Einsteigen - Verstehen - Beherrschen

DM 3,80 öS 30 sfr 3,80

computer kurs

Ein wöchentliches Sammelwerk

Heft 20

Speicherorganisation

Commodore/Plus 4

Roboter: Unter Kontrolle

Elektronisches Basteln

Mugsy — das Spiel der Ganoven

computer kurs

Heft 20

Inhalt

WIE SIE JEDE WOCHE IHR HEFT BEKOMMEN

Computer Kurs ist ein wöchentlich erscheinendes Sammelwerk. Die Gesamtzahl der Hefte ergibt ein vollständiges Computer-Nachschlagewerk. Damit Sie jede Woche Ihr Heft erhalten, bitten Sie Ihren Zeitschriftenhändler, Computer Kurs für Sie zu reservieren.

Zurückliegende Hefte
Ihr Zeitschriftenhändler besorgt Ihnen gerne zurückliegende Hefte. Sie können sie aber auch direkt beim Verlag bestellen.

Deutschland: Das einzelne Heft kostet DM 3,80. Bitte füllen Sie eine Postzahlkarte aus an: Marshall Cavendish Int. Ltd. (MCI), Sammelwerk-Service, Postgiroamt Hamburg 48064-202, Postfach 105703, 2000 Hamburg 1, Kennwort: Computer Kurs

Österreich: Das einzelne Heft kostet öS 30. Bitte füllen Sie eine Zahlkarte aus an: Computer Kurs, Wollzeile 11, 1011 Wien, Postscheckkonto Wien 7857201 oder legen Sie Ihrer Bestellung einen Verrechnungsscheck bei. Kennwort: Computer Kurs.

Schweiz: Das einzelne Heft kostet sfr 3,80. Bitte wenden Sie sich an Ihren Kiosk; dort werden Sie jederzeit die gewünschten Exemplare erhalten.

Abonnement

Sie können Computer Kurs auch alle 2 Wochen (je 2 Ausgaben) per Post zum gleichen Preis im Abonnement beziehen. Der Abopreis für 12 Ausgaben beträgt DM 45,60 inkl. Mwst., den wir Ihnen nach Eingang der Bestellung berechnen. Bitte senden Sie Ihre Bestellung an: Marshall Cavendish Int. Ltd. (MCI), Sammelwerk Service, Postgiroamt Hamburg 86853-201, Postfach 105703, 2000 Hamburg 1, Kennwort: Abo Computer Kurs. Bitte geben Sie an, ab welcher Nummer das Abo beginnen soll und ob Sie regelmäßig für jeweils 12 Folgen einen Sammelordner wünschen. Bei Bestellungen aus Österreich oder Schweiz senden Sie Ihren Auftrag bitte auch an die Hamburger Adresse. Berechnung und Zahlung erfolgen in Landeswährung zum Ladenpreis.

WICHTIG: Bei Ihren Bestellungen muß der linke Abschnitt der Zahlkarte Ihre vollständige Adresse enthalten, damit Sie die Hefte schnell und sicher erhalten. Überweisen Sie durch Ihre Bank, so muß die Überweisungskopie Ihre vollständige Anschrift gut leserlich enthalten.

SAMMELORDNER
Sie können die Sammelordner entweder direkt bei Ihrem Zeitschriftenhändler kaufen (falls nicht vorrätig, bestellt er sie gerne für Sie) oder aber Sie bestellen die Sammelordner für den gleichen Preis beim Verlag wie folgt:

Deutschland: Der Sammelordner kostet DM 12. Bitte füllen Sie eine Zahlkarte aus an: Marshall Cavendish International Ltd. (MCI), Sammelwerk-Service, Postgiroamt Hamburg 48064-202, Postfach 105703, 2000 Hamburg 1, Kennwort: Sammelordner Computer Kurs.

Österreich: Der Sammelordner kostet öS 98. Bitte füllen Sie eine Zahlkarte aus an: Computer Kurs Wollzeile 11, 1011 Wien, Postscheckkonto Wien 7857201 oder legen Sie Ihrer Bestellung einen Verrechnungsscheck bei. Kennwort: Sammelordner Computer Kurs

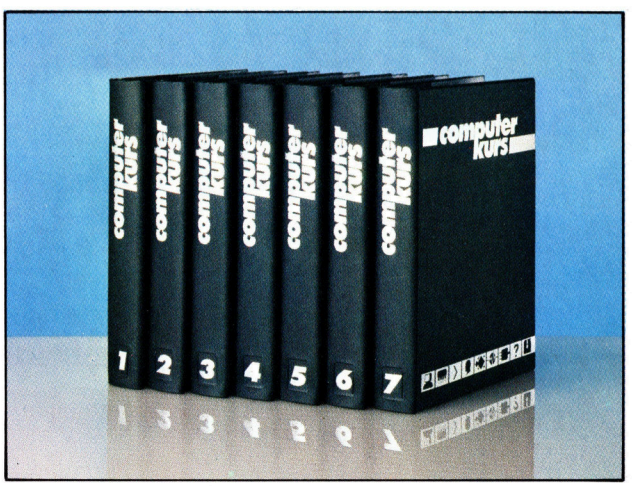
Schweiz: Der Sammelordner kostet sfr 15. Bitte wenden Sie sich an Ihren Kiosk; dort werden Sie jederzeit die gewünschten Exemplare erhalten.

INHALTSVERZEICHNIS

Alle 12 Hefte erscheint ein Teilindex. Die letzte Ausgabe von Computer Kurs enthält den Gesamtindex — darin einbezogen sind Kreuzverweise auf die Artikel, die mit dem gesuchten Stichwort in Verbindung stehen.

