A0 FF 28 53 50 A9 00 FF 28 039A
A7A8 43 A9 00 FF 41 2C D2 5F 0389
A7B0 52 2C C1 4F 41 2C C9 57 031B
A7B8 49 2C C1 47 FF 44 45 2C 0331
A7C0 48 CC EB 41 46 2C 41 46 0339
A7C8 A7 08 FF 41 2C 2B 42 43 02CB
A7D0 A9 0A 41 2C 28 44 45 A9 027A
A7D8 1A 28 42 43 29 2C C1 02 01DF
A7E0 28 44 45 29 2C C1 12 FF 02DB
A7E8 A3 00 FF A4 00 FF A5 00 03EA
A7F0 FF 28 23 A9 00 FF 28 24 033E
A7F8 A9 00 FF B0 00 B1 0B B2 03C3
A800 10 B3 18 B4 20 B5 28 B6 0342
A808 30 B7 38 FF 30 C8 00 38 034E
A810 C8 08 31 30 C8 10 31 38 0272
A818 C8 18 32 30 C8 20 32 38 0294
A820 C8 28 33 30 C8 30 33 38 02B6
A828 C8 38 FF 4E DA 00 DA 08 0409
A830 4E C3 10 C3 18 50 CF 20 033B
A838 50 C5 28 D0 30 CD 38 FF 0441
A840 4E DA 00 DA 08 4E C3 10 032B
A848 C3 18 FF B0 46 B1 56 B2 0489
A850 5E FF 22 24 A2 00 FF 7C 03C0
A858 CD 77 B5 58 FF 00 EF 80 04BF
A860 AB A9 FF 00 AB 01 FF 01 03FF
A868 9D 00 9D E1 95 20 DD E9 0496
A870 9D E1 BF 00 FD E1 95 E1 0591
A878 FF 00 95 20 DD 80 95 00 03A6
A880 FF 41 FF EB FF A3 BF A3 062E
A888 FF EB BF FB BF EB B7 EB 06F0
A890 FF B2 B3 F2 FF B3 BD E3 0648
A898 FF B3 BF FB FF E3 BB F3 06FC
A8A0 FB B0 FF 48 FF 00 FF 00 04C0
A8A8 FF 00 FF 00 FF 00 FF 00 03FC
A8B0 FD 00 F7 00 FF 02 FF 60 0454
A8B8 FF 78 FF 10 FF 00 FF 10 0494
A8C0 FF B8 F7 00 FF 00 FF 00 047C
A8C8 FF 00 FF 00 FF 00 FF 00 03FC
A8D0 F7 00 B7 00 FF 00 FF 00 03AC
A8D8 FF 00 F7 00 FF 00 F7 00 03EC
A8E0 FD E9 9D 00 9D E9 9D E1 0587
A8E8 9D 00 9D E1 95 20 9D E9 0456
A8F0 9D 00 00 00 00 00 00 00 009D
A8F8 00 00 00 00 00 00 00 00 0000
A900 11 96 01 7E 23 FE 23 38 02A2
A908 04 FE 7F 38 02 3E 20 12 022B
A910 13 10 F0 C9 11 FC 37 CF 03EF
A918 92 A3 00 00 00 00 00 00 0135
```

Listing 3. Maschinenspracheroutinen zum Assembler-Disassembler (Schluß)

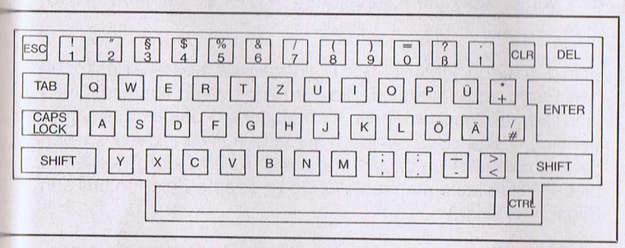
Eine deutsche Schreibmaschinentastatur für Ihren Schneider

Beim Schneider ist es im Prinzip sehr einfach, eine deutsche Tastatur zu definieren. Doch man muß sich erst einmal ein Programm schreiben. Diese Arbeit haben wir Ihnen erspart.

Das Betriebssystem des Schneiders ist sehr flexibel gestaltet. So kann man ganz einfach (sogar mit normalen Basic-Befehlen) neue Zeichen definieren oder die Tastatur umbelegen. Diese Voraussetzungen machen es natürlich leicht, ein Programm zu schreiben, das dem Computer zu einer echten deutschen Tastatur verhilft. Allerdings muß diese Routine nach dem Einschalten jedesmal geladen werden.

Der Unterschied zwischen amerikanischer und deutscher Tastatur liegt in acht neuen Zeichen und 16 umbelegten Tasten. In Bild 1 sehen Sie die deutsche Belegung und in Bild 2 die amerikanische.

Die Basic-Befehle, die wir brauchen, heißen »SYMBOL« (zur Definition der neuen Zeichen) und »KEY DEF« (um die Tasten umzubelegen). Falls Ihnen unsere Sonderzeichen nicht gefallen, so müssen Sie nur die Parameter in den Zeilen 100 bis 170 ändern (siehe auch im Handbuch). Auch spezielle andere Zeichen, wie griechische Buchstaben (für Mathematiker) oder französische Buchstaben (für Sprachenfans) lassen sich auf diese Weise sehr einfach erzeugen. (Michael Straßer)



▲ **Bild 1. Eine deutsche Schreibmaschinentastatur**

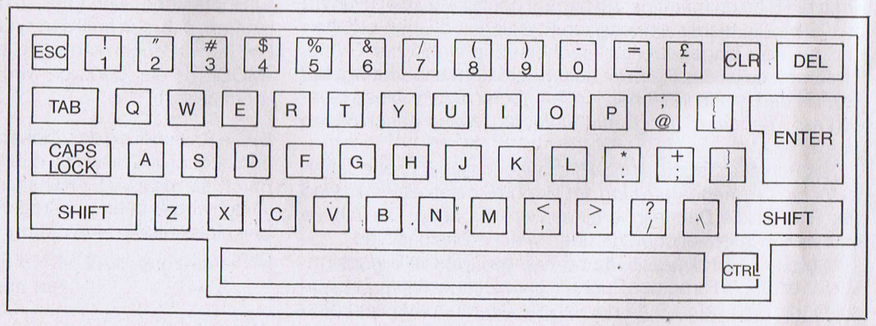


Bild 2. Die Tastatur mit amerikanischer Belegung ►

```
10 ' ***** Deutsche DIN-Tastatur *****
       ***** (c) Michael Strasser  *****
       *****      Rottalstr. 5     *****
       *****      8000 Muenchen 80 *****
20 MODE 1
30 LOCATE 10,1:PEN 1:PRINT"Deutsche DIN-
Tastatur"
40 SYMBOL AFTER 32
50 o$="#  '    _  = # ¦ { * + } < > ? `":u$="
3 7 0 - ^ @ [ : ; ] , . / \"
60 ' Anzeigen der normalen Tastatur-
     belegung
70 PEN 1:LOCATE 9,5:PRINT o$:LOCATE 9,12
:PRINT u$:PEN 3:LOCATE 1,5:PRINT"ASCII":
LOCATE 1,12:PRINT"ASCII"
80 PEN 2:LOCATE 9,6:PRINT STRING$(27,"="
):LOCATE 9,13:PRINT STRING$(27,"=")
90 ' Definierung der Umlaute,
     scharfes S, Paragraph
100 SYMBOL 64,60,96,60,102,60,6,60,0
110 SYMBOL 91,198,16,56,108,198,254,198,
0
120 SYMBOL 92,102,56,108,198,198,108,56,
0
130 SYMBOL 93,102,0,102,102,102,102,60,0
140 SYMBOL 123,108,0,120,12,124,204,118,
0
150 SYMBOL 124,102,0,60,102,102,102,60,0
160 SYMBOL 125,102,0,102,102,102,102,59,
0
170 SYMBOL 126,120,198,198,252,198,198,2
48,192
180 RESTORE 270
190 ' Tasten neu definieren
200 FOR n=1 TO 17:READ a,b,c:KEY DEF a,1
,b,c:NEXT
210 o$="@ / = ? ` ] * \ [ ' ; : _ >":u$=
"3 7 0 ~ ^ } + ¦ { # , . - <"
220 ' Anzeigen der neuen Tastatur-
     belegung
230 PEN 1:LOCATE 9,7:PRINT o$:LOCATE 9,1
4:PRINT u$:PEN 3:LOCATE 1,7:PRINT"Deutsc
h":LOCATE 1,14:PRINT"Deutsch"
240 LOCATE 13,20:PEN 1:PRINT"Y und Z ver
tauscht"
250 FOR n=1 TO 3000:NEXT:LOCATE 1,24:END
260 ' Datas fuer Tastaturumdefinierung
     (Tastennummer, Zchn 1, Zchn 2)
270 DATA 71,121,89,39,44,59,31,46,58,30,
45,95,22,60,62,29,124,92,28,123,91,19,35
,39,43,122,90,26,125,93,17,43,42,57,51,6
4,56,52,36,41,55,47,32,48,61,25,126,63,2
4,94,96
```

Listing. Mit diesem Basic-Programm bekommen Sie die vollständige DIN-Tastatur auf Ihren Schneider

Der neue Befehl: Mode 1, 2, 3

Nur für CPC 464

Mit einem Kommando alleine kann man zwar nicht alle drei Zeichengrößen gleichzeitig auf den Bildschirm bringen, aber wenige POKE-Befehle reichen aus, um dies zu ändern.

Haben Sie sich nicht auch schon einmal gefragt, ob es nicht möglich ist, verschiedene Schriftgrößen gleichzeitig auf dem Bildschirm des CPC 464 darzustellen? Gerade, weil im CPC 464 kein fester Zeichengenerator zum Bildaufbau benutzt wird und deshalb jedem Punkt auf dem Bildschirm wirklich ein oder mehrere Bits im Bildwiederholspeicher zugeordnet sind, müßte es mit geeigneter Software möglich sein, ein entsprechendes Bitmuster zu erzeugen.

In der Tat ist das wirklich sehr einfach. Man braucht nicht einmal ein kompliziertes Programm, sondern es genügt bereits die geschickte Anwendung von drei bis fünf POKE-Befehlen.

Wenn man beispielsweise im Modus 2 (640 Punkte pro Zeile) Zeichen mit der Größe des Modus 0 darstellen will, darf man natürlich nicht den »MODE 0«-Befehl verwenden. Man hat dann zwar die Zeichen in der richtigen Größe, aber dafür beträgt die Auflösung nur noch 160 Punkte pro Zeile. Die Zeichen des Modus 2 sind verschwunden. Man braucht also eine neue Mode-Routine, die nicht ganz, sondern nur »ein bißchen« in den Modus 0 umschaltet.

Wie man »ein bißchen« in Modus 0 schaltet

In der Speicherzelle &B1C8 steht der gerade aktuelle Bildschirmmodus. Der »MODE«-Befehl verändert unter anderem diese Speicherzelle — und damit auch die Bildschirmauflösung und die Anzahl der zur Verfügung stehenden Farben. Was passiert nun, wenn man unter Umgehung der Betriebssystem-Routine diese Speicherzelle verändert und sonst nichts?

Um das herauszufinden, kann man die Befehle »MODE 2« und »POKE &B1C8,0« eingeben. Das Resultat ist recht seltsam. Einerseits haben wir damit einen vollen Erfolg erzielt, denn die Bildschirmauflösung beträgt noch immer 640 Punkte pro Zeile. Die bisher eingegebenen Zeichen bleiben erhalten und die neu eingegebenen erscheinen in vierfacher Größe. Andererseits ist das Resultat aber noch recht unbefriedigend, denn in den großen Zeichen klaffen noch Lücken. Diese müssen auf irgendeine (möglichst einfache) Weise geschlossen werden.

Des Rätsels Lösung ist die »Bitmaskentabelle« (siehe Tabelle) für die Zeichendarstellung. Im Modus 0 beispielsweise sind vier Bits für die Darstellung eines Punktes nötig. Die Tabelle enthält nun Informationen darüber, welche Bits das sind.

Im Modus 1 sind die Bits 4 und 7 eines Bytes im Bildschirmspeicher für die Farbe des Punktes ganz links in einem Zeichen zuständig. Im ersten Byte der Tabelle (im Modus 1) sind deshalb die Bits 4 und 7 gesetzt. Entsprechendes gilt auch für die zweite, dritte und vierte Spalte in einem Zeichen. Die Spalten fünf bis acht des Zeichens werden schon im nächsten Byte des Bildschirmspeichers abgelegt. Deshalb wird für diese keine neue Tabelle angelegt, sondern dieselbe noch mal von vorn gelesen. Entsprechendes gilt auch für den Modus 0, nur daß hier schon nach zwei Spalten ein weiteres Byte des Bildschirmspeichers angesprochen wird. Schalten wir aus dem Modus 2 mit dem oben genannten POKE nur »halb« in den Modus 0 um, bleibt diese Tabelle natürlich unangepaßt. Da diese Tabelle im RAM liegt und zwar an den Speicherstellen &B1CF bis &B1D6, können wir diese Anpassung mit ein paar POKEs nachholen. Selbstverständlich dürfen wir aber die Bitmaskentabelle nicht blind abschreiben. Denn es soll ja zum Setzen eines Punktes (beispielsweise ganz links in einem Zeichen) nicht je nach Farbe eine Auswahl der Bits 7, 5, 3 oder 1 getroffen werden. Im noch »halb eingeschalteten« Modus 2 entspricht ja der ums Vierfache vergrößerte linke Punkt den ersten vier Bildpunkten auf dem Bildschirm. Es müssen also die Bits 4, 5, 6 und 7 gesetzt werden. Das erste Tabellenelement darf also nicht mehr »&x10101010«, sondern muß »&x11110000« heißen. Entsprechend liegt der zweite Punkt eines Zeichens in den vier niederwertigen Bits, das zweite Tabellenelement muß damit »&X00001111« heißen. Diese beiden Werte, beziehungsweise ihre hexadezimalen Äquivalente, müssen also mit »POKE &B1CF,&F0« und »POKE &B1D0,&0F« in die Tabelle eingepaßt werden. Diese POKEs stehen auch im beigefügten Demoprogramm (siehe Listing) in der Zeile 430. Nur mit »GOSUB 430« kann man diese Bildschirmdarstellung auswählen.

Für den experimentierfreudigen Praktiker gibt es noch eine Fülle weiterer Möglichkeiten, wovon das Demoprogramm eine kleine Auswahl zeigt.

Hier noch ein paar Tips:

1) Im Modus 2 sollte die Tabelle normalerweise acht Elemente umfassen. In der Praxis zeigt sich aber, daß hier zur Zeichendarstellung diese Tabelle überhaupt nicht ausgewertet wird. Man kann sich also die acht POKEs sparen. Die Zeile 570 besteht deshalb nur aus einem einzigen POKE.

2) Setzt man zur Darstellung von Buchstaben des Mode 0 im Modus 2 nicht wie oben gezeigt alle vier zuständigen Bits, sondern nur eine Auswahl daraus, so wird auch nur eine Auswahl auf dem Bildschirm angezeigt. So kann man zum Beispiel in den verschiedensten Variationen »karierte« Zeichen erzeugen.

3) In den (Grund-)Modi 1 und 0 sind jeweils mehrere Bits für einen Punkt verantwortlich, die die Farbe verschlüsseln. Setzt man in der Bitmaskentabelle für bestimmte Punkte nicht alle eigentlich nötigen Bits, sondern nur eine Auswahl, dann werden die betroffenen Punkte in einer anderen Farbe dargestellt. Auf diese Weise kann man mehrfarbig gestreifte Zeichen auf den Bildschirm bringen.

4) Spätestens, wenn man einen Fehler macht, wird man bemerken, daß man auch eine »Geheimschrift« aus unleserlichen Zeichen erzeugen kann. In diesem Fall gibt man einfach (blind) »MODE n« ein. Dadurch wird die Zeichendarstellung wieder »normalisiert«.

5) Die Anwendung des Tricks bringt die interne Bildschirmpositionssteuerung teilweise durcheinander. Ordnung schafft man durch ein »LOCATE«-Kommando. Deshalb eignet sich dieser Trick nicht so sehr, um Programme zu editieren, wohl aber um beispielsweise in einem Programm ein tolles Titelbild zu erzeugen.

(Helmut Tischer)

Bitmaskentabelle im Modus 2

Adresse	Zuständig für Spalten in einem Zeichen	Inhalt = zuständige für betreffenden Bildpunkt
&B1CF	0	&X10000000
&B1D0	1	&X01000000
&B1D1	2	&X00100000
&B1D2	3	&X00010000
&B1D3	4	&X00001000
&B1D4	5	&X00000100
&B1D5	6	&X00000010
&B1D6	7	&X00000001
Bitmaskentabelle im Modus 1		
&B1CF	0, 4	&X10001000
&B1D0	1, 5	&X01000100
&B1D1	2, 6	&X00100010
&B1D2	3, 7	&X00010001
Bitmaskentabelle im Modus 0		
&B1CF	0, 2, 4, 6	&X10101010
&B1D0	1, 3, 5, 7	&X01010101

Tabelle. Die Bildmaskentabelle für die Zeichendarstellung legt die Größe der Zeichen fest