Redaktion: Winfried Schmidt (verantw. f. d. Inhalt), Joachim Seidel, Elke Leibinger, Susanne Brandt, Uta Brandl (Layout), Sammelwerk Redaktions-Service GmbH, Paulstraße 3, 2000 Hamburg 1

Vertrieb: Marshall Cavendish International Ltd., Heidenkampsweg 74, 2000 Hamburg 1, Tel.: 040/23 40 85



Druck: E. Schwend GmbH, Schmollerstraße 31, 7170 Schwäbisch Hall

Viermal besser

Der Commodore 64 ist einer der weltweit bestverkauften Computer. Commodores neuester Heimcomputer, der Plus/4, stellt sich gleich auf verschiedene Arten verbessert dar: mit besserem BASIC, vier integrierten Applikations-Programmen und 64 K voll nutzbarer Speicherkapazität.

Commodore betont, daß der Commodore Plus/4 parallel zum Commodore 64 verkauft wird und diesen nicht ablösen soll. Doch im Vergleich zum C 64 weist das neue Modell so viele Verbesserungen auf, daß es den Erfolg seines Vorgängers noch weit übertreffen könnte.

Herz des Plus/4 ist der 7501-Microprozessor, eine Weiterentwicklung des 6502. Dieser Chip ist so konstruiert, daß er mehr als 64 KByte Speicherkapazität bereitstellt. Daher hat der Rechner Platz für ein beachtliches BASIC und dennoch genügend RAM zur freien Verfügung für den Benutzer. Unterm Strich stehen 64 KByte zur Verwendung in BASIC-Programmen bereit. Diese Kapazität verringert sich allerdings auf 50 KByte bei Verwendung von Grafiken. Damit leistet der Rechner mehr als viele andere Microcomputer.

Der Plus/4 erhielt eine ausgezeichnete Variante des Microsoft-BASIC. Die Leistungsfähigkeit der Grafik- und Soundbefehle ist erstaunlich hoch. Im Grafik-Modus erzeugt der DRAW-Befehl Punkte oder Linien; jede Kontur kann mit dem PAINT-Befehl farbig gefüllt werden. Der Befehl BOX erlaubt das Zeichnen von Quadraten und Rechtecken, sowohl im Umriß als auch flächig. Der CIRCLE-Befehl erweist sich als ungewöhnlich vielseitig. Neben dem Zeichnen von Kreisen können Ovale durch Festlegung ihrer Höhe und Breite erzeugt werden. Um Bögen zu konstruieren, werden Ausschnitte dieser Ovale gezeichnet. Dazu müssen lediglich die Start- und die Stop-Position angegeben werden.

Zusätzliche BASIC-Befehle

Zur Verstärkung des integrierten BASIC hat man eine Reihe bemerkenswerter Befehle integriert. So erzeugt der Befehl AUTO bei der Programmeingabe automatisch Zeilennummern. RENUMBER dient dazu, das Programm mit neuen Zeilennummern zu versehen. Mit VERIFY läßt sich überprüfen, ob die Programme auf Cassette oder Diskette abgespeichert wurden. Für den Einsatz der Diskettenstation hat man eine Reihe neuer Befehle entwickelt. Commodore hofft, daß gerade Plus/4-Besitzer überdurchschnittlich viele dieser Floppy-Laufwerke kaufen.

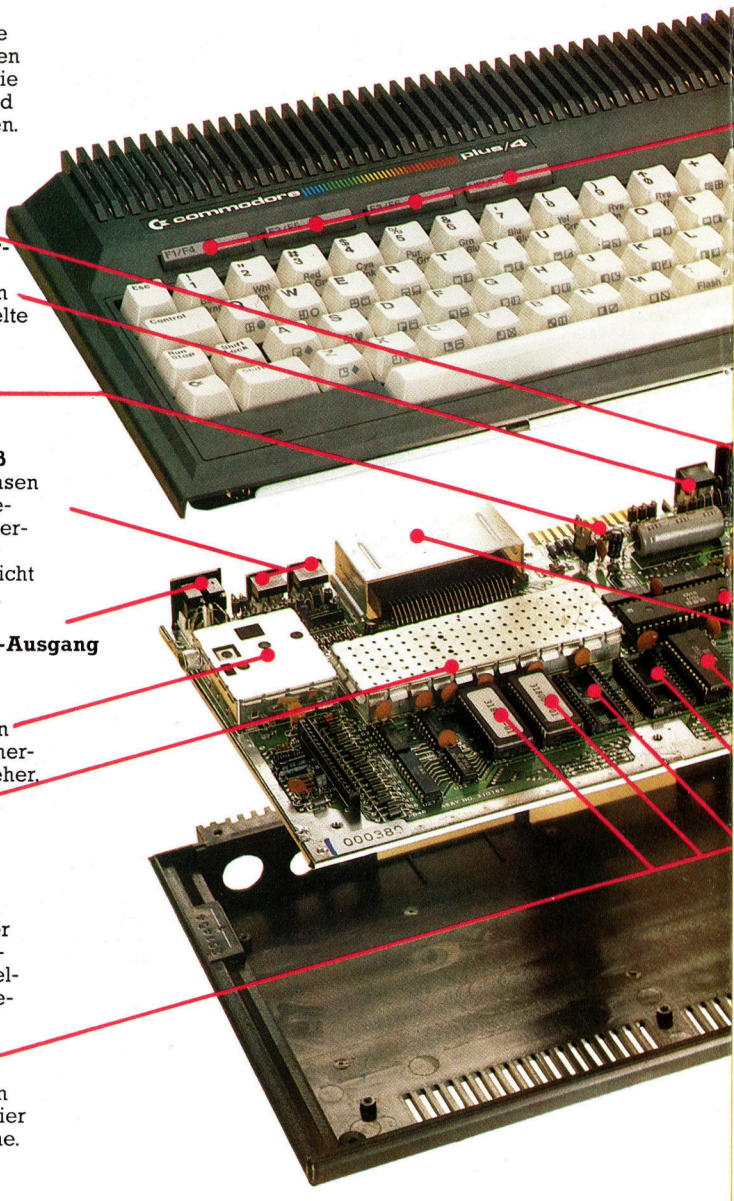
Commodores lange erwartetes Nachfolgemodell für den C 64 ist mit all dem ausgestattet, was zur Norm der neuen Rechnergeneration gehört: Design und Cursor-Steuerungstasten nach MSX-Vorbild, großer Speicher und integrierte Software. Die Konkurrenz ist jedoch groß, und die potentiellen Käufer zeigen sich informierter denn je. Auch ein Unternehmen wie Commodore hat keine Dauergarantie für seine Führungsposition im Markt. Diese Erkenntnis schlägt sich in dem Aussehen und der Ausstattung des Plus/4 nieder.

Die Textdarstellung beim Plus/4 erfolgt mit 40 mal 25 Zeichen. Der Anwender kann zwei Punkte auf dem Bildschirm als „Fenster"-Ekken definieren. Der komplette Text, ob nun Listing oder Befehle, erscheint anschließend in diesem Fenster. Der Rest des Bildschirms bleibt von den Eingaben unberührt.

Die Plus/4-Tastatur reagiert ausgesprochen empfindlich. Zur Eingabe reicht ein kaum spürbares Antippen. Zahlreiche Grafikzeichen können über die Tastatur erzeugt werden. Oberhalb des Tastenfeldes liegen vier Funktionstasten. Nach Einschalten des Rechners sind sie automatisch mit den am häufigsten verwendeten Befehlen belegt. Der Anwender hat aber die Möglichkeit, neue Befehle in einem Umfang von bis zu 128 Zeichen zu definieren und die Funktionstasten damit zu belegen. Durch gleichzeitiges Drücken der Funktions- und der „Shift"-Taste können insgesamt acht Funktionen erzeugt werden.

Dank des großen Speicherplatzes ließen sich vier Anwendungsprogramme in den Plus/4 integrieren: eine Textverarbeitung, ein

QL oder Plus/4?
Ein Vergleich der beiden Systeme scheint nicht möglich. Der QL hat integrierte Micro-Drives, mehr Speicherplatz, ein SuperBASIC und ausgezeichnete Softwareunterstützung in vernünftiger Relation zum Preis. Der Plus/4 zeichnet sich durch die Qualität seiner Tastatur aus. Sicher wird der Plus/4 schon aufgrund des Firmennamens ein Renner.

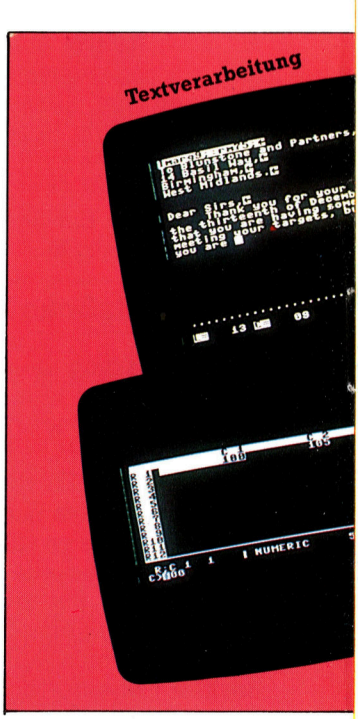


Spreadsheet (Tabellenkalkulation), eine Dateiverwaltung sowie ein Grafikprogramm. Die Textverarbeitung ist allerdings anwenderunfreundlich. Der Plus/4 kann nur 40 Zeichen auf dem Bildschirm darstellen. Viele Drucker aber drucken 80 Zeichen pro Zeile. Um dem Rechnung zu tragen, verschiebt sich die Bildschirmdarstellung nach Erreichen der 37. Spalte seitlich. Das Programm bietet allerdings Formatierungsmöglichkeiten. So gibt es Befehle, um Fußnoten zu setzen oder Text auszugleichen. Sie werden jedoch nur beim Textausdruck wirksam. Mit den Befehlen SEARCH und REPLACE lassen sich bestimmte Wörter oder Sätze innerhalb eines Textes finden und gegebenenfalls austauschen.

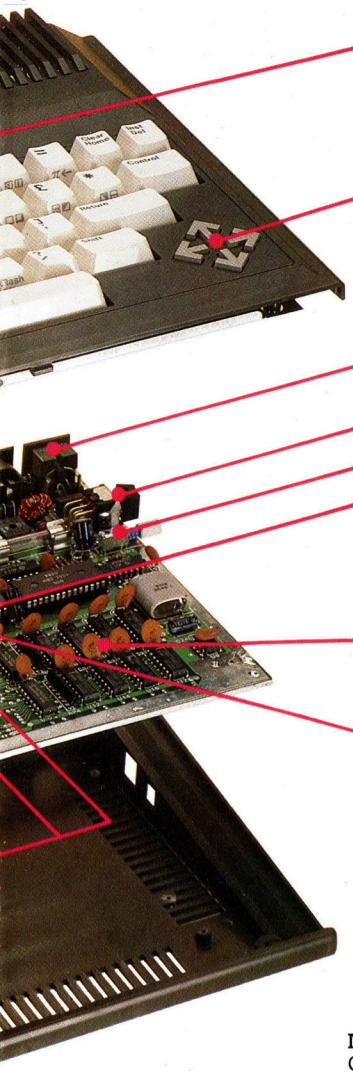
Einfacher in der Anwendung scheint das Spreadsheet. Seine Begrenzung ergibt sich aber ebenfalls aus der 40-Zeichen-Darstellung. Das bedeutet, daß über die gesamte Bildschirm-Breite nur drei Spreadsheet-Teile und in der Höhe nur zwölf sichtbar sind, obwohl 17 mal 50 verarbeitet werden können.