```
10 '************************************
20 '*        M U L T I M O D E         *
30 '*     (c) 1985 by Helmut Tischer   *
40 '*         Asternstra~e 40          *
50 '*         8052 Moosburg            *
60 '************************************
70 'Verschiedene Schriftgroe~en zugleich
 auf den Bldschirm bringen
80 '
90 'Anwendungsbeispiele - Demo's
100 '
110 a$="abcdefghijklmnoprst"
120 MODE 2:BORDER 1:INK 0,1:INK 1,24:PAP
ER 0:PEN 1
130 PRINT"Hier ein paar Schriftbeispiele
 im Modus 2:"
140 PRINT"(Theoretisch m!glich sind aber
 noch wesentlich mehr)"
150 PRINT STRING$(80,"=");:PRINT:PRINT
160 FOR i%=1 TO 8
170 ON i% GOSUB 430,450,470,490,510,530,
550,570
180 PRINT"abcdefghijklmnopqrst"
190 NEXT
200 PRINT:PRINT:PRINT STRING$(80,"=")"We
iter ? - eine Taste dr}cken"
210 q$=INKEY$:IF q$=""THEN GOTO 210
220 '
230 MODE 1:BORDER 1:INK 0,1:INK 1,24:INK
 2,6:INK 3,15:PAPER 0:PEN 3
240 PRINT"Und jetzt ein paar Beispiele i
m Modus 1:";
250 PRINT"(Wieder nur eine kleine Auswah
l)"
260 PRINT STRING$(40,"=");:PRINT:PRINT
270 FOR i%=1 TO 5
280 ON i% GOSUB 590,610,630,650,670
290 PRINT"abcdefghijklmnopqrst"
300 NEXT
310 PRINT:PRINT:PRINT STRING$(40,"=")"We
iter ? - eine Taste dr}cken"
320 q$=INKEY$:IF q$=""THEN GOTO 320
330 '
340 MODE 0:INK 0,1:INK 15,24:INK 1,15:PE
N 15:POKE &B1CF,&AA:POKE &B1D0,1
350 PRINT"Auch im Modus 0 gibt's 'Zebras
treifen'"
360 PRINT:PRINT:PRINT STRING$(20,"=")"We
iter ? - ","eine Taste dr}cken"
370 q$=INKEY$:IF q$=INKEY$ THEN GOTO 370
 ELSE GOTO 120
380 '
390 '
400 'Umschaltroutinen f}r die verschiede
nen Modis
410 '
420 'Mode 0-Lettern im Mode 2 (Version 1
)
430 POKE &B1C8,0:POKE &B1CF,&F0:POKE &B1
D0,&F:RETURN
440 'Mode 0-Lettern im Mode 2 (Version 2
)
450 POKE &B1C8,0:POKE &B1CF,&E0:POKE &B1
D0,&E:RETURN
460 'Mode 0-Lettern im Mode 2 (Version 3
)
470 POKE &B1C8,0:POKE &B1CF,&C0:POKE &B1
D0,&C:RETURN
480 'Mode 0-Lettern im Mode 2 (Version 4
)
490 POKE &B1C8,0:POKE &B1CF,&80:POKE &B1
D0,&8:RETURN
500 'Mode 0-Lettern im Mode 2 (Version 5
)
510 POKE &B1C8,0:POKE &B1CF,&A0:POKE &B1
D0,&A:RETURN
520 'Mode 1-Lettern im Mode 2 (Version 1
)
530 POKE &B1C8,1:POKE &B1CF,&C0:POKE &B1
D0,&30:POKE &B1D1,&C:POKE &B1D2,&3:RETUR
N
540 'Mode 1-Lettern im Mode 2 (Version 2
)
550 POKE &B1C8,1:POKE &B1CF,&80:POKE &B1
D0,&20:POKE &B1D1,&8:POKE &B1D2,&2:RETUR
N
560 'Normale Ausgabe im Mode 2
570 POKE &B1C8,2:RETURN
580 'Mode 0-Lettern im Mode 1 (Version 1
)
590 POKE &B1C8,0:POKE &B1CF,&CC:POKE &B1
D0,&33:RETURN
600 'Mode 0-Lettern im Mode 1 (Version 2
)
610 POKE &B1C8,0:POKE &B1CF,&88:POKE &B1
D0,&22:RETURN
620 'Mode 0-Lettern im Mode 1 (Version 3
)
630 POKE &B1C8,0:POKE &B1CF,&CC:POKE &B1
D0,&30:RETURN
640 'Mode 1-Lettern im Mode 1 (Version 1
)
650 POKE &B1C8,1:POKE &B1CF,&88:POKE &B1
D0,&40:POKE &B1D1,&2:POKE &B1D2,&11:RETU
RN
660 'Normale Ausgabe im Mode 1
670 POKE &B1C8,1:POKE &B1CF,&88:POKE &B1
D0,&44:POKE &B1D1,&22:POKE &B1D2,&11:RET
URN
```

Listing. »Multimode« demonstriert drei Modi gleichzeitig auf dem Bildschirm

Verbessern Sie den LIST-Befehl!

Ein Programmlisting zu lesen ist am Bildschirm oft nicht ganz einfach. »Superlist« schafft da Abhilfe.

Unsere verbesserte List-Routine ist vollständig kompatibel zu der des Schneider-Computer. Alle Befehlsformate (LIST, LIST Zeilennummer, LIST Zeilennummer-, LIST- Zeilennummer und LIST Zeilennummer-Zeilennummer) sind erlaubt. Die Verbesserungen beziehen sich auf die Farbwerte (die jetzt zuerst zurückgestellt werden), automatischen Aufruf von »MODE 2« und daß am Seitenende auf einen Tastendruck gewartet wird, bevor der nächste Teil des Programms gezeigt wird.

Die Leer- und die Escape-Taste haben Sonderfunktionen. Mit der Leertaste wird das Listing zeilenweise gedruckt, Escape beendet das Auflisten ohne das obligatorische »Ready«. Es wird ferner berücksichtigt, ob die letzte Zeile vollständig auf den Bildschirm paßt. Reicht der Platz nicht aus, so wird sie erst gar nicht ausgegeben. Ist der Bildschirm schon voll, wenn der Befehl »LIST« eingegeben wird, so ruft das Programm automatisch den Befehl CLS auf.

Mit dem Basic-Lader wird eine Maschinencode-Routine erzeugt. »SAVE "Superlist 1.0 MC",B,&AA80,244« sorgt dafür, daß die Betriebssystem-Erweiterung als binäres File auf der Kassette gesichert wird. Geladen wird die Routine mit »MEMORY &AA7F:LOAD "Superlist 1.0 MC":CALL &AA80«. Bis zum Zurücksetzen des Computers bleibt sie nun gespeichert. Mit »POKE &AC13,&C9« kann sie aber jederzeit abgeschaltet werden. Normalerweise steht an dieser Stelle der Wert &C3.

(Bernhard Straub)

```
10 'SUPERLIST 1.0
20 '(C) 1985 by Bernhard Straub
30 '             An der Tagmess 13
40 '             7600 Offenburg
50 'Basic-Programm zur Erzeugung des MC
von SUPERLIST 1.0
60 'Startaddr. des MC :&AA80   Endaddr.
:&AB7F
70 'SUPERLIST wird ueber die normalen LI
ST- Funktionen aufgerufen
80 MODE 2
90 PRINT "******************** SUPERLIS
T 1.0   Maschinencode-Generator ********
**********":PRINT
100 PRINT "Bitte nicht stoeren, Maschine
ncode wird gepokt!":PRINT:PRINT
110 FOR n=&AA80 TO &AB73
120 READ h$
130 h=VAL("&"+h$)
140 POKE n,h
150 i=i+h
160 NEXT
170 IF i<>30335 THEN PRINT:PRINT "Sie ha
ben leider einen Fehler beim Abtippen ge
macht, bitte ueberpruefen!":PRINT:PRINT:
END
180 CALL &AA80
190 PRINT "Alles okay!":PRINT:PRINT
```

Listing zu »Superlist«

```
200 END
210 DATA 21,13,AC,36,C3,23,01,8F,AA,79,7
7,23,78,77,C9,F5
220 DATA E5,21,40,00,7E,FE,96,28,04,E1,F
1,18,03,E1,F1,C9
230 DATA ED,43,74,AB,ED,53,76,AB,22,78,A
B,DD,22,7A,AB,FD
240 DATA 22,7C,AB,C1,D1,DD,E1,FD,E1,E1,2
1,D8,AA,E5,FD,E5
250 DATA DD,E5,D5,C5,ED,4B,74,AB,ED,5B,7
6,AB,2A,78,AB,DD
260 DATA 2A,7A,AB,FD,2A,7C,AB,C9,F5,C5,D
5,E5,DD,E5,FD,E5
270 DATA CD,11,BC,FE,02,28,05,3E,02,CD,0
E,BC,3E,01,01,18
280 DATA 18,CD,32,BC,3C,01,01,01,CD,32,B
C,21,8F,B2,36,FF
290 DATA 21,90,B2,36,00,01,01,01,CD,38,B
C,CD,78,BB,AF,7D
300 DATA D6,16,38,29,AF,7D,D6,1A,20,2E,2
1,40,00,7E,FE,A7
310 DATA 28,16,CD,06,BB,FE,20,28,14,FE,F
C,C2,38,AB,26,01
320 DATA 2E,19,CD,75,BB,C3,96,C0,3E,0C,C
D,5A,BB,FD,E1,DD
330 DATA E1,E1,D1,C1,F1,C3,33,E1,21,40,0
0,36,00,06,00,16
340 DATA 00,21,A4,AC,7E,B7,28,10,04,78,F
E,50,28,03,23,18
350 DATA F3,14,06,00,23,23,18,EC,CD,78,B
B,AF,7D,82,D6,1A
360 DATA 30,B0,18,C9          »Superlist« (Schluß)
```

Wem der Druckerpuffer zu klein ist ...

Der Scheider NLQ 401 besitzt nur einen kleinen Puffer für die Zwischenspeicherung von Daten. Dadurch wird die Ausgabe zur Geduldsprobe, wenn man nicht im Speicher des Computers Platz zur Verfügung stellt.

Preiswerte Drucker haben oft den Fehler, daß sie entweder keinen oder nur einen sehr kleinen Zwischenspeicher für die Daten haben, die vom Computer gesendet werden. Diesen Mangel kann man beheben, indem im RAM-Bereich des Schneiders ein Platz als Puffer reserviert wird.

Das Programm ist vollständig in Maschinencode geschrieben und liegt zwischen den Adressen A4A0 hex und A670 hex. Mit dem RSX-Befehl »ISPOOLER, Anfangsadresse, Endadresse« wird unsere Routine aufgerufen und der Bereich zwischen der Anfangs- und Endadresse als Zwischenspeicher reserviert. Werden beide Werte weggelassen, so sind 8000 hex (als Anfangs-) und A3FE hex (als Endadresse) voreingestellt. Mit »ISPOOLEROFF« wird der Bereich wieder freigegeben.

Unser Programm ändert den Betriebssystemvektor, der normalerweise auf die Ausgaberoutine für den Drucker weist. Dieser legt jetzt die Zeichen im Zwischenspeicher ab. Eine weitere Änderung wird in der Betriebssystemroutine FAST TICKER

COMPUTER-ZEITSCHRIFTEN VON PROFIS FÜR PROFIS

COMPUTER PERSÖNLICH

Das aktuelle Fachmagazin für Personal-Computer.

- ★ **Wenn Sie jetzt den Schritt vom Heim-Computer zur professionellen Anwendung eines Personal Computers planen**
- ★ **Wenn Sie beruflich oder privat bereits einen Personal Computer benutzen**
- ★ **Wenn Sie regelmäßig Informationen über das aktuelle Produktangebot benötigen**
- ★ **Wenn Sie selbst programmieren**
- ★ **Wenn Sie professionelle Hard- und Softwaretests suchen**
- ★ **Wenn Sie Ihr eigenes System möglichst effizient einsetzen wollen**

dann ist »Computer persönlich«, das aktuelle Fachmagazin für Personal Computer, genau Ihre Zeitschrift.

Die konsequente Ausrichtung auf professionelle Anwendungen bietet Ihnen alle wichtigen Informationen.

Von Profis für Profis!

»Computer persönlich« gibt es alle 14 Tage neu bei Ihrem Zeitschriftenhändler oder im Computer-Fachgeschäft.

PC MAGAZIN

Einzige Wochenzeitung für Personal Computer im IBM-Standard.

Sie beschäftigen sich beruflich oder privat mit dem Einsatz und der Anwendung von Personal Computern?

Sie sind an aktuellen, professionellen Informationen über IBM-PCs, kompatible Systeme und deren professionellen Einsatz interessiert? Dann ist das PC Magazin genau auf Ihre persönlichen Bedürfnisse zugeschnitten.

Es wird von anerkannten und erfahrenen Fachjournalisten für professionelle Anwender und Fachleute geschrieben.

Es berichtet jede Woche ausschließlich über Computer im IBM-Standard und kompatible Systeme, über Hard- und Softwareneuheiten. Es bringt ausführliche Testberichte und gibt Ihnen wichtige Informationen über Netzwerke sowie die PC/Host-Verbindung.

Nur diese Spezialisierung ermöglicht eine gezielte Berichterstattung und bietet genügend Raum, um auf Anwenderprobleme spezifisch eingehen zu können.

Von Profis für Profis!

Und das jeden Mittwoch neu bei Ihrem Zeitschriftenhändler oder im Computer-Fachgeschäft.

GUTSCHEIN

für ein kostenloses Probeexemplar

Senden Sie mir die neueste Ausgabe der von mir angekreuzten Zeitschrift kostenlos als Probeexemplar:

☐ **COMPUTER PERSÖNLICH**
Wenn mir Computer persönlich zusagt und ich es regelmäßig weiterbeziehen möchte, brauche ich nichts zu tun: Ich erhalte Computer persönlich dann regelmäßig alle 14 Tage per Post frei Haus geliefert und bezahle pro Jahr nur DM 98,—. Zustellung und Postgebühren übernimmt der Verlag.

☐ **PC-MAGAZIN**
Wenn mir das PC-Magazin zusagt und ich es regelmäßig weiterbeziehen möchte, brauche ich nichts zu tun: Ich erhalte mein PC-Magazin dann regelmäßig jede Woche per Post frei Haus geliefert und bezahle pro Jahr nur DM 155,—. Zustellung und Postgebühren übernimmt der Verlag.

Vorname/Name

Straße PLZ/Ort

Datum 1. Unterschrift

Mir ist bekannt, daß ich diese Bestellung innerhalb von 8 Tagen bei der Bestelladresse widerrufen kann und bestätige dies durch meine zweite Unterschrift. Zur Wahrung der Frist genügt die rechtzeitige Absendung des Widerrufs.

Datum 2. Unterschrift

Gutschein ausfüllen, ausschneiden, auf Postkarte kleben und einsenden an:

CS285

QUEWE gemacht. Sie wird so verändert, daß die Ausgabe an den Drucker interruptgesteuert wird und somit unabhängig vom Betriebszustand des Schneiders ist.

(E. Hinrichsen und J. Völker)

```
10 ' S P O O L E R  >CPC 464<
20 ' (c) Eckhard Hinrichsen
30 '     Daniel-Hinsche-Str. 6
40 '     2050 Hamburg 80
50 ' Der Spooler wird mit :SPOOLER,Anfan
gsadr.,Endadr. aufgerufen.
60 ' Die Adressen geben dabei Anfang und
 Ende des Speicherbereiches
70 ' an, der als Druckerpuffer genutzt w
ird.
80 ' Wird beim Aufruf eine oder beide de
r Adressen weggelassen, so
90 ' werden dafuer Standardwerte eingese
tzt.
100 ' Das sind: Untergrenze= &8000
110 '            Obergrenze = &A3FE
120 ' Nach Aufruf des Spoolers laufen sa
emtliche Druckerausgaben ueber
130 ' den Puffer.
140 ' Mit :SPOOLEROFF wird die Routine a
bgeschaltet, d.h. die Druckerausgabe
150 ' erfolgt wieder normal.
160 '
170 CLS:MEMORY &7FFF
180 ad=42144
190 sw=0
200 FOR i=0 TO 15
210 READ a$
220 FOR k=1 TO LEN(a$) STEP 2
230 w=VAL("&"+MID$(a$,k,2))
240 POKE ad,w
250 sw=sw+w
260 ad=ad+1
270 NEXT
280 NEXT
290 IF sw=56098 THEN PRINT "Alles in Ord
nung" ELSE PRINT"Pruefsummenfehler":END
300 '
310 CALL &A4A0 ' Dem Rechner die neuen B
efehle beibringen.
320 '
330 DATA 01A9A421C3A4C3D1BCB1A4C3DFA4C32
4A553504F4F4C45D253504F4F4C45
340 DATA 524F46C6000000A9A40000550000833
7A500C1C92BE5EDFEA3FEA30080FE
350 DATA A3F2BFED73DDA4FE002811FE012003E
B180DFE02DD6603DD6E0228062100
360 DATA 8011FEA3CD8CA522D9A4ED53DBA421C
7A41137A5010083CDE0BC214DA522
370 DATA F2BD2ADBA422D5A422D7A4C921C7A41
137A5010083CDE6BC21F80722F2BD
380 DATA C9CD81A5CBCD2EBDD82AD5A4CD72A57
E22D5A4CD31BDC9C5D5E5F32AD7A4
390 DATA E5CD72A522D7A4CD81A5E1C268A522D
7A4B718052AD7A47737FBE1D1C1C9
400 DATA 2BE5ED5BD9A4B7ED52E1C02ADBA4C92
AD5A4ED5BD7A4B7ED52C9E521FFA3
410 DATA A7ED52E1E5FA2DA6A7ED52F2ACA5ED4
B7BAEE1E5A7ED42FAE4A5E1C9CD5C
420 DATA A607182045205220520 4F2052203A2
0416E66616E6773616472657365
430 DATA 203E20456E6461647265737365202211
80D0AFFED7BDDA4C9CD5CA6071820
440 DATA 4520522052204F2052203A205370656
9636865726F6265726772656E7A65
450 DATA 206D6974204D454D4F5259206865727
56E7465727365747A656E2021180D
460 DATA 0AFFED7BDDA4C9CD5CA607182045205
220522042052203A20456E646164
470 DATA 726573736520 7A7520686F636820211
80A0DFFED7BDDA4C9E37E23FEFF28
480 DATA 05CD5ABB18F5E3C9
```

Listing 1. Der Basic-Lader für die Maschinencode-Routine

```
10 MEMORY &7FFF:LOAD "!SPOOLER.BIN":CALL
 &A4A0:DELETE 10-20
20 SAVE "SPOOLER.LOAD":SAVE "!SPOOLER.BI
N",B,&A4A0,&1CA
```

Listing 2. Das Laden und Speichern ist kinderleicht

Menü-gesteuerte Diskettenstation

Mit »Disk-Helper« wird die Arbeit mit der Diskettenstation sehr einfach. Sie müssen keine umständlichen Anweisungen mehr eintippen, sondern steuern alle Befehle über das Menü. Zwei Versionen gibt es: eine für das Laufwerk von Schneider und eine für das von Vortex.

Die Routine »Disk-Helper« ist eine Hilfe für alle Besitzer einer Diskettenstation für den Schneider, sowohl der DD1 von Schneider (Listing 1), wie auch dem Vortex-Laufwerk (Listing 2). Umständliche Befehle müssen nicht mehr eingegeben werden. Sie werden alle mit einem Tastendruck angesprochen. Eventuelle Parameter oder Filenamen werden vom Computer angefordert. Neben den üblichen DOS-Befehlen gibt es noch die Anweisungen »Exit« zum Beenden des Programms und »Type«, die — wie unter CP/M — Textfiles auf den Bildschirm ausgibt.

(Martin Huber)