Ebenso enttäuschend ist das Grafikprogramm. Es kann lediglich eine Zahlenreihe aus dem Spreadsheet mittels Blockgrafik in „Balken" umwandeln. Diese sind in die Textverarbeitung integrierbar und lassen sich sowohl auf dem Bildschirm darstellen als auch ausdrucken.

Textverarbeitung wie Spreadsheet sind beim Grundmodell zwar lauffähig, doch die Speicherung der Ergebnisse (Daten) ist nur mit einer Diskettenstation möglich. Für Anwender, die lediglich über einen Cassettenrecorder als Massenspeicher verfügen, sind die Programme wertlos. Die Dateiverwaltung arbeitet ausschließlich mit einer Diskettenstation. Zunächst wird ein Standardformat definiert, das auf Diskette als Formular gespeichert werden muß. Erst dann sind die eigentlichen Daten einzugeben. Daraus folgt, daß eine Diskette nur für eine Datei zur Verfügung steht und das Datei-Format nach Speicherung nicht verändert werden kann. In jeder Datei lassen sich bis zu 999 Informationen, bestehend aus jeweils 17 Feldern mit 38 Zeichen, festhalten. Die Software ist also enttäuschend, da sie sich für professionelle Anwendung – etwa im Geschäftsbereich – als unzureichend erweist und den Besitz einer Diskettenstation voraussetzt. Der ernsthafte private Anwender wird dagegen über den integrierten Maschinensprache-Monitor Tedmon glücklich sein, da mit dem Befehl MONITOR die Maschinensprache aufgerufen werden kann.



Mit dem Grundgerät stellte Commodore einiges an Peripherie für den Plus/4 her. Am interessantesten für die meisten Anwender dürfte der Cassettenrecorder sein. Wie bei allen anderen Commodore-Heimcomputern ist auch für den Plus/4 ein spezieller Commodore-Recorder erforderlich. Die Buchse stimmt aber nicht mit denen älterer Modelle überein,



Funktionstasten
Jede dieser Tasten kann mit zwei Befehlen von je bis zu 128 Zeichen belegt werden.

Cursor-Steuerungstasten

Stromversorgung

Ein/Aus-Schalter

Reset-Schalter

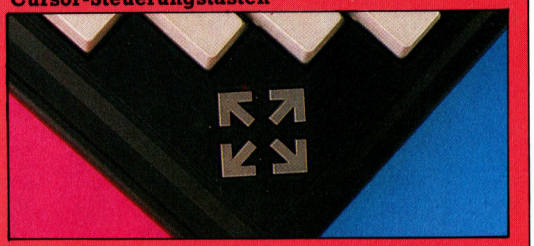
7501 CPU

64 K RAM

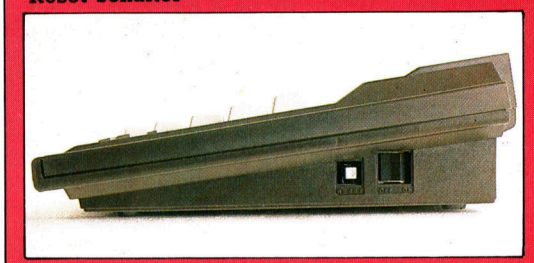
Modul-Steckplatz
Cartridge-Software sowie eine „schnelle" Diskettenstation werden an dieser Stelle angeschlossen.

Funktionstasten

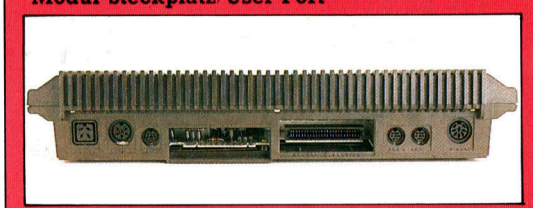
Cursor-Steuerungstasten



Reset-Schalter



Modul-Steckplatz/User Port



Neue Elemente
Überzeugte Commodore-Freunde wird sicherlich freuen, daß der Plus/4 mit Escape- und Reset-Taste ausgestattet wurde. Neu sind auch die Cursor-Steuerungstasten, die Funktionstasten und die Help-Taste. Stromversorgung, Recorder- und Joystickanschlüsse unterscheiden sich von denen des C 64 und des VC 20.

Commodore Plus/4

PREIS
ca. 1350 Mark

ABMESSUNGEN
67 × 203 × 338 mm

ZENTRALEINHEIT
7501

TAKTFREQUENZ
0,9 oder 1,8 MHz

SPEICHERKAPAZITÄT
64 K RAM; 32 K ROM

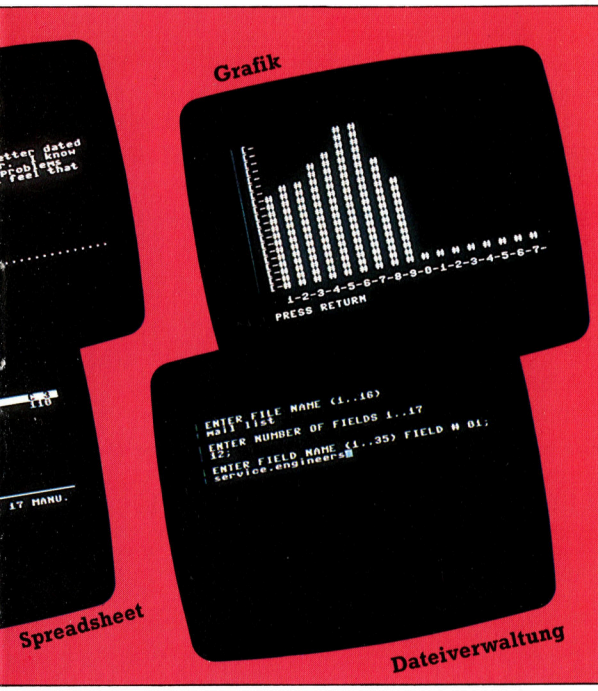
BILDSCHIRM-DARSTELLUNG
Text: 40 × 25; Grafik: 320 × 200 Zeichen in 121 Farben

SCHNITTSTELLEN
2 Joystick-Anschlüsse, serielle Schnittstelle, Cassetten-Interface, Cartridge/Parallel-Anschluß

PROGRAMMIERSPRACHEN
BASIC

TASTATUR
67 Schreibmaschinentasten einschließlich vier Funktionstasten

DOKUMENTATION
Annehmbare Handbücher, die Programmiergrundkenntnisse sowie den Umgang mit der integrierten Software erläutern und vermitteln. Teilweise unglücklich formuliert.



Versteckte Kosten

Die in den Plus/4 auf ROM-Basis integrierte Software beinhaltet eine Textverarbeitung, ein Spreadsheet, eine Dateiverwaltung sowie ein Grafikprogramm. Der Vorteil dieses Programmpaketes liegt in der Kombinierbarkeit der Programme untereinander. Ohne Diskettenstation aber sind sie nutzlos. Damit steigen die Gesamt-Kosten für das System beträchtlich.

so daß ein neues Gerät gekauft werden muß. Die Joystick-Anschlüsse sind beim Plus/4 ebenfalls verändert.

Für den Rechner wird eine geringfügig modifizierte Version der langsamen alten Commodore-Diskettenstation angeboten. Außerdem entwickelt Commodore eine „schnellere" Floppy, die an den Cartridge-Schacht des Rechners anzuschließen ist. Immerhin stehen für den Micro gleich fünf Commodore-Drucker zur Verfügung: ein Schönschreibdrucker, zwei normale Matrixdrucker, ein Farb-Matrixdrucker sowie ein Vierfarb-Drucker/Plotter.

Wenngleich der Plus/4 um rund 300 Mark teurer als der C 64 ist, scheint der Preis in Anbetracht des verbesserten BASIC und des zusätzlichen Speicherplatzes gerechtfertigt. Ein Nachteil des Rechners ist zweifelsfrei der derzeitige Mangel an Software. Für ein so erfolgreiches Unternehmen wie Commodore dürfte es aber langfristig gesehen kein Problem sein, die erforderliche Softwareunterstützung zu bekommen.

Klangspektrum

Das BASIC des Commodore 64 wird den erstaunlichen Klangeigenschaften des Gerätes nicht gerecht.

Im Vergleich mit anderen Heimcomputern verfügt der Commodore 64 über sehr gute Klangfähigkeiten. Diese liefert ein Spezialchip, der „Sound Interface Device" oder „SID" genannt wird.

Der SID verfügt über Möglichkeiten der Klangerzeugung, die denen eines einstimmigen Synthesizers nicht nachstehen. Mit drei Oszillatoren erreicht er einen Tonumfang von insgesamt acht Oktaven (0 bis 3900 Hz), der in 65536 Schritte unterteilt ist. Jeder Oszillator kann vier Schwingungsformen produzieren (Dreieck-, Sägezahnschwingungen, variable Impulse und Rauschen). Es gibt eine Hauptsteuerung für die Lautstärke (von 0 bis 15), eine Oszillator-Synchronisation und Hüllkurvengeneratoren für die ADSR-Steuerung jedes Oszillators. Der SID besitzt außerdem die Möglichkeit der Ringmodulation und hat programmierbare Filter für High Pass, Band Pass und Low Pass, einen Notch-Filter (mit dem die Ausgabe eines engen Frequenzbandes blockiert werden kann), variable Resonanz, Hüllkurvenfilter, zwei Analog/Digital-Schnittstellen für den Anschluß von Reglern und einen Audioeingang für weitere SID-Chips. Externe Audiosignale lassen sich ebenfalls in die Tonausgabe des SID mischen.

Im Rahmen dieses Artikels ist es nicht möglich, auf die Funktionsweise aller Fähigkeiten einzugehen, aber wir können erklären, was die Begriffe im einzelnen bedeuten. Die vielleicht wichtigste Komponente ist die Synchronisation der Oszillatoren, mit der sich die harmonische Verbindung zweier Signale (in diesem Fall zweier unterschiedlicher Stimmen) herstellen läßt, wobei als Ergebnis ein einziger Ton mit komplexem Schwingungsmuster entsteht.

Die Veränderung eines Signals durch ein anderes wird Modulation genannt, wobei entweder die Frequenz oder die Amplitude (Lautstärke) beeinflußt wird. Eine Ringmodulation ist die Amplitudenmodulation einer Stimme durch eine andere. Das Ergebnis ist ein klarer, aber harter Ton mit dissonanter Wirkung, womit sich glockenähnliche Klänge wie die der lateinamerikanischen Steeldrums erzeugen lassen. Töne dieser Art haben sogenannte „nicht-harmonische" Obertöne.