```
10 REM ******************************
20 REM * Disk-Helper fuer DD1        *
30 REM *    (C) 1985 by Haiteck & Co *
40 REM *        Holzleiten 1         *
50 REM *        8318 Bodenkirchen 1  *
60 REM ******************************
70 REM
80 MODE 2:i$=CHR$(24)
90 INK 0,0:INK 1,26:PEN 1:PAPER 0
100 :
110 REM ----------Routine zum Abfangen v
on Diskfehlern
120 POKE &AC01,&C3:POKE &AC02,&B6:POKE &
AC03,&CA
130 ON ERROR GOTO 1150
140 :
150 REM ----------Routine zum Lesen eine
s einzigen Zeichens
160 MEMORY HIMEM-20:POKE HIMEM+1,&CD:POK
E HIMEM+2,&80:POKE HIMEM+3,&BC
170 POKE HIMEM+4,&32:POKE HIMEM+5,UNT(HI
MEM+9) AND 255
180 POKE HIMEM+6,INT((HIMEM+9)/256):POKE
 HIMEM+7,&C9:highmem=HIMEM
190 :
200 REM ----------Titelbild ausgeben
210 PRINT i$" Disk-Helper
            (C) 1985 by Haiteck & Co
            "i$
220 PRINT:PRINT"Befehle: "i$"A"i$":,"i$"
B"i$":,"i$"C"i$"at,"i$"D"i$"irectory,"i$
"E"i$"rase,CP/"i$"M"i$","i$"R"i$"ename,"
i$"S"i$"elect"
230 PRINT"          "i$"L"i$"OAD,R"i$"U"i
$"N,E"i$"X"i$"IT,T"i$"Y"i$"PE"
240 WINDOW 1,80,6,25
250 b$="abcdefmorstluxy":s$="a":u$="0":G
OSUB 790
260 :
270 REM ----------Eingabeschleife
280 PRINT">";:a$="":WHILE a$="":a$=INKEY
$:WEND
290 IF INSTR(b$,LOWER$(a$))=0 THEN PRINT
 CHR$(7):GOTO 280 ELSE PRINT " "i$a$i$
300 ON INSTR(b$,LOWER$(a$)) GOSUB 330,36
0,390,460,540,310,650,310,710,760,310,88
0,910,940,970
310 PRINT"Ok":GOTO 280
320 :
330 REM ----------A:
340 s$="a":GOTO 790
350 :
360 REM ----------B:
370 s$="b":GOTO 790
380 :
390 REM ----------Cat
400 GOSUB 1070 'Bereich holen
410 IF NOT fl THEN CAT:RETURN
420 PRINT STRING$(7,10):uu$=u$:FOR u=0 T
O 15:LOCATE 1,13
430 GOSUB 780:CAT:IF u<>15 AND VPOS(#0)=
20 THEN PRINT STRING$(4,10)
440 NEXT:u$=uu$:GOTO 790
450 :
460 REM ----------Directory
470 GOSUB 1070 'Bereiche holen
480 GOSUB 1120 'Wildcard holen
490 IF NOT fl THEN !DIR,@wc$:RETURN
500 PRINT STRING$(7,10):uu$=u$:FOR u=0 T
O 15:LOCATE 1,13
510 GOSUB 780:!DIR,@wc$:IF u<>15 AND VPO
S(#0)=20 THEN PRINT STRING$(4,10)
520 NEXT:u$=uu$:GOTO 790
530 :
540 REM ----------Erase
550 GOSUB 1070 'Bereiche holen
560 GOSUB 1120 'Wildcard holen
570 IF NOT fl THEN !ERA,@wc$:RETURN
580 PRINT STRING$(3,10):uu$=u$:FOR u=0 T
O 15:LOCATE 1,18:PRINT"User"u
590 GOSUB 780:!ERA,@wc$:IF u<>15 AND VPO
S(#0)=20 THEN PRINT STRING$(3,10)
600 NEXT:u$=uu$:GOTO 790
610 :
620 REM ----------Format
630 PRINT"FORMAT":INPUT"Press Enter",a$:
!FORMAT,VAL(s$):RETURN
640 :
650 REM ----------CP/M
660 !CPM
670 :
680 REM ----------Code
690 INPUT"Codenummer: ",a$:!CODE,INT(VAL
(a$)):RETURN
700 :
710 REM ----------Rename
720 INPUT"Alter Filename: ",afn$
730 INPUT"Neuer Filename: ",nfn$
740 !REN,@nfn$,@afn$:RETURN
750 :
760 REM ----------Select
770 INPUT"Usernummer (0...15)";u:IF u<0
OR u>15 THEN 770
780 u$=MID$(STR$(INT(u)),2,LEN(STR$(INT(
u)))-1)
790 !DRIVE,@s$:!USER,VAL(u$):RETURN
800 :
810 REM ----------Attribut
820 INPUT"Filename: ",nfn$
830 PRINT"Welches Attribut? "i$"S"i$"YS
 "i$"D"i$"IR  "i$"R"i$"/O  R/"i$"W"i$" "
840 a$="":WHILE a$="":a$=LOWER$(INKEY$):
WEND
850 IF INSTR("sdrw",a$)=0 THEN PRINT CHR
$(7);:GOTO 840
860 !ATTRIBUT,@nfn$,@a$:RETURN
870 :
880 REM ----------Load
890 INPUT"Filename: ",nfn$:LOAD nfn$:STO
P
900 :
910 REM ----------Run
920 INPUT"Filename: ",nfn$:RUN nfn$:STOP
930 :
940 REM ----------Exit
950 POKE &AC01,&C9:PRINT:END:RUN
960 :
970 REM ----------Type
980 INPUT"Filename: ",nfn$:nfn$="  "+nfn
$+"  "
990 OPENIN nfn$
1000 IF EOF OR INKEY$<>"" THEN CLOSEIN:P
RINT:RETURN
1010 CALL highmem+1:a=PEEK(highmem+9)
1020 IF a>31 THEN PRINT CHR$(a);:GOTO 10
00
1030 IF a=13 THEN PRINT" <CR> ";:GOTO 10
00
1040 IF a=10 THEN PRINT"<LF> ":GOTO 1000
1050 PRINT" ^"CHR$(a+64)" ";:GOTO 1000
1060 :
1070 REM ----------Bereich holen
1080 PRINT"Alle Userbereiche? (J/N)":a$=
"":WHILE a$="":a$=LOWER$(INKEY$):WEND
1090 IF a$<>"j" AND a$<>"n" THEN PRINT C
HR$(7);:GOTO 1080
1100 fl=(a$="j"):RETURN
1110 :
1120 REM ----------Wildcard holen
1130 INPUT"Wildcard: ",wc$:RETURN
1140 :
1150 REM ----------Disk-Fehler
1160 PRINT:PRINT"Diskfehler ist aufgetre
ten.
1170 PRINT"Operation wird abgebrochen.
1180 RESUME 310
```

Listing 1. »Disk-Helper« für das Laufwerk von Schneider

```
10 REM ******************************
20 REM * Disk-Helper fuer Vortex     *
30 REM *    (C) 1985 by Haiteck & Co *
40 REM *        Holzleiten 1         *
50 REM *        8318 Bodenkirchen 1  *
60 REM ******************************
70 REM
80 MODE 2:i$=CHR$(24)
90 INK 0,0:INK 1,26:PEN 1:PAPER 0
100 :
110 REM ----------Routine zum Abfangen v
on Diskfehlern
120 POKE &AC01,&C3:POKE &AC02,&B6:POKE &
AC03,&CA
130 ON ERROR GOTO 1150
```

Listing 2. »Disk-Helper« für das Laufwerk von Vortex

```
140 :
150 REM ----------Routine zum Lesen eine
s einzigen Zeichens
160 MEMORY HIMEM-20:POKE HIMEM+1,&CD:POK
E HIMEM+2,&80:POKE HIMEM+3,&BC
170 POKE HIMEM+4,&32:POKE HIMEM+5,UNT(HI
MEM+9) AND 255
180 POKE HIMEM+6,INT((HIMEM+9)/256):POKE
 HIMEM+7,&C9:highmem=HIMEM
190 :
200 REM ----------Titelbild ausgeben
210 PRINT i$" Disk-Helper
          (C) 1985 by Haiteck & Co
          "i$
220 PRINT:PRINT"Befehle: "i$"A"i$":,"i$"
B"i$":,"i$"C"i$"at,"i$"D"i$"irectory,"i$
"E"i$"rase,"i$"F"i$"ormat,CP/"i$"M"i$",C
"i$"O"i$"DE,"i$"R"i$"ename,"i$"S"i$"elec
t,A"i$"T"i$"tribut"
230 PRINT"            "i$"L"i$"OAD,R"i$"U"i
$"N,E"i$"X"i$"IT,T"i$"Y"i$"PE"
240 WINDOW 1,80,6,25
250 b$="abcdefmorstluxy":s$="a":u$="0":G
OSUB 790
260 :
270 REM ----------Eingabeschleife
280 PRINT">";:a$="":WHILE a$="":a$=INKEY
$:WEND
290 IF INSTR(b$,LOWER$(a$))=0 THEN PRINT
 CHR$(7):GOTO 280 ELSE PRINT " "i$a$i$
300 ON INSTR(b$,LOWER$(a$)) GOSUB 330,36
0,390,460,540,620,650,680,710,760,810,88
0,910,940,970
310 PRINT"Ok":GOTO 280
320 :
330 REM ----------A:
340 s$="a":GOTO 790
350 :
360 REM ----------B:
370 s$="b":GOTO 790
380 :
390 REM ----------Cat
400 GOSUB 1070 'Bereich holen
410 IF NOT f1 THEN CAT:RETURN
420 PRINT STRING$(7,10):uu$=u$:FOR u=0 T
O 15:LOCATE 1,13
430 GOSUB 780:CAT:IF u<>15 AND VPOS(#0)=
20 THEN PRINT STRING$(4,10)
440 NEXT:u$=uu$:GOTO 790
450 :
460 REM ----------Directory
470 GOSUB 1070 'Bereiche holen
480 GOSUB 1120 'Wildcard holen
490 IF NOT f1 THEN !DIR,@wc$:RETURN
500 PRINT STRING$(7,10):uu$=u$:FOR u=0 T
O 15:LOCATE 1,13
510 GOSUB 780:!DIR,@wc$:IF u<>15 AND VPO
S(#0)=20 THEN PRINT STRING$(4,10)
520 NEXT:u$=uu$:GOTO 790
530 :
540 REM ----------Erase
550 GOSUB 1070 'Bereiche holen
560 GOSUB 1120 'Wildcard holen
570 IF NOT f1 THEN !ERA,@wc$:RETURN
580 PRINT STRING$(3,10):uu$=u$:FOR u=0 T
O 15:LOCATE 1,18:PRINT"User"u
590 GOSUB 780:!ERA,@wc$:IF u<>15 AND VPO
S(#0)=20 THEN PRINT STRING$(3,10)
600 NEXT:u$=uu$:GOTO 790
610 :
620 REM ----------Format
630 PRINT"FORMAT":INPUT"Press Enter",a$:
!FORMAT,VAL(s$):RETURN
640 :
650 REM ----------CP/M
660 !CPM
670 :
680 REM ----------Code
690 INPUT"Codenummer: ",a$:!CODE,INT(VAL
(a$)):RETURN
700 :
710 REM ----------Rename
720 INPUT"Alter Filename: ",afn$
730 INPUT"Neuer Filename: ",nfn$
740 !REN,@nfn$,@afn$:RETURN
750 :
760 REM ----------Select
770 INPUT"Usernummer (0...15)";u:IF u<0
OR u>15 THEN 770
780 u$=MID$(STR$(INT(u)),2,LEN(STR$(INT(
u))-1)
790 q$=u$+s$+":":!SELECT,@q$:RETURN
800 :
810 REM ----------Attribut
820 INPUT"Filename: ",nfn$
830 PRINT"Welches Attribut? "i$"S"i$"YS
 "i$"D"i$"IR  "i$"R"i$"/O  R/"i$"W"i$" "
840 a$="":WHILE a$="":a$=LOWER$(INKEY$):
WEND
850 IF INSTR("sdrw",a$)=0 THEN PRINT CHR
$(7);:GOTO 840
860 !ATTRIBUT,@nfn$,@a$:RETURN
870 :
880 REM ----------Load
890 INPUT"Filename: ",nfn$:LOAD nfn$:STO
P
900 :
910 REM ----------Run
920 INPUT"Filename: ",nfn$:RUN nfn$:STOP
930 :
940 REM ----------Exit
950 POKE &AC01,&C9:PRINT:END:RUN
960 :
970 REM ----------Type
980 INPUT"Filename: ",nfn$:nfn$="  "+nfn
$+"  "
990 OPENIN nfn$
1000 IF EOF OR INKEY$<>"" THEN CLOSEIN:P
RINT:RETURN
1010 CALL highmem+1:a=PEEK(highmem+9)
1020 IF a>31 THEN PRINT CHR$(a);:GOTO 10
00
1030 IF a=13 THEN PRINT" <CR> ";:GOTO 10
00
1040 IF a=10 THEN PRINT"<LF> ":GOTO 1000
1050 PRINT" ^"CHR$(a+64)" ";:GOTO 1000
1060 :
1070 REM ----------Bereich holen
1080 PRINT"Alle Userbereiche? (J/N)":a$=
"":WHILE a$="":a$=LOWER$(INKEY$):WEND
1090 IF a$<>"j" AND a$<>"n" THEN PRINT C
HR$(7);:GOTO 1080
1100 f1=(a$="j"):RETURN
1110 :
1120 REM ----------Wildcard holen
1130 INPUT"Wildcard: ",wc$:RETURN
1140 :
1150 REM ----------Disk-Fehler
1160 PRINT:PRINT"Diskfehler ist aufgetre
ten.
1170 PRINT"Operation wird abgebrochen.
1180 RESUME 310
```

Listing 2. »Disk-Helper« für das Laufwerk von Vortex (Schluß)

CP/M — Das zweite Gesicht Ihres Schneiders

Der Weltstandard für 8-Bit-Computer heißt CP/M. Die meisten Programme wurden für dieses Betriebssystem geschrieben. Sicher wollen auch Sie wissen, wie CP/M funktioniert.

CP/M verwaltet nicht nur die gesamte Hardware des Computers, sondern bietet auch komfortable Routinen, um in Assembler zu programmieren. Dazu stehen ein Editor (EDCOM), ein Assembler (ASM.
COM) und ein Programm zur Fehlersuche (DDT.COM) zur Verfügung. Was dem Benutzer mit CP/M geboten wird, das war den Heimcomputer-Besitzern bisher versperrt.

Es gibt zwei verschiedene Befehlsarten unter CP/M. Systemresidente stehen nach dem Laden des Betriebssystems im Speicher des Computers zur Verfügung. Diskresidente (die zweite Art) Anweisungen befinden sich auf der Systemdiskette und müssen, bevor sie bearbeitet werden können, geladen werden.

CP/M meldet sich zu Anfang immer mit »A>«. Damit wird angezeigt, daß das Laufwerk A aktiv ist und daß der Computer auf einen Befehl wartet. Dateien auf der Diskette, die mit dem Zusatz COM versehen sind, werden vom Betriebssystem als Befehle oder als sofort ausführbare Programme angesehen und beim Aufruf sofort gestartet.

Wollen wir jetzt unter CP/M einen Befehl eingeben, so müssen wir uns verschiedene Dinge überlegen. Handelt es sich um eine systemresidente Anweisung, so gibt es keine Probleme mit dem Laufwerk. Wir geben einfach den Namen (zum Beispiel »DIR«) ein, schließen ihn mit der ENTER-Taste ab und er wird ausgeführt. Anders ist das bei den diskresidenten Kommandos. Zuerst müssen wir klären — sofern wir zwei Diskettenstationen benutzen — in welchem Laufwerk sich die Systemdiskette mit dem Befehl befindet.

Ein Programm auf der nicht aktuellen Station kann auf zwei Arten angesprochen werden. Entweder definiert man dieses Laufwerk als aktuelle Station (beispielsweise mit »B:«) und ruft dann das Programm auf oder man stellt dem Namen das Laufwerk voran (»B:Name).

Ein Befehl kann sich aber auch auf ein bestimmtes Laufwerk beziehen. Dieses wird dann hinter der Anweisung angegeben. So gibt zum Beispiel die Sequenz »DIR B:« das Inhaltsverzeichnis der Diskette in der zweiten Station aus, obwohl das erste Laufwerk als Hauptstation angewählt ist.

Wenden wir uns nun der Praxis zu. Schalten Sie Ihren Schneider-Computer ein und geben Sie »ICPM« ein. Beim CPC 464 und beim CPC 664 meldet sich CP/M mit »CP/M 2.2 — Amstrad Consumer Electronics plc« und »A>«. Beim neuen CPC 6128 wird neben CP/M 2.2 auch die Version 3.1 mitgeliefert, die ein etwas anderes Bild zeigt.

Bevor wir weitermachen, sollten wir uns eine Sicherheitskopie unserer Systemdiskette machen, damit wir bei unseren Versuchen nicht irgendwelche Dateien zerstören. Dieses »Backup« macht man mit dem (diskresidenten) Befehl »DISKCOPY«, der im Handbuch erklärt ist. Alle weiteren Untersuchungen wollen wir nun mit unserer Kopie machen.

Insgesamt gibt es sechs systemresidente Anweisungen. Mit »DIR« wird der Inhalt der Diskette ausgegeben. Man kann bestimmte Vorgaben geben, so daß nur Teile gelistet werden. So gibt beispielsweise »DIR B:*.BAS« alle Dateien mit Zusatz »BAS« auf Laufwerk B an. Der »*« steht als sogenannte Wildcard und ist als Platzhalter für eine ganze Zeichenfolge zu verstehen. Die zweite Art von Wildcard ist das »?«, das einen bestimmten Buchstaben ersetzt. So würde die Anweisung »DIR NAME????.BAS« alle die Dateien auf dem Bildschirm ausgeben, deren Name mit »Name« anfängt und die den Zusatz »BAS« tragen.

»ERA« wird ähnlich benutzt wie der DIR-Befehl. Allerdings werden jetzt die Dateien nicht aufgelistet, sondern gelöscht. Ist eine Datei schreibgeschützt, so wird der Befehl abgebrochen, bevor größeres Unheil angerichtet wird. Nur bei »ERA*.*« fragt der Computer, ob wirklich die ganze Diskette gelöscht werden soll.

Mit »REN neuer Name = alter Name« können bestehende Dateien umbenannt werden. Wildcards dürfen nicht benutzt werden, denn sie würden eine Fehlermeldung verursachen. Der Befehl kann sich aber auf das jeweils andere Laufwerk beziehen. Vor der Namensangabe muß dann der Buchstabe für diese Station mit nachfolgendem Doppelpunkt stehen.

Der vierte systemresidente Befehl heißt »TYPE Name«. Er stellt ASCII-Dateien, das heißt reine Datenfelder auf dem Bildschirm dar. Wildcards dürfen natürlich nicht benutzt werden, denn wie sollen mehrere Dateien gleichzeitig auf dem Bildschirm erscheinen. Werden Nicht-ASCII-Dateien angesprochen, so kann es zu unvorhersehbaren Ereignissen auf dem Bildschirm kommen, da bestimmte (Nicht-ASCII-) Zeichen als Steuercodes für die Bildausgabe interpretiert werden.

Mit »SAVE x Name« sichert man Programme oder Daten, die im Speicher stehen, auf Diskette. x bezeichnet dabei die Länge des Feldes (in Einheiten von 256 Byte langen Blöcken). Eine Anfangsadresse braucht nicht angegeben zu werden, da CP/M-Programme immer an Adresse 0100 hex beginnen. Bei allen Namen kann übrigens ein beliebiger Zusatz definiert werden. Aber Vorsicht! Bestimmte Zusätze werden von CP/M mit einer festen Bedeutung bedacht. »COM« zum Beispiel als Routine, die direkt ausgeführt werden soll.

Wie bei Amsdos kann man auch unter CP/M seine Diskette in verschiedene Bereiche unterteilen. Der Befehl dazu lautet »USER x«, wobei x zwischen 0 und 15 liegen darf. Programme oder Daten, die unter einer User-Ebene abgelegt wurden, können auch nur unter dieser Nummer gelistet oder aufgerufen werden. In allen anderen Ebenen tauchen sie nicht mehr auf.

Die diskresidenten Kommandos stehen — wie schon gesagt — nur auf der Diskette zur Verfügung und müssen nach dem Aufruf erst einmal geladen werden, bevor sie ausgeführt werden können. Einige von diesen Befehlen gibt es nur auf dem Schneider. »SYSGEN«, »BOOTGEN«, »FILECOPY«, »COPYDISC«, »DISCCOPY«, »CHKDISK«, »DISKCHK«, »FORMAT«, »SETUP«, »MOVCPM«, »CSAVE«, »CLOAD« und »AMSDOS« sind Hilfsprogramme, die bei ande-

ren CP/M-Computern nicht oder in anderer Form vorhanden sind. Da sie im Handbuch sehr gut erklärt werden, wollen wir sie hier nicht mehr behandeln. Statt dessen wollen wir uns mehr mit den Routinen beschäftigen, die CP/M von Hause aus besitzt und die nicht oder nicht ausführlich beschrieben sind.

Als ersten Befehl wollen wir die Anweisung »DUMP Name« betrachten. Ähnlich wie mit »TYPE« wird eine Datei auf dem Bildschirm ausgegeben. Einige Unterschiede bestehen allerdings doch. Es können alle Felder von Diskette gelesen und auf den Bildschirm gebracht werden. Die einzelnen Zeichen werden nämlich mit ihren hexadezimalen Werten dargestellt und nicht im Klartext. Somit werden auch eventuelle Steuercodes nicht als solche erkannt. Um dies zu verdeutlichen, zeigen wir Ihnen im folgenden die Ausgabe einer Datei einmal mit »TYPE« und einmal mit »DUMP«. Der Befehl »TYPE DUMP.ASM« zaubert folgendes Bild auf den Monitor:

```
   FILE DUMP PROGRAMM.
READS AN INPUT FILE AND
PRINTS IN HEX
COPYRIGHT (C) 1975, 1976, 1977,
1978
DIGITAL RESEARCH
BOX 579. PACIFIC GROVE
CALIFORNIA, 93950
.....
```

Mit »DUMP DUMP.ASM« schaut das Ganze dann so aus:

```
0000 3B 09 46 49 4C 45 20 44 55 4D
50 20 50 52 4F 47
0010 52 41 4D 2C 20 52 45 41 44 53
20 41 4E 20 49 4E
0020 50 55 54 20 46 49 4C 45 20 41
4E 44 20 50 52 49
0030 4E 54 53 20 49 .. ..
```

Will man Auskunft haben über die Systemkonfiguartion mit der man arbeitet, so gibt es hierfür das Kommando »STAT«. Verschiedene Zusätze konkretisieren die gwünschte Auskunft. »STAT A:« zeigt den Disketten-Status und den freien Speicherplatz an. Ohne spezifiziertes Laufwerk bezieht sich der Befehl automatisch auf die aktuelle Station. Mit »STAT *COM« erhält man erweiterte Inhaltsverzeichnis-Informationen für einen bestimmten Bereich (in diesem Beispiel für die COM-Dateien). Wildcards sind bei diesem Befehl gültig und so kann man verschiedenste Auswahlkriterien treffen. Gibt man nach diesem Befehl (beispielsweise »STAT EX1.BAS«) einen Zusatz ein, so kann der Zustand der Dateien geändert werden. Mit »$R/O« (die gesamte Anweisung lautet dann »STAT EX1.BAS $R/O«) wird eine Datei auf »Nur lesbar« gesetzt. Die Umkehrung (»Schreib-/Lese«-Status) erfolgt mit »$R/W«. Der »System«-Status wird mit »$SYS« aufgerufen und bewirkt, daß die Datei nicht mehr gelistet werden kann. »$DIR« setzt wieder den »Inhaltsverzeichnis«-Status und ist damit die Umkehrung des Befehls »$SYS«.

Mit »STAT DEV:« bekommt man Informationen über den Hardware-Zustand des Systems. Folgende Geräte werden überprüft: CON steht für Konsole und bezeichnet den Ausgabekanal, auf den alle normalen Ein- und Ausgaben gemacht werden. RDR bezeichnet den Eingabekanal, der im Grundzustand mit »TTY« (serielle Schnittstelle) belegt ist. PUN gibt den Ausgabekanal an, der standardmäßig auch mit TTY belegt ist. LST zeigt im Normalzustand LPT (Line Printer) an. Dies ist der Standardkanal für den Drucker, an dem zur Zeit die parallele Schnittstelle liegt.

Gibt man den Zusatz »VAL:« an, so bekommen Sie die einzelnen Kommandos von »STAT« ausgedruckt. Haben Sie die Systemdiskette im aktuellen Laufwerk, so müssen auf dem Bildschirm folgende Informationen stehen:

```
Temp R/O Disk: d:=R/O
Set Indicator: d:filename.typ $R/O
$R/W $SYS $DIR
Disk Status : DSK: d:DSK:
User Status : USR:
Iobyte Assign:
CON:  = TTY:  CRT:  BAT:  UC1:
RDR:  = TTY:  PTR:  UR1:  UR2:
PUN:  = TTY:  PTP:  UP1:  UP2:
LST:  = TTY:  CRT:  LPT:  UL1:
```

Was verbirgt sich nun hinter diesen scheinbar wirren Buchstabenkombinationen?

Mit »STAT d: = R/O« (d steht im folgenden immer für den Kennbuchstaben des Laufwerks) setzen Sie das angesprochen Laufwerk in den »Nurlesezustand«. Diese Anweisung heißt »Temp R/O Disk« und sorgt dafür, daß die Diskette zeitweise (temporär) nicht beschrieben werden kann. Mit »STAT d:=R/W« wird dieser Zustand wieder aufgehoben. Wie man den Indikator für ein bestimmtes Datenfeld setzt, das haben Sie schon am Anfang dieses Abschnitts gelesen. Damit lassen sich Dateien für das Directory sichtbar oder unsichtbar machen und als »Nurlesedateien« definieren.

Der Diskettenstatus kann mit »STAT DSK:« abgefragt werden. Auf Ihrem Bildschirm muß jetzt erscheinen:

```
A:    Drive Characteristics
1368: 128 Byte Record Capacity
171:  Kilobyte Drive Capacity
64:   32 Byte Directory Entries
64:   Checked Directory Entries
128:  Records/ Extent
8:    Records/ Block
36:   Sectors/ Track
2:    Reserved Tracks
```

Den »User«-Status fragen Sie mit »STAT USR:« ab. Die Antwort gibt die derzeitige User-Ebene an (»Active User : x « (wobei x den Bereich anzeigt)) und in welchen User-Ebenen Dateien abgelegt sind (»Active Files: x «).

Unter dem Stichwort »Iobyte Assign« findet man die möglichen Zuordnungen der Ein- und Ausgabekanäle. Die derzeitige Zuordung erfahren wir mit dem schon bekannten »STAT DEV:«. Normalerweise haben Sie jetzt auf dem Bildschirm stehen:

```
CON: IS   CRT:
RDR: IS   TTY:
RUN: IS   TTY:
LST: IS   LPT:
```

Wollen wir die Kanalzuordnung für den Drucker statt auf die parallel (LPT) nun auf die serielle Schnittstelle (TTY) legen, so ändern wir die Zuordnung mit » STAT LST:=TTY:«. Überprüfen wir die neue Zuordnung, so muß die letzte Zeile jetzt »LST: IS TTY:« lauten.

Informationen über bestimmte Dateien bekommt man mit »STAT Name«, wobei Wildcards — wie schon erwähnt — erlaubt sind. Die Ausgabe auf dem Monitor sieht dann für die Systemdiskette in etwa so aus:

```
Recs   Bytes   Ext Acc
2      1k      1 R/W A:AMSDOS.COM
64     8k      1 R/W A:ASM.COM
10     2k      1 R/W A:BOOTGEN.COM
... ... ... ...
... ... ... ...
... ... ... ...
6 1k 1 R/W A:XSUB.COM
Bytes Remaining On A: 26k
```

»Recs« bezeichnet die Anzahl der Blöcke zu je 128 Bytes.

»Bytes« gibt die tatsächliche Länge der Datei in 1-KByte-Schritten an.

»Ext« zeigt die Zahl der Einträge im Directory an. CP/M kann pro Eintrag im Inhaltsverzeichnis der Diskette nur eine bestimmte Länge pro Datei verwalten.

»Acc« zeigt den Zustand des Files an. »R/O« steht für schreibgeschützt und »R/W« für nicht schreibgeschützt. Als letztes findet man den Namen der Datei angegeben. Die letzte Zeile gibt an, wieviel Platz auf der Diskette noch frei ist.

Für die Peripherie-Verwaltung steht die Anweisung »PIP« (Peripheral Interchange Program) zur Verfügung. Es regelt den Datentransfer zwischen dem Computer und seiner Peripherie. Allgemein lautet das Kommando »PIP (Ziel)=(Quelle)«.

Von einen Laufwerk zum anderen kopiert man beispielsweise mit »PIP B:Name=A:Name«. Jetzt wird das entsprechende File vom Laufwerk A auf die Diskette im Laufwerk B geschrieben. Im Namen dürfen Wildcards benutzt werden, so daß man auch eine ganze Diskette oder Teile davon kopieren kann.

Die Varianten des Kommandos sind vielfältig. So kann der eigentliche Name oder dessen Erweiterung geändert werden, oder, wenn er gleich bleiben soll, kann er in der Befehlsfolge weggelassen werden. »PIP A:=B:TEST.BAS« schreibt so beispielsweise das Feld »Name« vom Laufwerk B, unter dem gleichen Namen auf das Laufwerk A. Gleiches macht, wenn B die aktuelle Diskettenstation ist, auch der Befehl »PIP A:=TEST.BAS«.

Es gibt dann zwei Felder mit dem gleichen Inhalt, aber unterschiedlichen Namen.

Soll eine Datei über die serielle Schnittstelle ausgegeben werden und diese liegt auf dem Kanal »PUN«, so lautet der Befehl dazu: »PIP PUN:=Name«. Man muß dabei aber beachten, daß sich nur ASCII-Files mit dem »PIP«-Kommando ausgeben lassen, da die Routine den Code 1A hex als »End of file« (Ende der Datei) interpretiert und damit die Übertragung abbricht.

Der umgekehrte Weg geht mit »PIP A:Name=RDR:«, wenn »RDR« als Eingang für die serielle Schnittstelle definiert ist. Dies kann man aber leicht mit dem oben erwähnten »STAT«-Kommando überprüfen. Weitere Variationen des »PIP«-Kommandos werden durch einen angehängten Parameter angegeben. »PIP A:neuer Name=A:alter Name:P:« ist ein typisches Aussehen eines solchen Befehls. Die folgende Tabelle gibt die möglichen Parameter und ihre Bedeutung an:

- Z — das höchstwertige Bit in jedem Byte wird auf Null gesetzt
- U — steht für Upper, das heißt alle ASCII-Zeichen werden in Großbuchstaben umgewandelt
- L — steht für Lower, das heißt alle ASCII-Zeichen werden in kleine Buchstaben verwandelt
- N — jeder Zeile wird eine 6stellige Nummer vorangestellt (mit vorangehenden Leerstellen auf sechs Stellen formatiert), die immer höhergezählt (von 1 an) wird und mit einem Doppelpunkt aufhört
- N2 — wie N, aber statt der Leerstellen werden Nullen vorangestellt
- Tx — Tabulatoren können frei gesetzt werden (unter CP/M normalerweise immer jede 8. Stelle)
- Dx — löscht den Rest einer Zeile, wenn diese länger als die vorgegebene Norm (x) ist
- F — löscht alle Seitenvorschübe in einem File
- Px — fügt bei jedem Seitenvorschub x Zeilen ein
- V — kontrolliert die übertragenen Daten bei einem Kopiervorgang
- E — gibt alle Zeichen, die kopiert werden auf den Bildschirm aus (Achtung: nur Text-Datei verwenden, da eventuelle Steuerzeichen den Bildschirmaufbau stören können)
- Gx — kopiert eine Datei aus dem User- Bereich x in den momentan aktiven User-Bereich

(Manfred Mayer)

Das neueste deutsche Grafik-Adventure für Ihren Commodore 64

Suchen Sie die Pforte zu einer anderen Welt! Beweisen Sie Ihren Spürsinn, denn der richtige Weg ist schwer zu finden, und überall lauern Gefahren!

- hochauflösende Grafik
- ausführliche Spielanweisungen
- riesiger Befehlsvorrat
- Eingabe von ganzen Sätzen möglich
- variabler Spielablauf

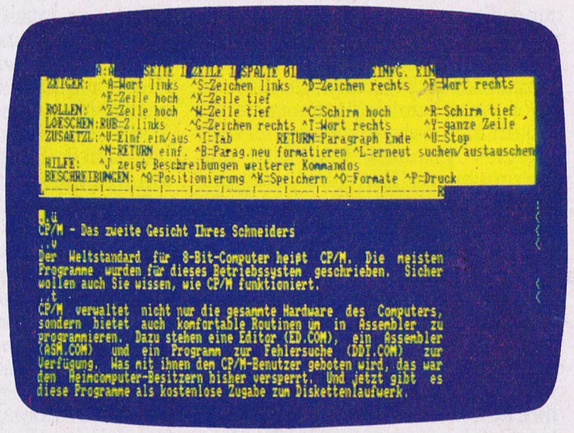
Markt&Technik
Verlag Aktiengesellschaft
Unternehmensbereich Buchverlag
Hans-Pinsel-Straße 2, 8013 Haar bei München
Schweiz: Markt & Technik Vertriebs AG, Kollerstraße 3, CH-6300 Zug, ☎ 042/223155
Österreich: Rudolf Lechner & Sohn, Heizwerkstraße 10, A-1232 Wien, ☎ 02 22/67 75 26

Gordon Saga
Best.-Nr. MD 240 A
DM 39,-* (Sfr. 35,50/öS 351,—)
*) inkl. MwSt. Unverbindliche Preisempfehlung.

Eine neue Dimension der Abenteuerspiele: Kein Spiel gleicht dem anderen — Sie geraten in Situationen, in denen Sie Ihre Spieltaktik völlig ändern müssen.
Überzeugen Sie sich selbst!
Happy Software gibt's beim Buchhändler, bei Horten, Quelle und im Computershop. Bestellkarten bitte an Ihren Buchhändler oder an eine unserer Depotbuchhandlungen.
Adressenverzeichnis am Ende des Heftes!

Wordstar — A Star was born

Tausenden von Sekretärinnen ist er aus dem Büro als leistungsfähiges Textverarbeitungsprogramm bekannt. Bisher war Wordstar für Heimcomputer-Besitzer finanziell unerschwinglich. Mit einem Preis von nur 199 Mark ist das jetzt anders.



Wordstar läuft unter CP/M. Deshalb muß dies zuerst von der Diskette geladen werden. Nun nur noch die Buchstabenkombination »WS« eingeben und mit einer Portion Interesse und Neugier kann's losgehen. Sobald der Schneider »Wordstar« geladen hat, »begrüßt« das Startmenü den Einsteiger. Übersichtlich listet es die Befehle auf, die notwendig sind, um überhaupt anfangen zu können. Dieses Menü erspart lästiges Wühlen im Handbuch, das gerade zu Beginn noch ein Buch mit, wenn auch nicht gerade sieben, doch mehreren Siegeln zu sein scheint. So finden wir gleich den für uns im Moment wichtigsten Befehl: »Datei eröffnen«. Dazu muß man einfach den Buchstaben d eingeben.

Wordstar: Etwas umständlich?

Der Bildschirm wechselt nun sein Aussehen und Wordstar fragt nach dem Namen der Datei, die Sie anlegen wollen. Geben Sie einen beliebigen Namen ein — wie er aufgebaut sein muß, das steht auf dem Bildschirm —, drücken Sie die ENTER-Taste und erst jetzt sind Sie mittendrin in der eigentlichen Textverarbeitung. Bis hierher scheint dem Anfänger — meist, gewohnt, sich an die Schreibmaschine zu setzen und loszulegen — alles etwas umständlich. Diese Meinung wird er jetzt aber schnell revidieren, wenn er näher mit den Fähigkeiten von Wordstar bekannt wird.

Das endgültige Aussehen des Bildschirms auf dem wir arbeiten werden, wird geprägt durch das Hauptkommando-Menü, das den oberen Teil des Bildschirms einnimmt. Der untere Freiraum steht für unseren Text zur Verfügung. Gerade am Anfang, wenn man noch nie mit Wordstar gearbeitet hat und noch weit davon entfernt ist, auch nur die wichtigsten Befehle im Kopf zu haben, wird man für diese Hilfe recht dankbar sein. Denn hier sind die Kommandos für Cursorbewegungen, Lösch-, Formatierungsbefehle und so weiter zu finden. Fangen wir an, unseren Text zu schreiben, so wird sehr schnell klar, daß diese »Gedächtnisstütze« das zeitraubende, mühselige Suchen im Handbuch vorerst unnötig macht. Vor allem, da dem »Wordstar«-Handbuch bedauerlicherweise ein Schlagwortregister fehlt. Ein Fauxpas, den vor allem jeder Anfänger bitter zu spüren bekommt.

Grundsätzlich sind beim Wordstar alle Kommandos — außer denen des Startmenüs — über eine mehrfache Tastenbedienung — CTRL plus Buchstabe — aufzurufen. Beim Schneider ersparen die speziellen Cursortasten dies zumindest für die Cursorbewegungen. Mit ihnen ist der Zeiger buchstabenweise nach rechts und links, sowie zeilenweise nach oben und unten zu steuern. Anfangs ist es auch recht ungewohnt und lästig, zwei Tasten drücken zu müssen, um eine Funktion zu erhalten. Doch die Resultate, die man damit erzielen kann, gleichen dies bei weitem wieder aus. Schreiben wir also unseren Text weiter und behalten dabei das Hauptkommando-Menü im Auge. Über kurz oder lang sind wir sicherlich mit dem nächsten Befehl konfrontiert: mit dem Löschen. Kein Problem, halten wir uns an die oben genannten Befehle und ... es klappt! Dasselbe gilt für alle anderen Kommandos ebenso. Es ist ohnehin empfehlenswert, einfach einmal alle Befehle durchzuprobieren. Ohne Übung kommt man nämlich mit dem tollsten Textverarbeitungssystem nicht zu Rande. Also auf geht's.

Sobald man die wichtigsten Befehle intus hat, das heißt je weniger man mit fortschreitender Übung des Anzeigenmenüs bedarf, desto störender wirkt es. Fast die gesamte obere Hälfte des Bildschirms geht nämlich dadurch für unseren Text verloren. Und das natürlich auf Kosten der Übersichtlichkeit. Dem ist aber eigentlich problemlos abzuhelfen — vorausgesetzt man weiß wie. Hier ist es erstmals unumgänglich sich intensiver ins Handbuch zu vertiefen. Denn so sehr man auch sucht, in den Hilfsmenüs ist nichts zu finden. Der Befehl dazu wird mit »CTRL J« und »H« aufgerufen. Wählt man diese Routine, so kann man den Umfang des Hilfsmenüs von 0 bis 3 regulieren. Bei Stufe 0 steht der komplette Bildschirm für den Text zur Verfügung. Ein wirkliches Erfolgserlebnis!

Die Tasten — ein Erfolgserlebnis

Hat man nun eine Weile vor sich hin gearbeitet, so sind die wichtigsten Kommandos geläufig. Es stellt sich die Frage: Wo liegt nun das große Plus von Wordstar? Bisher ist doch wirklich kein phänomenaler Unterschied zu einer komfortablen Schreibmaschine festzustellen! Ist das wirklich schon alles? Nein, bei weitem nicht. Entweder Sie schmökern nun im Handbuch weiter oder Sie rufen mit »CTRL J«, »CTRL Q«, »CTRL K«, »CTRL O« oder »CTRL P« (angezeigt im Hauptkommando-Menü) die restlichen Hilfsanzeigen auf. Diese demonstrieren dann

sämtliche Bearbeitungsbefehle, die Wordstar so zu bieten hat. Auf die beeindruckendsten Funktionen, die eine wirkliche Textverarbeitung eigentlich ausmachen, wollen wir nun eingehen. Die hervorstechendste — und auch charakteristischste — Eigenschaft mit einer Textverarbeitung per Computer, also auch mit Wordstar, ist folgende: Nachdem die Eingabe des gesamten Textes völlig problemlos zu bewerkstelligen war — die Zeilenschaltung entfiel ebenso wie die Trennung —, kann die fertige Datei jetzt im nachhinein in die gewünschte Fasson gebracht werden. Diese Aufgabe übernehmen in der Hauptsache die Formatierbefehle. Als erstes werden Sie wohl die Breite Ihres Textes definieren wollen. Durch die Befehle »CTRL O« und »L« wird der linke, durch »CTRL« und»R« der rechte Rand begrenzt. Jetzt präsentiert Ihnen Wordstar Ihre Datei rechts- und linksbündig als Blocksatz, das heißt mit Randausgleich. Durch den Befehl »CTRL O« und »J« der wie ein Ein-/Ausschalter zu bedienen ist, läßt sich das Ganze aber auch auf Flattersatz (ohne Randausgleich) umstellen.

Nun ergibt sich aber folgende Schwierigkeit: Dadurch, daß beim Schreiben keinerlei Trennungen durchgeführt wurden, entstanden zwischen den einzelnen Worten des öfteren große Zwischenräume, die einen etwas unschönen Eindruck machen. Wordstar bietet hier eine sogenannte »Trennhilfe« an, die während des Formatierens zur Anwendung kommt. Sie läßt sich über das »CTRL O«-Menü an- und ausschalten. Erteilt man das Formatkommando »CTRL B«, so erfolgt bei zu langen Worten an entsprechender Stelle ein Trennungsvorschlag durch den Cursor. Sind Sie damit einverstanden (das heißt stimmt es mit der Grammatik überein) so setzen Sie einen Trennstrich und das Formatieren geht weiter. Ist keine Trennung erwünscht, einfach noch einmal »CTRL B« eingeben. Leider muß jeder Block neu mit diesem Befehl formatiert werden. Aber immer noch besser als Neuschreiben wie bei einer Schreibmaschine.

Die Möglichkeit, die Randbegrenzungen beliebig fixieren zu können, stellt eine große Erleichterung bei Erstellung von Tabellen und ähnlichem dar. Wollen Sie beispielsweise eine Zahlenkolonne untereinander in der achten Spalte linksbündig auflisten, geben Sie ein: »CTRL O« und »L«, worauf Wordstar Sie nach der Spaltennummer fragt. Nun geben Sie »8« ein und die gesamten nachfolgenden Daten werden in die achte Spalte gerückt. Zum selben Thema gehört noch der Befehl »CTRL O« und »C«, der dann den Satz — oder das Wort — in der Mitte der Zeile zentriert. Bei Überschriften, ebenso wie bei Tabellen, ist dies eine große Hilfe.

Nie wieder Probleme mit Fußnote

Von Tabellen zu Fußnoten: Jeder, der schon einmal eine wissenschaftliche Arbeit getippt hat, wird von folgender Fähigkeit des Wordstar begeistert sein. Sie können die Fußnoten, die ansonsten mühselig unten angefügt werden müssen, einfach im Text mitschreiben. Lediglich das Kommando »FO« veranlaßt, daß der Rest der Zeile am Ende der Seite als Fußnote plaziert wird.

Was tun Sie eigentlich, wenn Sie 30 Seiten Text geschrieben haben und beim Durchlesen feststellen, daß ein Auszug von zwei Seiten thematisch viel besser an einer anderen Stelle passen würde? Normalerweise müßten Sie den gesamten Textblock noch einmal, nun an die richtige Stelle, schreiben. Dies bleibt Ihnen mit der »Blockverschiebung« erspart. Man markiert einfach Blockanfang und -ende, plaziert den Cursor an der gewünschten Stelle und gibt ein »CTRL K« und »V«. Wie durch Zauberei ist der Text da wo er sein soll.

Eine andere recht nützliche Sache ist das Kommando »Suchen/Tauschen«, mit dem man eine beliebige Textstelle ohne Schwierigkeiten immer wieder auffinden kann. Den Suchvorgang leitet das Kommando »CTRL Q« und »F« ein, indem Wordstar nach dem zu findenen Wort fragt. Dies kann eine bis zu 30 Zeichen lange Buchstabenkombination sein. Drücken Sie nun zweimal ENTER und der Cursor wird den gesuchten Begriff anzeigen. Zeichenfolgen, die Sie öfter suchen, können Sie mit der Merkerfunktion speichern — bis zu 10 gleichzeitig. Diese Funktion, kombiniert mit Tauschen, eröffnet noch weitere interessante Möglichkeiten. Will man zum Beispiel ein Wort durch ein anderes ersetzen, wendet man genau diese Befehlsfolge an. So können Sie beispielsweise, sollten Sie ihre Liebesbriefe nicht nur an »Michael«, sondern auch an »Robert« schicken wollen, einfach den einen Namen durch den anderen ersetzen. Angebrachter ist die Anwendung natürlich im Geschäftsbereich, wo bei immer gleich bleibenden Anschreiben lediglich eine Summe oder der Name geändert werden muß.

Auf diesem Grundgedanken basiert das Programm »Mailmerge«, mit dem Wordstar geliefert wird. Speziell für den Geschäftsbereich ist es nützlich. Es ist hier nur kurz erwähnt, da seine Anwendung komplex ist und diesen Artikel sehr aufblähen würde. »Mailmerge« bietet ein breites Anwendungsspektrum und erleichtert oder erspart einen großen Teil der im Büro anfallenden, gleichartigen Schreibarbeiten. Ausführliche Erklärungen kann man wieder aus dem mitgelieferten Handbuch beziehen. Oder aus anderer Literatur, wie es eigentlich zu empfehlen wäre.

Abschließend einiges zum Ausdrucken, um unsere Textdatei vom Bildschirm oder Diskette auf Papier zu bringen. Hervorhebungen wie Fettdruck und Unterstreichen sind einfach über CTRL-Befehle, die vor und hinter das zu markierende Wort gesetzt werden, zu machen. Details wie Seitenlänge, Seitenumbruch, Drucktypen, Zeilenabstand, Seitennummern und ähnliches, sofern Sie diesbezüglich von den vorgegebenen Werten abweichen wollen, sind mit sogenannten Punktkommandos zu steuern. Es sind mit Wordstar den Druckmöglichkeiten kaum Grenzen gesetzt, doch um sich durch alle diese Befehle durchzufinden, braucht man schon einiges an Geduld und meines Erachtens Routine. Wer sich aber intensiv damit befaßt, der wird ohne Zweifel zufriedenstellende Resultate erzielen.

Wie Sie jetzt Ihre Datei zum Druck aufrufen, haben Sie sicherlich schon dem Startmenü entnommen: »P« eingeben, Datei benennen und einfach einmal die erscheinenden Abfragen mit ENTER beantworten. So bekommen Sie dann endlich Ihren Text aufs Papier — immer vorausgesetzt, Ihr Drucker ist angeschlossen! Probieren Sie mal alles aus — nach einigen Anfangsschwierigkeiten kann Arbeiten mit Wordstar richtig Spaß machen.

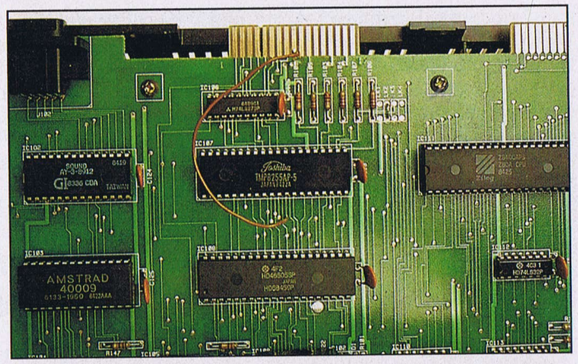
Sonderangebot auf dem Software-Markt

Neben Wordstar gibt es inzwischen noch zwei weitere professionelle CP/M-Programme für 199 Mark. Eine Datenverwaltung (dBase II) und eine Tabellenkalulation (Multiplan) machen das Angebot komplett.

(Eva Hierlmeier)

Centronics mit 8 Bit

Mit minimalem Aufwand läßt sich die Centronics-Schnittstelle der Schneider-Computer auf volle acht Bit erweitern. Mit einem kleinen Maschinenprogramm arbeitet sie mit jeder beliebigen Software zusammen.



Dank der eingebauten Centronics-Schnittstelle ist der Anschluß beliebiger Drucker an die Schneider-Computer kein Problem. Einfach ein Centronics-Kabel und einen passenden Platinenstecker besorgen, Drucker und Computer damit verbinden — fertig. Im Gegensatz zu anderen Heim-Computern braucht man weder teure Hardware-Interfaces, noch muß man irgendwelche Treibersoftware laden, um einen firmenspezifischen Zeichensatz an die vom Drucker verwendete ASCII-Norm anzupassen.

Einen kleinen Nachteil hat die Sache aber doch: Die Centronics-Schnittstelle des Schneider überträgt nur sieben Bit, das achte Bit wurde als Strobe-Bit zweckentfremdet. Das heißt, dieses Bit sagt dem Drucker, daß gültige Daten an den restlichen sieben Bit anliegen. Für reine Textausgaben ist das völlig ausreichend, schließlich handelt es sich beim ASCII-Code um einen Sieben-Bit-Code. Probleme treten aber dann auf, wenn man spezielle Eigenschaften seines Druckers — wie beispielsweise Einzelnadel-Ansteuerung für Grafikzwecke — ausnutzen will. Falls man nicht gerade über den Schneider-Drucker verfügt, der auch Sieben-Bit-Grafik beherrscht, fehlt hier einfach ein achtes Datenbit.

Eine Hardware-Erweiterung auf acht Bit ist eigentlich problemlos möglich. Etwas handwerkliches Geschick, vier Zentimeter Draht und ein kleiner Lötkolben sind alles, was man dazu braucht. Bereits in der Ausgabe 7/85 von Happy-Computer wurde die Hardware-Seite der Erweiterung beschrieben. Das Prinzip ist so einfach, daß man sich wirklich wundern muß, warum der Schneider-Computer nicht serienmäßig so angeboten wird. Und so wird's gemacht:

Lösen Sie die sechs Schrauben an der Unterseite des Gehäuses Ihres Computers und entfernen Sie den Teil mit der Tastatur. Dazu sind zwei Steckverbindungen zu lösen. Aber keine Angst, die Stecker können beim Zusammenbau nicht falsch herum aufgesetzt werden. Stellen Sie die Platine nun mit den Erweiterungsanschlüssen nach hinten vor sich auf. Am Centronics-Port zählen Sie nun von links her den neunten

Zwei Lötstellen und vier Zentimeter Draht

Platinenfinger ab. Dies ist das achte Bit für die Centronics-Leitung, das beim Schneider der Einfachheit halber auf Masse gelegt wurde. Mit einem Leiterbahnunterbrecher (ein spitzes Messer tut's auch) unterbrechen Sie diese Verbindung knapp unterhalb des Fingers. Nun wird eine neue Leitung gelegt, und zwar von Pin 12 des 8255-Ein-/Ausgabebausteins aus. Über Pin 12 werden normalerweise bei Kassettenaufzeichnungen die Daten zum Recorder übertragen. Da Recorder und Drucker aber sowieso nicht gleichzeitig betrieben werden können, bietet sich diese Leitung zur Centronics-Erweiterung geradezu an. Die von Pin 12 ausgehende Leiterbahn führt zu einer nur wenige Millimeter entfernten Lötöse, in die ein etwa vier Zentimeter langer, isolierter Draht eingelötet wird. Das andere Drahtende wird mit dem vorhin unterbrochenen neunten Platinenfinger verbunden, und zwar mit möglichst wenig Lötzinn unmittelbar oberhalb der Unterbrechungsstelle, um den Platinenfinger nicht auf ganzer Länge zu blockieren. Nun noch die Lötstellen ein letztes Mal überprüfen, den Computer wieder zuschrauben, und fertig ist die Centronics-Schnittstelle mit acht Bit.