Mit Filtern werden bestimmte Frequenzbereiche eines Signals ausgeblendet. Der Commodore besitzt eine ganze Anzahl davon: Low-Pass-Filter blockieren alle Frequenzen, die eine bestimmte Höhe überschreiten. Band-Pass-Filter schneiden alle Frequenzen ab, die oberhalb und unterhalb eines bestimmten Bereiches liegen, während Notch-Filter genau das Gegenteil bewirken – sie blockieren nur das angegebene Frequenzband. High-Pass-Filter schneiden alle Frequenzen ab, die eine bestimmte Höhe nicht erreichen. Dabei werden mit der „variablen Resonanz" die Frequenzen der Schnittpunkte stärker herausgehoben. Hüllkurvenfilter sind ein Spezialfall: Anders als bei den zuvor erwähnten Filtern werden dabei die digitalisierten ADSR-Werte der Hüllkurve 3 vom SID-Chip eingelesen und derart auf ein Signal projiziert, daß sich die harmonische Struktur des Tones während seines Verlaufs verändert.

Steuerregister

Mit diesen hochentwickelten Möglichkeiten lassen sich auch komplizierte Klänge mit interessanten Effekten verwandeln, und herkömmliche Instrumente können täuschend ähnlich nachgeahmt werden. Leider unterstützt das mitgelieferte V2-BASIC von CBM keine Befehle zur Steuerung von SID. Nur über PEEK und POKE lassen sich die 29 Steuerregister des Spezialchips ansprechen, womit auch die einfachsten Effekte sehr umständliche Anweisungen erfordern.

Eine vollständige Beschreibung der Steuerregister des SID würde ein ganzes Heft des Computerkurses füllen. Das nebenstehende Programmbeispiel vermittelt jedoch ein akustisches Beispiel über die Sound-Möglichkeiten.

Obwohl das Programm einen Umfang von 22 Zeilen hat, kann es nur fünf Töne auf einem einzigen Oszillator spielen. Zeile 20 trennt die Filter von den Oszillatoren, Zeile 30 setzt die Gesamtlautstärke auf das Maximum, und die Zeilen 40 und 50 legen eine klavierähnliche Hüllkurve fest. In Zeile 80 wird die Tonfrequenz bestimmt, Zeile 90 und 100 lösen den ADSR-Zyklus aus, beenden ihn und wählen für die Stimme 1 die Sägezahnschwingung an. Die Tondauer wird über die FOR . . . NEXT-Schleifen der Zeilen 100, 120 und 140 festgelegt.

Auf dem Commodore 64 lassen sich Melodien nur recht umständlich programmieren. Einfacher geht es mit kommerziell angebotenen Editierprogrammen, die im Maschinencode geschrieben sind und die hervorragenden Eigenschaften des Commodore 64 voll zum Einsatz bringen können.

```
10 SID=54272
20 POKESID+23,0
30 POKESID+24,15
40 POKESID+5,40
50 POKESID+6,201
60 FOR N=1TO5
70 READ FH,FL,D
80 POKESID+1,
   FH:POKESID,
   FL:REM *TON
   SPIELEN*
90 POKESID+4,33
100 FORI=1TO300*
   D:NEXT I
110 POKESID+4,32
120 FORI=1TO100:
   NEXT I
130 NEXT N
140 FORI=1TO2000:
   NEXT I
150 POKESID+24,0
160 REM**FH FL D**
170 DATA 57,172,1
180 DATA 64,188,1
190 DATA 51,97,1
200 DATA 25,177,1
210 DATA 38,126,2
220 END
```

PM-Grafiken

Die Player-Missile-Grafik bringt die Fähigkeiten der Atari-Computer richtig zur Geltung.

Die Player-Missile- oder „PM"-Grafik ist eine wesentliche Komponente der Grafikfähigkeiten von Atari. Sie funktioniert ähnlich wie die Sprite-Grafik des Commodore 64 und gibt dem Programmierer die Möglichkeit, bis zu acht Figuren mit hoher Auflösung zu entwerfen und zu steuern. Die Figuren lassen sich unabhängig von der Hintergrunddarstellung bewegen. Sie können vor oder hinter anderen Bildelementen erscheinen und somit dreidimensionale Effekte hervorrufen. PM-Grafik läßt sich schnell und kontinuierlich über den Schirm bewegen und ist daher ideal für die Programmierung von Spielen geeignet.

Wie bei allen Spritesystemen liegt das Geheimnis der PM-Grafik in der speziell darauf ausgerichteten Hardware. Spezialregister steuern Bewegung, Farbe und Bildschirmdarstellung der PM-Objekte. Der Programmierer braucht nur die Werte dieser Register zu verändern, wenn er die Figuren bewegen will. Von BASIC aus läßt sich diese Wirkung mit dem POKE-Befehl erzielen.

Die Figuren („Players") sind aus einem vertikalen Streifen aufgebaut, der acht Pixel breit und 128 oder 256 Pixel hoch ist. Jede Zeile der Streifen ist im Computer als ein Byte gespeichert. Die Form der Figur läßt sich mit POKE ähnlich wie bei frei definierbaren Zeichen festlegen. Auf diese Weise können bis zu vier Figuren aufgebaut werden, wobei jede Figur 256 oder 128 Bytes im Speicher belegt.

Jeder dieser vier Figuren ist ein „Missile" mit einer Breite von 2 Bits zugeordnet, deren Formen ebenfalls mit POKE an bestimmte Bereiche des Speichers gesetzt werden. Der Programmierer kann den RAM-Bereich dafür frei wählen, muß allerdings den entsprechenden Zeiger (Pointer) auf die Anfangsadresse des Bereichs setzen.

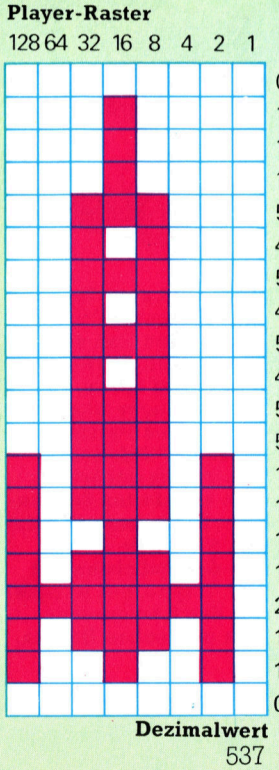
Die vertikale Auflösung mit der Einheit von einem Pixel benötigt doppelt soviel Speicherplatz wie die Auflösung mit der Grundeinheit von zwei Pixeln. Das folgende Programm legt die Figur 1 (Player 1) als Raumschiff mit einer Auflösung von zwei Pixeln fest:

```
10 REM *** FIGUR DEFINIEREN***
20 P=PEEK (106)-8:REM SETZT P AUF 2K
   UNTER DER OBERGRENZE DES RAM
30 POKE 54279,P:REM SETZT DEN
   POINTER DES PM-SPEICHERBEREICHS
40 BASE=256*P:REM SETZT DIE
   GRUNDADRESSE DES PM-BEREICHS
50 FOR I=BASE+ 512 TO BASE+640
60 POKE I,0:REM SPEICHERBEREICH
   DER FIGUR 0 LOESCHEN
70 NEXT I
80 FOR I =BASE+512+50 TO
   BASE+530+50
90 READ A:POKE I,A:REM FIGUR
   FESTLEGEN
100 NEXT I
110 DATA 16,16,16,56,40,56,40,56,40
120 DATA 56,56,186,186,146,186,254,
   186,146
```

Jeder Player-Figur sind bestimmte Register zugeordnet, die die Farbe, horizontale Position und Größe enthalten, wobei die Größe des Players auf das Doppelte oder Vierfache ausgedehnt werden kann. Weitere Register steuern die Beziehung des Players zu den Elementen des Hintergrundes. Die Missiles nehmen zwar die Farbe der zugehörigen Figur an, können ihre Größe aber unabhängig von ihr verändern. Für die Spielprogrammierung gibt es weitere Register, mit denen Zusammenstöße zwischen Players, Missiles und Hintergrund angezeigt werden. Da keine Register für die vertikalen Positionen der Player-Figuren und Missiles vorhanden sind, lassen sich vertikale Bewegungen nur über die Verlegung der Bitmuster programmieren. In der Assemblersprache ist dies ein einfacher Vorgang.

Die Player-Missile-Grafik erweitert die Möglichkeiten des Grafiksystems von Atari erheblich, läßt sich aber nicht so flexibel und leicht einsetzen wie die Sprites des Commodore 64. Das folgende Programm schließt an unser erstes Beispiel an. Es färbt das Raumschiff rosa und bewegt es von links nach rechts über den Bildschirm.

```
130 POKE 559,46:REM PM-2-LINIENDAR-
    STELLUNG EINSCHALTEN
140 POKE 53277,3:REM PM-DARSTEL-
    LUNG EINSCHALTEN
150 POKE 704,88:REM PLAYER 0 ROSA
    EINFAERBEN
160 GRAPHICS 0
170 SETCOLOUR 2,8,2:REM HINTER-
    GRUND DUNKELBLAU EINFAERBEN
180 FOR I=0 TO 320
190 POKE 53248,I:REM HORIZONTALE
    POSITION SETZEN
200 NEXT I
210 END
```



Vor der Programmierung muß eine Figur zunächst gezeichnet und die Dezimalwerte jeder einzelnen Pixelreihe müssen berechnet werden.

Mugs Spiel

„Mugsy" ist ein Strategie-Spiel von dem englischen Software-Hersteller Melbourne House, jener Gesellschaft, die auch das erfolgreiche Grafik-Abenteuer „The Hobbit" produzierte.

In diesem ungewöhnlichen Spiel schlüpft der Spieler in die Rolle von Mugsy, einem Bandenboß, der in der amerikanischen Gangster-Ära der dreißiger Jahre eine „Schutz-Organisation" leitet. Seine Aufgabe besteht darin, verschiedene Entscheidungen zu fällen. So etwa, wie viele „Kunden" der Bande „bedient" werden sollen, wieviel Geld für die Munition der Organisation ausgegeben werden darf und welcher Betrag für die Bestechung der Polizei zur Verfügung steht. Wird nicht genügend Geld für Munition ausgegeben, ist die Bande erledigt. Bekommt die Polizei zu wenig, beschlagnahmt sie das Vermögen der Gangster.

Hauptdarsteller auf dem Bildschirm ist Louey, Mugsys rechte Hand. Zu Beginn des Spiels erläutert Louey kurz die Regeln. Der Spieler trifft seine Entscheidungen durch Tastatureingabe und wartet dann auf die Ergebnisse. In der Zwischenzeit werden animierte Grafikfolgen dargestellt. Das erste Bild bringt eine Gangsterkneipe, in der ein Gast auf einen Rivalen schießt. Das zweite zeigt eine Straßenszene mit einem Auto, aus dessen Fenster sich ein Gangster lehnt und mit einem Maschinengewehr feuert.