```
1 REM  ********************************
3 REM  *        8 BIT DRUCKER         *
4 REM  *              am              *
5 REM  *      SCHNEIDER CPC 464       *
6 REM  *              von             *
7 REM  *   Bruno Kesel, Birkenweg 6   *
8 REM  *   6973 Boxberg, 07930/2948   *
10 REM ********************************
15 MEMORY &9FFF
20 FOR z=&A000 TO &A036
30 READ byte:POKE z,byte
40 NEXT
45 REM Umlegen der Druckroutine auf Adre
sse &A000
50 POKE &BDF2,&0
60 POKE &BDF3,&A0
70 DATA &01,&32,&00,&CD,&1B,&08,&30,&07
80 DATA &10,&F9,&0D,&20,&F6,&B7,&C9,&C5
90 DATA &06,&F6,&CB,&7F,&28,&06,&F5,&3E
100 DATA &20,&ED,&79,&F1,&06,&EF,&E6,&7F
110 DATA &ED,&79,&F6,&80,&F3,&ED,&79,&E6
120 DATA &7F,&FB,&ED,&79,&F5,&06,&F6,&3E
130 DATA &00,&ED,&79,&F1,&C1,&37,&C9
```

Listing 1. Der Basic-Lader für die Druckerroutine

Allerdings betrifft dies nur den Hardwareteil. Natürlich braucht man jetzt noch geeignete Software, um dieses achte Bit auch ansprechen zu können. Die Betriebssystem-Routinen des Schneider betrachten nämlich den Centronics-Port immer noch als Sieben-Bit-Schnittstelle und denken gar nicht daran, das achte Bit an den Drucker weiterzuleiten. Nach unserem kleinen Hardware-Eingriff ist das achte Bit mit Bit 5 am Port C des 8255-Bausteins verbunden. Von Basic aus kann dieses Bit also durch »OUT &F600,32« gesetzt und durch »OUT &F600,0« gelöscht werden. Im Klartext bedeutet das: Wenn ein Zeichen an den Drucker gesendet werden soll, dann geschieht das im Falle eines echten Sieben-Bit-Zeichens wie gewohnt mit »PRINT#8,...«. Falls aber ein Zeichen gesendet werden soll, bei dem das achte Bit gesetzt ist, dann muß dieses Bit zunächst auch für die Druckerausgabe aktiviert werden. Der erste Befehl lautet also »OUT &F600,32«. Anschließend sendet man das Zeichen an den Drucker: »PRINT#8,CHR$(Zeichencode)«.

Schließlich muß man natürlich noch das achte Datenbit zurücksetzen, um für weitere Druckausgaben den Originalzustand wiederherzustellen: »OUT &F600,0«.

Diese Vorgehensweise ist zugegebenermaßen etwas umständlich und langsam, da alles explizit in Basic programmiert werden muß. Das macht insbesondere bei geschützten Programmen oder bei Maschinensprache-Software einige Probleme. Mit dem hier vorgestellten Maschinenprogramm »Acht-Bit-Drukker« werden alle diese Schwierigkeiten elegant umgangen. Die Betriebssystem-Routine zur Ausgabe eines Zeichens an den Drucker wird durch den Basic-Lader (Listing 1) auf den Start des Maschinenprogramms »umgeleitet«. Nach dem Durchlauf des Basic-Laders kann man daher alle Manipulationen mit OUT-Befehlen vergessen, die erledigt das Maschinenprogramm viel schneller und unauffälliger.

Der Programmablauf erklärt sich weitgehend aus dem kommentierten Assemblerlisting (Listing 2). Es wird im Prinzip die Druckerroutine des Betriebssystems verwendet.

In den Speicherzellen BDF2 hex sowie BDF3 hex im RAM ist die Einsprungadresse für diese Routine abgelegt. Wenn man die Adresse ändert, kann man die Druckroutine ins RAM legen und sie seinen Wünschen entsprechend anpassen.

```
A000          1000          ORG  &A000
A000          1010  BUSY    EQU  &081B
A000          1020  SEND    EQU  &A00F
A000          1030  WEITER  EQU  &A01C
A000 013200   1040          LD   bc,&32
A003 CD1B08   1050  WARTE   CALL busy          ;MC BUSY PRINTER
A006 3007     1060          JR   nc,send
A008 10F9     1070          DJNZ warte
A00A 0D       1080          DEC  c
A00B 20F6     1090          JR   nz,warte
A00D B7       1100          OR   a
A00E C9       1110          RET
A00F          1120                          ;Beginn der 8 Bit Routine
A00F C5       1130  SEND    PUSH bc           ;BC Register retten
A010 06F6     1140          LD   b,&F6        ;8255 Port C
A012 CB7F     1150          BIT  7,a          ;Bit 7 im Akku gesetzt?
A014 2806     1160          JR   z,weiter     ;wenn nicht normal weiter
A016 F5       1170          PUSH af           ;Datenbyte retten
A017 3E20     1180          LD   a,&20        ;Bit 5 setzen und
A019 ED79     1190          OUT  (c),a        ;und an Port C ausgeben
A01B F1       1200          POP  af           ;Datenbyte zurueck
A01C 06EF     1210  WEITER LD   b,&EF
A01E E67F     1220          AND  &7F          ;Byte ohne Strobe
A020 ED79     1230          OUT  (c),a        ;an Drucker
A022 F680     1240          OR   &80          ;Strobe Bit setzen
A024 F3       1250          DI
A025 ED79     1260          OUT  (c),a        ,Strobe ein
A027 E67F     1270          AND  &7F          ;Strobe zuruecksetzen
A029 FB       1280          EI
A02A ED79     1290          OUT  (c),a        ;Strobe aus
A02C F5       1300          PUSH af
A02D 06F6     1310          LD   b,&F6        ;Port C
A02F 3E00     1320          LD   a,&00        ;Bit 5 an Port C
A031 ED79     1330          OUT  (c),a        ;ausschalten
A033 F1       1340          POP  af
A034 C1       1350          POP  bc
A035 37       1360          SCF
A036 C9       1370          RET

Programm :8-Bit-Drucker
Start    : &A000    Ende : &A036
Laenge   : 0037
 0  Fehler
Variablentabelle :
BUSY   081B  SEND   A00F  WEITER A01C  WARTE   A003  SEND   A00F  WEITER A01C
```

Listing 2. Das kommentierte Assemblerlisting

Dies geschieht beim Basic-Lader in den Zeilen 50 und 60. Die DATA-Zeilen enthalten das neue Maschinenprogramm für die Druckausgabe. In den Zeilen 20 bis 40 wird dieses Programm in den Speicherbereich ab A000 hex geschrieben, der vorher mit »MEMORY &9FFF« geschützt wurde.

So arbeitet das Maschinenprogramm

Im ersten Teil (Zeilen 1040 bis 1110) wartet das Programm, bis der Drucker sich druckbereit meldet. Ist er zum Empfang eines Zeichens bereit, wird zur eigentlichen Ausgabe verzweigt. Das Register B bekommt die Adresse des Port C des 8255 zugewiesen. Dann wird geprüft, ob der Zeichencode im Akku größer als 127 ist. Ist dies der Fall, so wird das Bit 5 des Port C gesetzt. Erst jetzt wird das Zeichen an die Druckerschnittstelle ausgegeben.

Nach der Ausgabe muß nur Bit 5 an Port C, welches ja das Datenbit Nummer acht darstellt, wieder gelöscht werden. Damit ist die Übertragung des Zeichens abgeschlossen.

Das Programm arbeitet mit jedem Drucker ohne Änderung, wobei nun der gesamte Zeichensatz des Druckers und eventuelle grafische Fähigkeiten nutzbar sind.

Der Basic-Coder wird nur zur einmaligen Initialisierung gebraucht. Nachdem das Programm geladen und ausgeführt wurde, kann es mit »NEW« wieder gelöscht werden. Anschließend kann man ein beliebiges Anwender-Programm laden. Einzige Voraussetzung: Der Speicherbereich des &A000 darf nicht benutzt werden. Das folgende kurze Testprogramm zeigt alle Druckercodes:

```
10  FOR N=&20 TO &FF
20  PRINT#8,CHR$(N);" ";
30  NEXT N
```

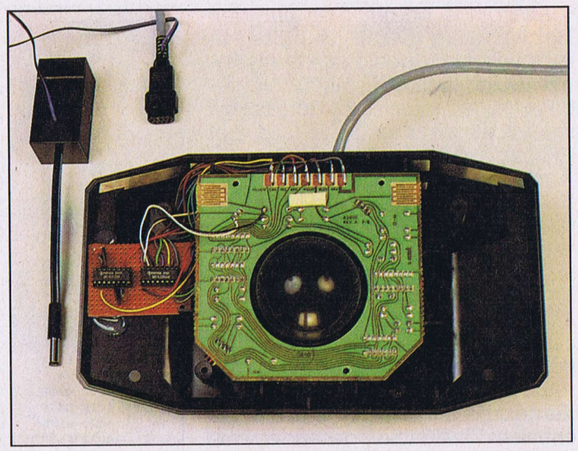
Dieses kurze Programm demonstriert aufschlußreich die Wirkung der relativ einfachen Änderung der Hardware in Verbindung mit diesem Maschinenprogramm.

Man kann sich so ohne wesentliche Kosten und Zeitaufwand eine vollwertige Druckerschnittstelle schaffen.

(Bruno Kesel/Volker Everts)

Ein Trakball für den Schneider

Eine Maus, ein Grafiktablett oder einen Trakball am Schneider anzuschließen, das ist nicht so ohne weiteres möglich. Am Joystick-Port fehlt nämlich die Stromversorgung. Dabei ist sie aber ganz einfach selbst zu basteln.



Die geänderte Platine ►

Leider gibt es für den Schneider-Computer noch relativ wenig Hardwarezusätze. Was liegt also näher, als bereits vorhandene Peripheriegeräte anderer Computer an den CPC anzupassen. Im übrigen hat dies den Vorteil, daß solche Zusatzgeräte, wenn schon länger auf dem Markt, auch noch billiger sind als Neuentwicklungen. Um einen dieser preiswerten Zusätze handelt es sich nun in diesem Artikel, nämlich den Atari-Trakball.

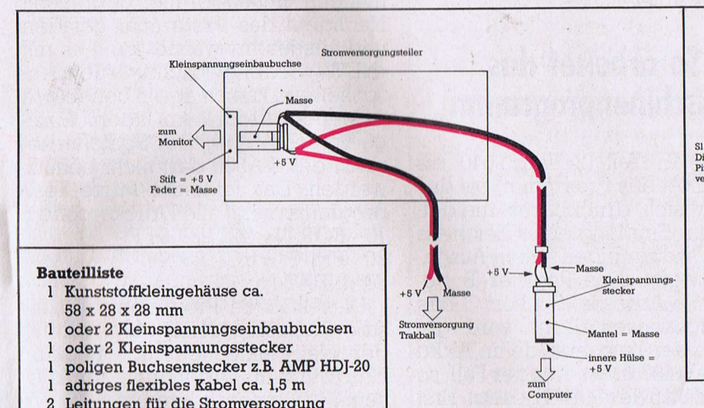
Da dieser Trakball schon längere Zeit sein Dasein fristet, ist er vom Preis her besonders günstig. Mit knapp unter 100 Mark bietet er ein hervorragendes Preis-Leistungsverhältnis. Um ihn jedoch für seinen Schneider zu benutzen, ist noch einiges an Bastelei erforderlich — ansonsten wäre es ja auch zu einfach. Wie man diese umgedrehte Maus nun am CPC 464 benutzen kann, das lesen Sie hier.

Wer sich schon einmal mit den Joystick-Ports der verschiedenen Computer befaßt hat, der weiß, daß der Atari außer den üblichen Kontakten zur Ein- und Ausgabe dem Benutzer auch noch +5 Volt und Masse zur Verfügung stellt.

Beim Schneider ist dies nicht der Fall. Er besitzt weder die Stromversorgung noch einen festen Massepunkt am Joystick-Port. Diesem Mißzustand kann man aber leicht Abhilfe schaffen. Das Kabel zur Stromversorgung des Computers, das vom Monitor kommt, liefert Masse und die benötigten 5 Volt. Man braucht dann lediglich einen Abzweig, am besten in einem Kunststoffgehäuse, welches man sich, genauso wie alle anderen Bauteile, in einem Elektronikladen besorgen kann. In Bild 1 sehen wir den Verdrahtungsplan des Stromversorgungsteilers. Nun braucht man für dieses Gehäuse noch eine Kleinspannungseinbaubuchse, sowie einen Kleinspannungsstecker und zwei zweiadrige Leitungen. Die Leitungen wählt man am besten in zwei verschiedenen Farben, um eventuelle Verwechslungen von Plus und Minus von vornherein auszuschließen. Die zum Computer führende Leitung muß nur etwa 10 cm, die andere, die den Trakball anschließt, etwa 35 cm lang sein. Wie Einbau und Verdrahtung genau vor sich gehen, zeigt die erste Abbildung.

Wenn man sich den Trakball betrachtet, fallen zwei Tasten auf, die rechts und links am Gehäuse sitzen. Nun sind bei der Belegung der Joystick-Ports zwei Eingänge für Funktionstasten vorgesehen. Um diese Funktionen auch an unserem Trakball voll auszuschöpfen, müssen wir allerdings eine kleine Änderung vornehmen.

Am Trakball sind die beiden Tasten nämlich parallel geschaltet, das heißt egal ob man rechts oder links drückt, es wird immer als der gleiche Tastendruck gewertet. Dieses Manko muß man aus der Welt schaffen. Für die Modifizierung benötigen wir einen neunpoligen Buchsenstecker, wie zum Beispiel den Typ AMP HDJ-20, sowie ein neunadriges flexibles Kabel. Die Länge des Kabels bleibt dem Benutzer selbst überlassen, je länger das Kabel, um so beweglicher ist man mit dem Trakball. Allerdings sollte man es nicht übertreiben; irgendwann ist das Kabel zu lang, um eine sichere Funktion der Schaltung zu gewährleisten. Wie die genaue Verbindung des Steckers und des Kabels auszusehen hat, werden wir später noch genauer erläutern. Bild 2 zeigt den Anschlußpunkt der zweiten Taste auf der Platine des Trakballs.



Bauteilliste

- 1 Kunststoffkleingehäuse ca. 58 x 28 x 28 mm
- 1 oder 2 Kleinspannungseinbaubuchsen
- 1 oder 2 Kleinspannungsstecker
- 1 poligen Buchsenstecker z.B. AMP HDJ-20
- 1 adriges flexibles Kabel ca. 1,5 m
- 2 Leitungen für die Stromversorgung
- 1 Lochrasterplatte
- 2 16 pol. IC-Fassungen
- 1 74 LS 123
- 1 74 LS 365
- 2 Widerstände 82 k Ohm
- 2 Kondensatoren 100 nF

Bild 1. Die Stromversorgung des Trakballs erfordert eine eigene 5-Volt-Leitung

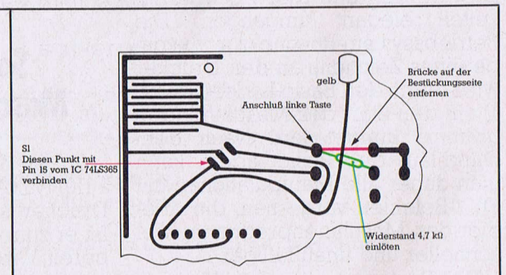


Bild 2. In der linken Ecke der Trakballplatine muß umgebaut werden.

Wenden wir uns dem nächsten und letzten Problem zu, der Anpassung der Signale vom Trakball zum Computer. Kurz zur Funktion eines Trakball: Um dem Computer verständlich zu machen, daß sich die Kugel in Bewegung befindet, sendet die Elektronik des Trakball jeweils für die horizontale und vertikale Bewegung eine Anzahl von Impulsen aus. Die Impulsfrequenz nimmt zu, je schneller die Kugel bewegt wird. Außerdem bekommt der Computer noch die Information, in welche Richtung sich die Kugel gerade bewegt. Wiederum je eine für die vertikale und eine für die horizontale Richtung.

Da der Trakball nur dann Impulse liefert, wenn sich die Kugel in Bewegung befindet, kann man davon ausgehen, daß immer, wenn am entsprechenden Eingang des Joystick-Ports ein High-Signal (logischer Wert 1) anzufinden ist, die Kugel sich bewegt (Impulse sind ja nichts anderes als ein ständiger Wechsel von 0 nach 1). Allerdings liefert die Schaltung bei einer ungünstigen Ruhestellung des Trakballs einen ständigen High-Zustand an den Computer. Damit kann es natürlich zu einer falschen Auswertung des Signals kommen. Aus diesem Grund wird diesem Signal einfach ein sogenannter monostabiler Multivibrator nachgeschaltet. Dieser hat die Aufgabe, ankommende Impulse eine gewisse Zeit auf High-Pegel zu halten, und nach dieser Zeit wieder nach Low (logischer Wert 0) zu legen. Dieser Vorgang erfolgt unabhängig davon, ob der Eingang des monostabilen Multivibrators auf High- oder Low-Pegel liegt. Da der Computer nun auf jeden Fall ein Low-Signal bekommt, wenn die Kugel ihre Ruhestellung eingenommen hat, ist eine falsche Auswertung des Signals ausgeschlossen. Wie ein monostabiler Multivibrator aufgebaut ist, zeigt uns Bild 3. Realisiert wird die Schaltung mit einem 74 LS 123. Dieser enthält zwei von einander unabhängige Multivibratoren, von denen je einer für die Geschwindigkeit in horizontaler und vertikaler Bewegung zuständig ist. Die Richtung in welche sich die Kugel bewegt, wird durch zwei andere Signale bestimmt. Wieder ist jeweils eine Leitung für die horizontale und eine für die vertikale Richtung zuständig. Eine logische 1 gibt zum Beispiel an, ob sich die Kugel nach oben oder nach links bewegt. Nun können wir festlegen, welche Bits am Joystick-Port welche Aufgabe übernehmen.

Der nachfolgenden Tabelle kann man genau entnehmen wie die einzelnen Bis belegt sind:

Bit 0 — Wert 1 Bewegung vertikal
Bit 0 — Wert 0 keine Bewegung vertikal
Bit 1 — Wert 1 Bewegung horizontal
Bit 1 — Wert 0 keine Bewegung horizontal
Bit 2 — Wert 1 Richtung nach unten
Bit 2 — Wert 0 Richtung nach oben
Bit 3 — Wert 1 Richtung nach rechts
Bit 3 — Wert 0 Richtung nach links
Bit 4 — Wert 0 Taste rechts gedrückt
Bit 5 — Wert 0 Taste links gedrückt

Diese Werte behalten nur dann ihre Gültigkeit, wenn man sie direkt per Maschinensprachenprogramm ermittelt. Liest man den Joystick-Port von Basic aus, werden die Daten invertiert (alles was 1 ist wird 0 und umgekehrt) dargestellt. Dies darf man nicht vergessen, da sonst falsche Ergebnisse die Folge sind. Laut der Tabelle können diejenigen Bits, welche die Richtung angeben, auch in der Ruhephase des Trakballs einen festen 0-Wert an den Computer liefern.

Da der Joystick-Port beim Schneider leider parallel zur Tastatur gelegt wurde, um Bauteile einzusparen, passiert es bei angeschlossenem Trakball hin und wieder, daß eine korrekte Eingabe nicht mehr möglich ist. Der Computer ist dann fälschlicherweise der Meinung, daß eine Taste gedrückt wurde. Dieses Phänomen hat sicher jeden schon einmal irritiert, der einen Joystick am Computer angeschlossen hatte und bei Eingabe über Tastatur versehentlich an den Joystick gestoßen ist. Um jetzt nicht jedesmal den Trakball vom Computer trennen zu müssen, kann man ihn ganz einfach abschalten. Wie dies realisiert wird, ist aus Bild 3 zu erkennen. Der Baustein 74 LS 365 eignet sich hervorragend für diese Aufgabe. Seine Funktionsweise ist folgende:

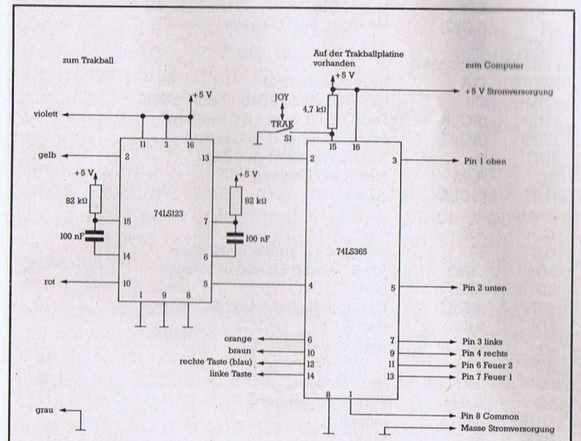


Bild 3. Der Schaltplan für den monostabilen Multivibrator

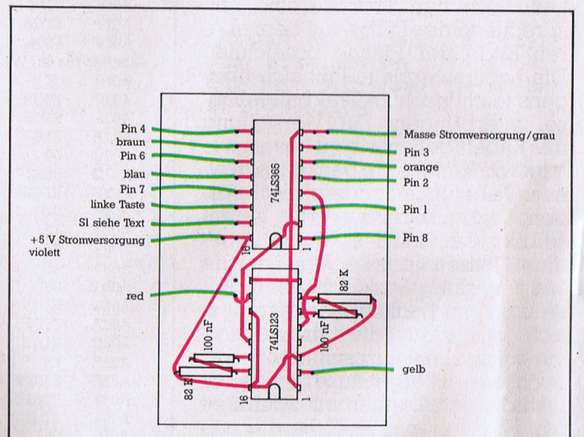


Bild 4. So kann die Zusatzplatine aussehen

Im Inneren des ICs befinden sich sechs Schalter. Diese Schalter sind nur dann geschlossen, wenn man an den Pins 15 und 1 einen Pegel von 0 Volt (logischer Wert 0) anlegt. Bei geöffnetem Zustand kann keine Information weitergeleitet werden.

Wenn man sich Bild 3 betrachtet, kann man erkennen, daß der Eingang 1 des 74 LS 365 mit Pin 8 des Joystick-Ports verbunden wird. Somit wird festgelegt, daß nur bei Abfrage des Joysticks 0 die Informationen des Trakballs weitergegeben werden. Richten wir unser Augenmerk nun auf Pin 15 des 74 LS 365. Dieser liegt über einem Widerstand ständig auf High-Potential und kann nur mit dem Schalter S1 auf Low-Potential gezogen werden. High sagt, daß die Informationen nicht an den Computer weitergeleitet werden sollen. Befindet sich der Schalter in Stellung Trak (am Gehäuse vermerkt), so wird Pin 15 auf Low-Potential geschaltet und die Information gelangt zum Computer. Dieser Schalter, und der Widerstand, sind bereits auf der Trakballplatine vorhanden. Der Schalter befindet sich auf der Rückseite des Gerätes, der Widerstand ist direkt auf der Platine zu finden. Wie man den Pin 15 mit der Platine verbinden muß, wird aus Bild 2 ersichtlich.

Da nur wenige Bauteile benötigt werden, kann man die ganze Schaltung leicht auf einer Lochrasterplatte mit der Größe 54 x 38 mm aufbauen. Einen Vorschlag, wie man den Aufbau vornehmen kann, zeigt Bild 4.

Die zum Computer führenden Leitungen sind entsprechend der Pins des Joystick-Ports bezeichnet, die Leitungen zum Trakball sind entsprechend der Farbe, auf dem oberen Ende der Platine bezeichnet. Die Lochrasterplatte läßt sich übrigens leicht noch im Trakballgehäuse unterbringen. Zur Herstellung des Kabels gehen wir am besten wie folgt vor: Welche Farben den einzelnen Funktionen zugeordnet werden, das überlassen wir jedem selbst. Man sollte sich nur unbedingt Notizen dazu machen, um eine Verwechslung auszuschließen. Ein neunpoliger Buchsenstecker eignet sich gut dazu, die Stromversorgungskabel mit anzuschließen. Das Buchsensteckergehäuse versieht man dann von hinten mit zwei kleinen Bohrungen, durch die die beiden Stromversorgungskabel gezogen und mit zwei Adern des neunpoligen Kabels verlötet werden. Nun bleibt noch, die übrigen Kabel am Buchsenstecker entsprechend der

```
A000  0109A0   LD    BC,A009   Adresse der Befehlstabelle
A003  2113A0   LD    HL,A013   Kernal
A006  C3D1BC   JP    BCD1      einbinden. Bitte
A009  0EA0     LD    C,A0      Zeiger auf das Wort 'Ball'
A00B  C317A0   JP    A017      Anfang Befehl 'Ball'
A00E  42       LD    B,D
A00F  41       LD    B,C
A010  4C       LD    C,H
A011  CC0013   CALL  Z,1300
A014  A0       AND   B
A015  09       ADD   HL,BC
A016  A0       AND   B
A017  F3       DI
A018  CD59A0   CALL  A059      Pfeil an aktuelle Grafikcursorpos.
A01B  CD6AA0   CALL  A06A      Joyport 0 lesen
A01E  3AFEB4   LD    A,(B4FE)  Wert sichern
A021  CB6F     BIT   5,A       Ist Taste links gedrückt
A023  2804     JR    Z,A029    ja dann Ende ausführen
A025  E603     AND   03        die Bewegungsbits ausklammern
A027  28F2     JR    Z,A01B    wenn die Kugel steht, dann weiter warten
A029  CD59A0   CALL  A059      Pfeil wieder löschen
A02C  3AFEB4   LD    A,(B4FE)  Joystickwert noch mal holen
A02F  CB6F     BIT   5,A       wenn Taste links gedrückt ist
A031  2824     JR    Z,A057    dann zurück zum Basicprogramm
A033  210000   LD    HL,0000   HL = Y-Distanz gleich 0
A036  110000   LD    DE,0000   DE = X-Distanz gleich 0
A039  CB47     BIT   0,A       bewegt sich die Kugel rauf/runter
A03B  2807     JR    Z,A044    nein, dann links/rechts abfragen
A03D  CB57     BIT   2,A       ja Richtung ermitteln
A03F  2B       DEC   HL        -1 Richtung rauf
A040  2002     JR    NZ,A044   ist das Bit 1 dann Richtung = rauf
A042  23       INC   HL        wenn Bit = 0
A043  23       INC   HL        dann +1 Richtung runter
A044  CB4F     BIT   1,A       bewegt sich die Kugel links/rechts
A046  2807     JR    Z,A04F    nein, dann Pfeil an neue Pos. plotten
A048  CB5F     BIT   3,A       ja Richtung ermitteln
A04A  1B       DEC   DE        -1 Richtung links
A04B  2802     JR    Z,A04F    ist das Bit 0 dann Richtung links
A04D  13       INC   DE        wenn Bit = 1
A04E  13       INC   DE        dann +1 Richtung rechts
A04F  CDC3BB   CALL  BBC3      Move Grafikcursor relativ
A052  CD59A0   CALL  A059      Pfeil an Grafikcursorpos. plotten
A055  18C4     JR    A01B      und nächste Abfrage starten
A057  FB       EI              Ende des Programms
A058  C9       RET
A059  ED5B2CB3 LD    DE,(B32C) X-Pos. aus dem Zwischenspeicher holen
A05D  D5       PUSH  DE        X-Pos. retten
A05E  3EFF     LD    A,FF      Zeichen 255 ist Pfeil
A060  CDFCBB   CALL  BBFC      Zeichen auf Bildschirm bringen
A063  F3       DI              Interrupt unterdrücken
A064  D1       POP   DE        alte X-Pos. wiederholen
A065  ED532CB3 LD    (B32C),DE und in den Zwischenspeicher zurücklegen
A069  C9       RET
Unterprogramm Joystickabfrage
A06A  010EF4   LD    BC,F40E   Port A 8255
A06D  ED49     OUT   (BC),C    Reg. 14 8912 (I/O Port A)
A06F  06F6     LD    B,F6      Port C 8255
A071  ED78     IN    A,(BC)    Wert von Port C holen
A073  E630     AND   30
Alle Bits bis auf WR-Data und Motor löschen
A075  4F       LD    C,A       Ergebnis nach C
A076  F6C0     OR    C0        Bit 6+7 setzen BDIR und BC1 8912
A078  ED79     OUT   (BC),A    Port C Bit 0..3 und 6,7 löschen
A07A  ED49     OUT   (BC),C    Port C BDIR und BCi setzen
A07C  04       INC   B         Control-Register des 8255
A07D  3E92     LD    A,92      Port A als Eingang
A07F  ED79     OUT   (BC),A    setzen
A081  C5       PUSH  BC
A082  79       LD    A,C
A083  F649     OR    49        BC1 und Common maskieren
A085  4F       LD    C,A       Joy 0 Leitung Common setzen
A086  06F6     LD    B,F6      Port C
A088  ED49     OUT   (BC),C    Common Leitung und BC1 setzen
A08A  06F4     LD    B,F4      Port A
A08C  ED78     IN    A,(BC)    Joyport 0 lesen
A08E  C1       POP   BC
A08F  32FEB4   LD    (B4FE),A  Wert sichern
A092  3E82     LD    A,82      Port A als Ausgang
A094  ED79     OUT   (BC),A    setzen
A096  05       DEC   B         Port C
A097  ED49     OUT   (BC),C    auf Urzustand setzen
A099  C9       RET
```

Listing 2. Das Assembler-Listing für »I BALL«


```
10 '************************************
20 '
30 ' Trakballbefehlserweiterung
40 '
50 ' RSX-Einbindung
60 '
70 '************************************
80 '
90 '
100 MEMORY &9FFF:CLS
110 FOR p=&A000 TO &A099
120 READ d$:POKE p,VAL("&"+d$)
130 NEXT
140 CALL &A000
150 PRINT"Der Befehl '|BALL' steht zur V
erfuegung"
200 DATA 1,9,a0,21,13,a0,c3,d1,bc,e,a0,c
3,17,a0,42,41
210 DATA 4c,cc,0,0,0,0,0
220 DATA f3,cd,59,a0,cd,6a
230 DATA a0,3a,fe,b4,cb,6f,28,4,e6,3,28,
f2,cd,59,a0,3a,fe,b4,cb,6f,28,24
240 DATA 21,0,0,11,0,0,cb,47,28,7,cb,57,
2b,20,2,23,23,cb
250 DATA 4f,28,7,cb,5f,1b,28,2,13,13,cd,
c3,bb,cd,59,a0
260 DATA 18,c4,fb,c9,ed,5b,2c,b3,d5,3e,f
f,cd,fc,bb,f3,d1,ed,53,2c,b3,c9
270 DATA 1,e,f4,ed,49,6,f6,ed,78,e6,30,4
f,f6,c0,ed,79,ed
280 DATA 49,4,3e,92,ed,79,c5,79,f6,49,4f
,6,f6,ed,49,6
290 DATA f4,ed,78,c1,32,fe,b4,3e,82,ed,7
9,5,ed,49,c9
```

Listing 1. Die RSX-Erweiterung »I BALL« steuert die Trakballabfrage

Belegung anzuschließen. Dabei wird man feststellen, daß ein Kabel übrig bleibt. Dieses ist das Kabel für den Pin Common 2, den wir aber nicht benötigen. Somit muß man ihn nicht mit anschließen. Nun sollte man noch die beiden im Gehäuse verlöteten Kabel mit Isolierband umwickeln, um einen eventuellen Kurzschluß der Stromversorgungsleitungen zu verhindern. Hat man das Kabel hergestellt, entfernt man das alte vom Trakball und führt das neue in das Gehäuse, wo man es mit der Lochrasterplatte nach der Verbindungsliste anschließt. Zu guter Letzt werden noch die Leitungen, die von der Rasterplatte zur Trakballplatine führen, miteinander verbunden. Die Lochrasterplatte kann man in das Trakballgehäuse links von der Platine einkleben oder festschrauben. Jetzt sind wir mit den Bastelarbeiten soweit fertig, daß wir das Trakballgehäuse wieder verschließen und uns an die Software wagen können. Natürlich ist nicht unbedingt ein spezielles Programm nötig, um den Trakball zu benutzen. Am Beispielprogramm aber wird es offensichtlich, wie man den Trakball sinnvoll in der Praxis anwenden kann. Das erste Programm (Listing 1) zeigt, wie mit Hilfe der RSX-Befehlserweiterung ein neuer Befehl geschaffen

```
10 '************************************
20 '
30 ' DEMO  BEFEHL-BALL
40 '
50 '************************************
60 '
70 '
80 '
90 '
100 MODE 1:INK 0,0:INK 1,26:INK 2,15:INK
 3,14:BORDER 0:PRINT CHR$(23)"0"::PEN 1
110 SYMBOL 255,&FC,&88,&84,&82,&C1,&A2,&
14,8:REM Pfeil
120 x=0:y=386:GOSUB 310:x=160:GOSUB 310:
REM Kaestchen fuer x+y pos
130 LOCATE 3,2:PRINT"X-POS"::LOCATE 13,2
:PRINT"Y-POS":
140 y=280:FOR x=0 TO 600 STEP 40
150 GOSUB 320:NEXT:REM demo-kaestchen
160 TAG:FOR x=0 TO 15:MOVE 10+x*40,268:P
RINT CHR$(65+x)::NEXT:TAGOFF:REM kaestch
en markieren
170 PLOT 0,200,1:DRAWR 638,0:DRAWR 0,-10
0:DRAWR -638,0:DRAWR 0,100:REM kasten fu
er meldung
180 IF (JOY(0) AND 16)=0 THEN 180
190 PLOT 0,400,1:MOVE 320,220:REM farbe
und ausgansstellung des pfeils
200 PRINT CHR$(23)"1"::|BALL:PRINT CHR$(
23)"0"::REM Graphikfarbstiftmodus setzen
 und Trakballroutine aufrufen
210 x=XPOS:y=YPOS:REM X und Y-Position f
eststellen
220 IF y<280 AND y>240 AND x>0 AND x<640
 THEN c=0:GOTO 240:REM wenn der Pfeil im
 Bereich der Kaestchen ist dann Kaestche
nbuchstabe ermiteln
230 LOCATE 5,16:PRINT"DER PFEIL BEFINDET
 SICH AUSSERHALB"::LOCATE 10,17:PRINT"
   DES BEREICHS"::GOTO 280
240 x=x-40:c=c+1:IF x>0 THEN 240
250 K$=CHR$(64+C)
260 LOCATE 5,16:PRINT"ES WURDE DAS KAEST
CHEN   "::PEN 3:PRINT k$;"          ":
270 LOCATE 10,17:PEN 1:PRINT"ANGEWAEHLT
.      ";
280 LOCATE 3,4:PRINT USING"#####":XPOS;:
LOCATE 13,4:PRINT USING"#####":YPOS;
290 GOTO 180
300 END
310 PLOT x,y,2:DRAWR 158,0:DRAWR 0,-54:D
RAWR -158,0:DRAWR 0,54:RETURN
320 PLOT x,y,3:DRAWR 38,0:DRAWR 0,-40:DR
AWR -38,0:DRAWR 0,40:RETURN
```

Listing 3. Eine Demonstration unserer Hardware-Erweiterung

wird. Dieser lautet in unserem Beispiel »Ball«. Für all diejenigen, die ganz genau wissen wollen wie dieser Befehl funktioniert, haben wir die Mnemonics mit entsprechender Dokumentation abgedruckt (Listing 2). Der Maschinencode steht hierbei ab Adresse A000hex und kann natürlich nach Belieben und entsprechender Anpassung an eine andere Adresse gelegt werden.

Das zweite Programm (Listing 3) demonstriert, wie man den Befehl »Ball« benutzen kann. Am Bildschirm werden verschieden große Kästchen gezeichnet, deren genauere Bedeutung Sie bitte aus dem Listing selbst entnehmen. Betätigt man die rechte Taste, wird der neue Befehl aufgerufen und es erscheint ein Pfeil auf dem Bildschirm. Mit Hilfe der Kugel kann man nun diesen Pfeil auf dem Monitor hin und her bewegen. Drückt man die linke Taste, so verschwindet der Pfeil wieder und man kehrt ins Basic-Programm zurück. Weist der Pfeil innerhalb des Kästchens auf einen der Buchstaben, so erscheint im darunterstehenden Kasten eine Anzeige, um welchen Buchstaben es sich handelt. Dies ist eine Möglichkeit, menügesteuerte Bilder zu realisieren.

(Michael Bauer)

Markt&Technik-Buchverlag

Depot-Händler

Tragen Sie Ihre Buchbestellung und die Anschrift des Depotbuchhändlers auf die Bestellkarte in diesem Heft ein. Bitte vergessen Sie den Absender nicht.

Buchhandlung Herder, Kurfürstendamm 69
1000 Berlin 15, Tel. (030) 8835002,
BTX *921782 #
Computare Fachbuchhandlung, Keithstraße 18
1000 Berlin 30, Tel. (030) 2139021
Thalia Buchhaus, Große Bleichen 19
2000 Hamburg 36, Tel. (040) 3005050
Boysen + Maasch, Hermannstraße 31
2000 Hamburg 1, Tel. (040) 30505015
Electro-Data, Wilhelm-Heidsiek-Straße 1
2190 Cuxhaven, Tel. (04721) 51288
Buchhandlung Muehlau, Holtenauer Straße 116
2300 Kiel, Tel. (0431) 85085
ECL, Norderstraße 94-96
2390 Flensburg, Tel. (0461) 28181
Buchhandlung Weiland, Königstraße 79
2400 Lübeck, Tel. (0451) 74006-09
Buchhandlung Storm, Langenstraße 10
2800 Bremen 1, Tel. (0421) 321523
Buchhandlung Lohse-Eissing, Marktstraße 38
2940 Wilhelmshaven, Tel. (04421) 41687
Buchhandlung Schmorl u. v. Seefeld, Bahnhofstraße 13
3000 Hannover 1, Tel. (0511) 327651
Buchhandlung Graff, Neue Straße 23
3300 Braunschweig, Tel. (0531) 49271
Deuerlich'sche Buchhandlung, Weender Straße 33
3400 Göttingen, Tel. (0551) 56868
Buchhandlung an der Hochschule, Holländische Straße 22
3500 Kassel, Tel. (0561) 83807
Stern Verlag, Friedrichstraße 24-26
4000 Düsseldorf, Tel. (0211) 373033
Buchhandlung Baedeker, Kettwiger Straße 33-35
4300 Essen 1, Tel. (0201) 221381
Regensberg'sche Buchhandlung, Alter Steinweg 1
4400 Münster, Tel. (0251) 4054-5
Buchhandlung Acker, Johannisstraße 51
4500 Osnabrück, Tel. (0541) 28488
Buchhandlung Lensing, Westenhellweg 86-88
4600 Dortmund, Tel. (0231) 16980
Buchhandlung Brockmeyer, Querenburger Höhe 281/Unicenter
4630 Bochum, Tel. (0234) 701360
Buchhandlung Meier + Weber, Warburger Straße 98
4790 Paderborn, Tel. (05251) 63172
Buchhandlung Phönix GmbH, Oberntorwall 25
4800 Bielefeld 1, Tel. (0521) 69071
Buchhandlung Gonski, Neumarkt 24
5000 Köln 1, Tel. (0221) 210528
Mayer'sche Buchhandlung, Ursulinerstraße 17-19
5100 Aachen, Tel. (0241) 48142
Buchhandlung Behrendt, Am Hof 5a
5300 Bonn 1, Tel. (0228) 658021
Buchhandlung Cusanus, Schloßstraße 12
5400 Koblenz, Tel. (0261) 36239
Akad. Buchhandlung Interbook, Fleischstraße 61-65
5500 Trier, Tel. (0651) 43596
Buchhandlung W. Finke, Kipdorf 32
5600 Wuppertal 1, Tel. (0202) 454220
Buchhandlung Balogh, Sandstraße 1
5900 Siegen, Tel. (0271) 55298-9
Buchhandlung Naacher, Steinweg 3
6000 Frankfurt 1, Tel. (069) 298050
Buchhandlung Wellnitz, Lautenschlägerstraße 4
6100 Darmstadt, Tel. (06151) 76548
Buchhandlung Feller + Gecks, Friedrichstraße 31
6200 Wiesbaden, Tel. (06121) 304911
Ferber'sche UNI-Buchhandlung, Seltersweg 83
6300 Gießen, Tel. (0641) 12001
Sozialwissenschaftliche Fachbuchhandlung, Friedrichstraße 24
6400 Fulda, Tel. (0661) 75077
Gutenberg Buchhandlung, Große Bleiche 29
6500 Mainz, Tel. (06131) 37011
Buchhandlung Beck + Seip, Futterstraße 2
6600 Saarbrücken, Tel. (0681) 30677
Buchhandlung Wilhelm Hofmann, Bismarckstraße 98
6700 Ludwigshafen, Tel. (0621) 516001
Buchhandlung Loeffler, B 1,5
6800 Mannheim 1, Tel. (0621) 28912
Buchhandlung Stehn, Bahnhofstraße 13
7000 Stuttgart 50, Tel. (0711) 561476
Buchhandlung am Markt, Kramstraße 6
7100 Heilbronn, Tel. (07131) 68682
PCB Micro-Computer, Oskar-Kalbfell-Platz 8
7410 Reutlingen, Tel. (07121) 270443
UNI Buchhandlung Kellner + Moessner, Kaiserstraße 18
7500 Karlsruhe, Tel. (0721) 691436
Buchhandlung Roth, Hauptstraße 45
7600 Offenburg, Tel. (0781) 22097
Rombach Center, Bertholdstraße 10
7800 Freiburg, Tel. (0761) 49091
Fachbuchhandlung Hofmann, Hirschstraße 4
7900 Ulm, Tel. (0731) 60949
Schautes Elektronik, Bachstraße 52
7980 Ravensburg, Tel. (0751) 26138
Buchhandlung Hugendubel, Marienplatz
8000 München 2, Tel. (089) 2389-1
Computerbücher am Obelisk, Barerstraße 32-34
8000 München 2, Tel. (089) 282383
Pele's Computerbücher, Schillerstraße 17
8000 München 2, Tel. (089) 555229
Universitätsbuchhandlung Lachner, Theresienstraße 43
8000 München 2, Tel. (089) 521340
Buchhandlung Schönhuber, Theresienstraße 6
8070 Ingolstadt, Tel. (0841) 33146/47
Computerstudio Gertrud Friedrich, Ludwigstraße 3
8220 Traunstein, Tel. (0861) 14767
Buchhandlung Pustet, Kl. Exerzierplatz 4
8390 Passau, Tel. (0851) 56945
Buchhandlung Pustet, Gesandtenstraße 6
8400 Regensburg, Tel. (0941) 53061
Buchhandlung Dr. Büttner, Adlerstraße 10-12
8500 Nürnberg, Tel. (0911) 232318
STS Computer Vertrieb, Werner-Siemens-Straße 19
8580 Bayreuth, Tel. (0921) 62320
Computer-Center-Burger, Leimitzer Straße 11-13
8670 Hof, Tel. (09281) 40075
Sortiments- u. Bahnhofsbuchh. J. Strykowski, Bahnhofplatz 4
8700 Würzburg, Tel. (0931) 54389
Buchhandlung Pustet, Grottenau 4
8900 Augsburg, Tel. (0821) 35437
Kemptener Fachsortiment, Salzstraße 30
8960 Kempten, Tel. (0831) 14413

Belgien:
Eicher Micro & Personal Computer, Hünningen 56-58
B-4780 St. Vith, Tel. (080) 227393

Luxemburg:
Librairie Promoculture, 14, rue Duchscher (Pl. de Paris)
L-1011 Luxembourg-Gare, Tel. 480691, Telex 3112

Schweiz:
Buchhandlung Meissner, Bahnhofstraße 41
5000 Aarau, Tel. (064) 247151
Bücher Balmer, Neugasse 12
6300 Zug, Tel. (042) 214141
Buchhandlung Enge, Bleicherweg 56
8002 Zürich, Tel. (01) 2012078
Buchhandlung Orell Füssli, Pelikanstraße 10
8022 Zürich, Tel. (01) 2118011
Freihofer AG, Wissenschaftliche Buchhandlung, Universitätsstr. 11
8033 Zürich, Tel. (01) 3634282
Buchhandlung am Rößlitor, Webergasse 5
9001 St.Gallen, Tel. (071) 228726

Inserentenverzeichnis

Abacomp	25
Computershop	25
Data Becker	11, 21
Data Media	2, 15, 144
ERC-Soft	57
Forth Systeme	57
Fun Tastic	55
G + R Siemens	57
Janke	25
Joysoft	25
Markt & Technik Buchverlag	32/33, 61, 81, 129, 138, 143
Mükra	55
Vortex	39
Wagner	57

Das nächste Sonderheft Happy-Spiele erscheint am 5.11.1985

Anzeigenschluß ist der 11.10.1985

Impressum

Herausgeber: Carl-Franz von Quadt, Otmar Weber

Chefredakteur: Michael Scharfenberger (sc)
Redakteure: Andreas Hagedorn (hg; Inhalt)
Petra Wängler (wg; Koordination)
Mitarbeiter dieses Sonderheftes: Michael Bauer, Matthias Engelbach, Volker Everts, Oliver Felchner, Oliver Heggelbacher, Eckhard Hinrichsen, Eva Hirlmeier, Martin Huber, Stefan Langkau, Bruno Kesel, Manfred Mayer, Alfred Otto, Olav Panckow, Henning Roes, Michael Straßer, Bernhard Straub, Carsten Straush, Helmut Tischer, Günter Thurn, J. Völker, Wolfgang Volz, Erhart Wais, Jens Wulf
Redaktionsassistenz: Monika Lewandowski (222)
Fotografie: Jens Jancke

Layout: Leo Eder (Ltg.)

Auslandsrepräsentation:
Schweiz: Markt&Technik Vertriebs AG, Alpenstrasse 14, CH-6300 Zug,
Tel. 042-223155/56, Telex: 862329 mut ch
USA: M&T Publishing, 2464 Embarcadero Way, Palo Alto, CA 94303; Tel. 415-424-0600; Telex 752351

Manuskripteinsendungen: Manuskripte und Programmlistings werden gerne von der Redaktion angenommen. Sie müssen frei sein von Rechten Dritter. Sollten sie auch an anderer Stelle zur Veröffentlichung oder gewerblichen Nutzung angeboten worden sein, muß dies angegeben werden. Mit der Einsendung von Manuskripten und Listings gibt der Verfasser die Zustimmung zum Abdruck in von der Markt&Technik Verlags AG herausgegebenen Publikationen und zur Vervielfältigung der Programmlistings auf Datenträger. Mit der Einsendung von Bauanleitungen gibt der Einsender die Zustimmung zum Abdruck in von Markt&Technik Verlag AG verlegten Publikationen und dazu, daß Markt&Technik Verlag Geräte und Bauteile nach der Bauanleitung herstellen läßt und vertreibt oder durch Dritte vertreiben läßt. Honorare nach Vereinbarung. Für unverlangt eingesandte Manuskripte und Listings wird keine Haftung übernommen.

Herstellung: Klaus Buck (180)

Anzeigenverkauf: Brigitta Fiebig (211)

Anzeigenverwaltung und Disposition: Patricia Schiede (172)

Vertriebsleitung, Werbung: Hans Hörl (114)

Verlagsleiter M&T Buchverlag: Günther Frank (212)

Vertrieb Handelsauflage: Inland (Groß-, Einzel- und Bahnhofsbuchhandel) sowie Österreich und Schweiz: Pegasus Buch- und Zeitschriften-Vertriebs GmbH, Hauptstätter Straße 96, 7000 Stuttgart 1, Telefon (0711) 764830

Bezugsmöglichkeiten: Leser-Service: Telefon 089/4613-238. Bestellungen nimmt der Verlag oder jede Buchhandlung entgegen.

Bezugspreis: Das Einzelheft kostet DM 14,-.

Druck: E. Schwend GmbH, Schmollerstr. 31, Schwäbisch Hall.

Urheberrecht: Alle im Schneider-Sonderheft erschienenen Beiträge sind urheberrechtlich geschützt. Alle Rechte, auch Übersetzungen, vorbehalten. Reproduktionen gleich welcher Art, ob Fotokopie, Mikrofilm oder Erfassung in Datenverarbeitungsanlagen, nur mit schriftlicher Genehmigung des Verlages. Anfragen sind an Michael Scharfenberger zu richten. Für Schaltungen, Bauanleitungen und Programme, die als Beispiele veröffentlicht werden, können wir weder Gewähr noch irgendwelche Haftung übernehmen. Aus der Veröffentlichung kann nicht geschlossen werden, daß die beschriebenen Lösungen oder verwendeten Bezeichnungen frei von gewerblichen Schutzrechten sind. Anfragen für Sonderdrucke sind an Peter Wagstyl zu richten.

©1985 Markt & Technik Verlag Aktiengesellschaft, Redaktion »Happy-Computer«.

Verantwortlich: Für redaktionellen Teil: Michael Scharfenberger
Für Anzeigen: Ralph Peter Rauchfuß (126).

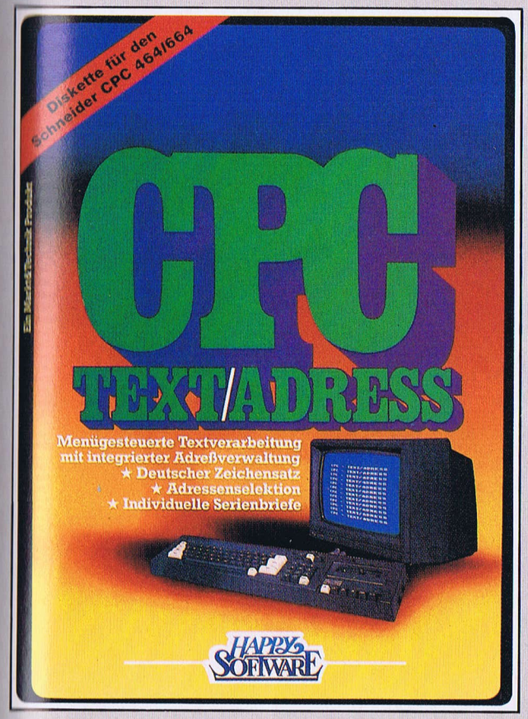
Vorstand: Carl-Franz von Quadt, Otmar Weber

Anschrift für Verlag, Redaktion, Vertrieb, Anzeigenverwaltung und alle Verantwortlichen:
Markt&Technik Verlag Aktiengesellschaft, Hans-Pinsel-Straße 2, 8013 Haar bei München, Telefon 089/4613-0, Telex 5-22052

Telefon-Durchwahl im Verlag:
Wählen Sie direkt: Per Durchwahl erreichen Sie alle Abteilungen direkt. Sie wählen 089-4613 und dann die Nummer, die in Klammern hinter dem jeweiligen Namen angegeben ist.

Aktionäre, die mehr als 25% des Kapitals halten: Otmar Weber, Ingenieur, München; Carl-Franz von Quadt, Betriebswirt, München; Aufsichtsrat: Dr. Robert Dissmann (Vorsitzender), Karl-Heinz Fanselow, Eduard Heilmayr

CPC-TEXT/ADRESS



Serienbriefe – kein Problem! Textverarbeitung und Adreßverwaltung – ein kombiniertes Paket!

Das Programm unterstützt das Diskettenlaufwerk (Wahlmöglichkeit: Speicherung Ihrer Briefe und Adressen auf Kassette oder Diskette).

Leistungsbeschreibung von CPC-TEXT:
- Menügesteuerte Bedienerführung
- Automatische Trennvorschläge
- Blocksatz; Tabulatorfunktionen; Blockoperationen
- Deutsche Tastaturanpassung; deutscher Zeichensatz
- Texteingabe im 80-Zeichen-Modus (variable Zeilenbreite)
- Eigener Funktionsteil zur Druckeranpassung
- Cursororientierter Texteditor zur problemlosen Korrektur
- Serienbrieferstellung m. individuellen Empfängeradressen u. persönlicher Briefanrede
- Ansteuerung von Druckersonderfunktionen

Leistungsbeschreibung von CPC-ADRESS:
- Feste Eingabemaske mit sieben Eingabefeldern
- Ausgabe der selektierten Adressen in eine separate Textdatei
- Auswahlmöglichkeit der Suchroutinen nach Code, Name oder Maske
- Druck auf Endlospapier oder Adreßetiketten

Minimale Hardwareanforderungen:
- Schneider CPC 464, 664
- Beliebiger Drucker mit Centronics-Schnittstelle (standardmäßige Anpassung an alle Epson-Drucker und Schneider NLQ 401)

CPC-TEXT/ADRESS gibt es auf Diskette und auf Kassette!

M&T-Programme: Ihre ganz persönlichen Problemlösungen

DM 79,–* auf Kassette, Best.-Nr. MK 242 G (sFr. 73,–/öS 711,–) (ISBN 3-89090-101-8)

DM 89,–* auf Diskette, Best.-Nr. MD 244 G (sFr. 83,–/öS 801,–) (ISBN 3-89090-145-X)

* inkl. MwSt. Unverbindliche Preisempfehlung

Und dazu die richtige Literatur:

C. Straush

CPC 464 — Programmieren in Maschinensprache
Juli 1985, 276 Seiten
Vom Speicheraufbau bis hin zum Z80-Befehlssatz wird der fortgeschrittene BASIC-Programmierer in das Innenleben seines Schneider-Computers eingeweiht. Wichtige ROM-Routinen und ausgewählte Werkzeuge wie Disassembler und Monitor werden als nützliche Utilities für die eigene Programmerstellung mitgeliefert. Alle Beispiele auf Kassette erhältlich!
Best.-Nr. MT 829
(sFr. 42,30/öS 358,80)
ISBN 3-89090-166-2 **DM 46,–**
Best.-Nr. MT 833 (Kassette)
(sFr. 19,90/öS 179,10) **DM 19,90 –***
* inkl. MwSt. Unverbindliche Preisempfehlung.

C. Straush/H. Pick

CPC 464 für Ein- und Umsteiger
Februar 1985, 260 Seiten
Über die hervorragende Qualität des CPC 464 sind sich alle Experten einig; die mitgelieferte Dokumentation läßt jedoch eine Menge Fragen offen. Der Computerneuling, ebenso aber auch der 'alte Hase', der bereits ein anderes System kennt, benötigt hier Hilfe, die ihm dieses Buch gibt. Eine BASIC-Einführung (an den Beispielen Dateiverwaltung und Textverarbeitung!) ist selbstverständlich. Darüber hinaus konzentrieren sich die Autoren jedoch auf die besonderen Eigenschaften des 464ers, seine mächtigen Befehle, die andere BASICs ganz schön alt aussehen lassen und seine begeisternden Grafik- und Musikmöglichkeiten. Ein eigenes Kapitel ist der Echtzeitverarbeitung gewidmet, die wohl auf keinem anderen Homecomputer so problemlos zu bewältigen ist.
- Starthilfe für den Anfänger; Orientierungshilfe für den Umsteiger; die ideale Ergänzung zum Handbuch.

Best.-Nr. MT 801
(sFr. 42,30/öS 358,80)
ISBN 3-89090-090-9 **DM 46,–**

Th. Erpel

CPC-BASIC-Kurs
August 1985, 300 Seiten
Dieses Buch soll den Einstieg in die Bedienung und Programmierung der Schneider-Familie (464, 664, 6128) erleichtern und richtet sich daher an alle Anwender, für die das Gebiet »Computer« noch Neuland ist.
In den ersten Kapiteln werden Bedienung und Grundlagen der Programmierung angesprochen. Anschließend folgen die gängigsten BASIC-Standardbefehle, die anhand vieler konkreter Beispiele demonstriert werden. Wie Daten problemlos gespeichert werden können, zeigt ein Kapitel, das sich speziell mit der Datenspeicherung auf Diskette und Kassette beschäftigt.
Dann folgen spezielle BASIC-Befehle, Themen wie Grafik, Soundmöglichkeiten, Windows werden ausführlich dargestellt.
In den letzten Kapiteln geht es um die Erstellung von Software, Programmiertechniken, Fehlerursachen und deren Behebung.
- Ein Buch, das für jeden Schneider CPC-Besitzer interessant ist.

Best.-Nr. MT 828
(sFr. 42,30/öS 358,80)
ISBN 3-89090-167-0 **DM 46,–**

T. Mossakowski

Das »ROM« im Schneider CPC 464/664
September 1985, ca. 500 Seiten
Ausführliche Hardware-Beschreibung: Prozessor Z80A, Videocontroller 6845 CRTC, Gate Array 20 RA 043, Sound Generator AY-3-8912, I/O-Baustein 8255 PIO, Expansion-Port.
Die ROMs: Speicheraufteilung, Interrupt-Verwaltung, Datenformate, Erweiterungs- und Änderungsmöglichkeiten. Das ROM-Listing: Betriebssystem, BASIC-Interpreter.
Best.-Nr. MT 711
(sFr. 58,90/452,40)
ISBN 3-89090-134-4 **DM 64,–**

In Vorbereitung:

Das Hardware-Buch Schneider CPC 464
September 1985, ca. 300 Seiten
Best.-Nr. MT 831
(sFr. 42,30/öS 358,80)
ISBN 3-89090-169-7 **DM 46,–**

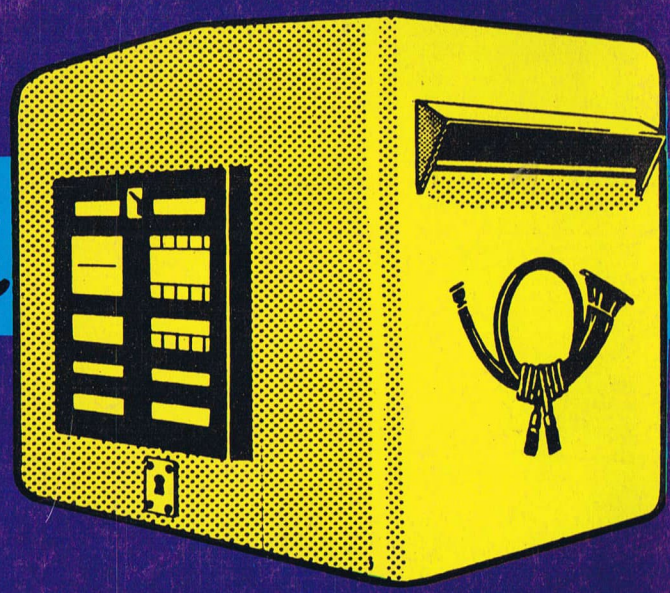
Markt&Technik-Produkte erhalten Sie bei Ihrem Buchhändler.

Bestellkarten bitte an Ihren Buchhändler oder an eine unserer Depot-Buchhandlungen. Adressenverzeichnis am Ende des Heftes! Beim Markt & Technik-

Markt&Technik
Verlag Aktiengesellschaft
Unternehmensbereich Buchverlag
Hans-Pinsel-Straße 2, 8013 Haar bei München
Schweiz: Markt&Technik Vertriebs AG, Kollerstraße 3, CH-6300 Zug, ☎ 042/223155

data media

MAILORDER



Wie bestellen?
Bestellung an DATA MEDIA senden. Lieferung erfolgt umgehend! Zahlung per Vorkasse oder Nachnahme zzgl. Porto- und Nachnahmegebühr. (Nachnahme ins Ausland ist nicht möglich.)

Schneider CPC Anwender-Software

Anwenderprogramme für CPC 464/664

Focus *UNSER RENNER!*

Programm zur perspektivischen Darstellung von 3D-Grafiken. Stellen Sie z.B. ein Haus dar und verändern Sie durch Eingabe der Winkel die Perspektive!

Cassette **79,00 DM** Diskette 3", 5 1/4" **89,00 DM**

Sekretariat

Ein Programmpaket bestehend aus: Textverarbeitung, Adressverwaltung und Faktura.

Cassette **179,00 DM** Diskette 3", 5 1/4" **189,00 DM**

Terminplaner

Planen Sie Ihre Termine! Wochen- und Monatsübersicht.

Cassette **59,00 DM** Diskette 3", 5 1/4" **69,00 DM**

Turbo Tape

Das Programm erhöht die Datenausgabe zwischen Rechner und Kassette erheblich.

Cassette **29,00 DM**

Neu **Multidatei**

Ein universelles Dateiverwaltungsprogramm.

Cassette **59,00 DM** Diskette 3", 5 1/4" **69,00 DM**

Zeichengenerator

Hilfsprogramm zum bildschirmorientierten Definieren von eigenen Zeichen. Laden, Sichern und Ausgeben von definierten Zeichen. Besonders geeignet für Adventures.

Cassette **49,00 DM** Diskette 3", 5 1/4" **59,00 DM**

Routinen, Tips und Tricks

Ca. 50 wichtige Programmroutinen für jeden Programmierer, wie z.B. Sortierroutine, Inkeyroutine, Diskettenroutine usw.

Cassette **79,00 DM** Diskette 3", 5 1/4" **89,00 DM**

Textverarbeitung

Aufwendige Textverarbeitung mit deutschem Zeichensatz und mathematischen Sonderzeichen. Schnittstelle zur Data Media Adressverwaltung vorhanden.

Cassette **79,00 DM** Diskette 3", 5 1/4" **89,00 DM**

Adressverwaltung

Kommerzielle Adressverwaltung zur Verarbeitung von Adressen. Schnittstelle zur Data Media Faktura und Data Media Textverarbeitung.

Cassette **67,00 DM** Diskette 3", 5 1/4" **79,00 DM**

Faktura

Schreibt Rechnungen, Mahnungen, Lieferscheine, Bestellungen, Quittungen und kalkuliert das Vertriebsprogramm und Verkaufspreise.

Diskette 3", 5 1/4" **89,00 DM**

Kassenbuch

Tageskassenbericht mit Einnahmen und Ausgaben, Kassenbestand bei Geschäftsschluß, Entnahmen sowie Ausgabe von Buchungsbelegen.

Cassette **79,00 DM** Diskette 3", 5 1/4" **89,00 DM**

Neu **Platinenkit**

Erstellt auf einfache Weise Layouts Ihrer elektronischen Schaltungen.
Ausgabe auf Drucker

Cassette **89,00 DM** Diskette 3", 5 1/4" **99,00 DM**

Vereinsverwaltung

Programm für Vereine zur Erleichterung der Verwaltung von Mitgliedern, Beitragszahlungen, Jubiläen, Geburtstage, Statistiken, Buchhaltung usw.

Cassette **79,00 DM** Diskette 3", 5 1/4" **89,00 DM**

Verwaltungsarchiv

Verwaltungsprogramm für Telefonnummern, Archiv für Bücher, Videocassetten, Schallplatten, Musikcassetten und Disketten.

Cassette **69,00 DM** Diskette 3", 5 1/4" **79,00 DM**

Lagerverwaltung

Lagerverwaltungsprogramm für jeden Betrieb. Automatische Artikelüberwachung, Aktualisierung der auf Datenträger gespeicherten Daten und Ausdruck von Bestands-, Bestell- und Verkaufslisten.

Cassette **79,00 DM** Diskette 3", 5 1/4" **89,00 DM**

Neu **Basic Compiler**

Übersetzt Ihre Basicprogramme in Maschinensprache und erhöht dadurch die Verarbeitungsgeschwindigkeit

Cassette **189,00 DM** Diskette 3", 5 1/4" **199,00 DM**

Neu **Gredi**

Hilfsprogramm zur Herstellung von Hires-Grafiken

Diskette 3", 5 1/4" **59,00 DM**

Neu **Abersoft Forth**

Programmiersprache Forth

Diskette 3" **148,00 DM**

Reisekosten

Eingabe aller Abrechnungsbelege, Personalnummer, Reisekostenvorschuß usw. Erstellung einer kompletten Reisekostenabrechnung für In- und Ausland unter Berücksichtigung der üblichen Tages- und Kilometerpauschalen.

Cassette **69,00 DM** Diskette 3", 5 1/4" **79,00 DM**

Videothek

Archivprogramm für Video und Film. Die Angaben umfassen: Titel, Spielzeit, Rangfolge, Regie, Darsteller usw.

Cassette **59,00 DM** Diskette 3", 5 1/4" **69,00 DM**

Münzarchiv

Archivprogramm für Münzen. Die Angaben umfassen: Katalog-Nr., Ordnungskriterien, Motive, Katalogpreise, Anzahl, Alter usw.

Cassette **59,00 DM** Diskette 3", 5 1/4" **69,00 DM**

Briefmarkenarchiv

Archivprogramm für Briefmarken. Die Angaben umfassen: Katalog-Nr., Ordnungskriterien, Motive, Katalogpreise, Anzahl usw.

Cassette **59,00 DM** Diskette 3", 5 1/4" **69,00 DM**

Bibliothek

Archivprogramm für Bücher. Die Angaben umfassen: Titel, Verlag, Autor, Auflage, Sachbücher, Fachbücher, Belletristik usw.

Cassette **59,00 DM** Diskette 3", 5 1/4" **69,00 DM**

Weitere Artikel in unserem Gesamtkatalog. Bitte anfordern (2,– DM für Rückporto beilegen).
DATA MEDIA GMBH, Bereich Mailorder, Postfach 1263, 4620 Castrop-Rauxel
Telefonische Auskünfte: 0231/125071