Dies sind zwei Szenen aus „Mugsy": Die erste zeigt einen Teil der Frage- und Antwort-Phase des Spiels, in der Louey Mugsy über die derzeitigen Preise für „Kunden" informiert. Die zweite gehört zu den animierten Sequenzen. Bemerkenswert dabei die weiße Kontur um den Polizisten auf der Treppe, ein Beispiel für das Hervorheben von grafischen Details.

Danach erscheint Louey wieder und erstattet Bericht über das abgelaufene „Geschäftsjahr". Vorausgesetzt, daß für alle anstehenden Zahlungen genügend Rücklagen bereitgestellt wurden, sollte das Jahr erfolgreich enden. Die nächste Runde wird wiederum durch das Frage- und Antwortspiel eingeleitet.

Am Ende des Spiels wird der Punktestand des Spielers in Prozenten ausgewiesen. Die Punktzahl hängt davon ab, wieviel „Kohle" gemacht, wie viele „Klienten" gewonnen wurden und wie lange Mugsy überlebt hat.

Abgesehen von den Anweisungs-Bildschirmen, an deren unterem Rand in einem schmalen Fenster die gegenwärtige Anzahl der „hoods" und „dough in da safe" etc. abzulesen ist, handelt es sich bei den Screens ausschließlich um grafische Darstellungen.

Eine intensivere Untersuchung der „Mugsy"-Grafiken zeigt, daß die Programmierer mehrere platzsparende Techniken verwendet haben. So wurden die Bilder überwiegend aus geraden Linien zusammengesetzt, und man ging mit der Farbe sehr behutsam um.

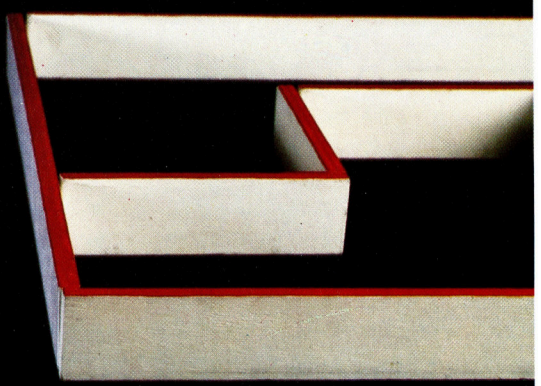
Aufgrund der besonderen Farb-Datenspeicherung des Spectrum sind die verwendeten Methoden vorgegeben. So gibt es ein Problem bei der animierten Straßenszene, in der ein Auto fährt und zugleich ein Gesicht aus dem Fenster schaut. Da mit dem Spectrum nur die Darstellung zweier Farben im selben Charakterformat möglich ist, mußte eine „Maske" entwickelt werden, um Farben schnell verändern zu können. Andernfalls würde das Gesicht ständig die Farbe wechseln. Farbdetails (FLASH, BRIGHT, INK und PAPER) sind in einem einzigen Byte zusammengefaßt. Um die Vordergrundmaske zu produzieren, ist das Element INK zu verändern, da es in der Wichtigkeit der Bytes an drittletzter Stelle steht. Das Byte wird zunächst mit 248 ANDed, um die INK-Farbe auf Null zu setzen, und dann wieder mit der neuen INK-Farbe zur Erzeugung einer neuen Maske ANDed. So wechselt das Gesicht entsprechend dem Auto tatsächlich die Farbe – und wieder zurück. Dies aber so schnell, daß das Auge die einzelnen Übergänge nicht wahrnehmen kann.

Die Grafiken von „Mugsy" sind überragend. Das Spiel selbst dagegen fällt ab, da die Aktion minimiert ist und der Spieler bald schon weiß, wie er mit den spielbestimmenden Faktoren auf Dauer richtig umgehen muß. „Mugsy" ist für Programmierer beispielhaft, die die Unterbringung hochauflösender Grafik auf kleinstem Raum verstehen und lernen wollen.

Mugsy: Für Spectrum 48 K, DM 49,80
Verlag: Melbourne House, Church Yard, Tring, Herts (Vertrieb: ISS/Fachhandel)
Autoren: Phillip Mitchell, Greg Cull, Clive Barrett, Russell Comte
Joystick: Nicht erforderlich
Programm: Cassette

Von Mäusen und Labyrinthen

Die Micro-Maus-Wettbewerbe, bei denen „Robotermäuse" durch ein Labyrinth gelangen müssen, sind eine wertvolle Quelle für praktische Erfahrungen vieler Amateur-Roboterfreunde. Der hier gezeigte „Quester" von David Buckley ist mit einer Reihe verschiedener Sensoren (optisch, akustisch und berührungsempfindlich) ausgestattet.



Jeder Micro-Maus steht ein bestimmter Übungszeitraum zur Verfügung, in dem sie sich mit dem Labyrinthgrundriß „vertraut" machen kann. Dabei darf sie allerdings nicht ferngesteuert werden. Danach muß sie in kürzester Zeit das Labyrinth durchdringen. Üblicherweise ist das Ziel, in die Mitte des Labyrinths zu gelangen.

Robotersteuerung

Mit den drei Grundmethoden der Roboterfortbewegung haben wir uns bereits befaßt und auch aufgezeigt, warum vorwiegend Elektromotoren verwendet werden. Wie aber kann man einen Roboter ohne ständigen Blickkontakt steuern?

Die einfachste Methode besteht darin: Der Roboter wird mit einer mechanischen Vorrichtung ausgestattet, die dem Weg entsprechend geformte Karten „liest", die sich im Roboter befinden. Der Umriß der Karte wird von einer kleinen Nocke abgetastet, die verschiedene Hebel zur Steuerung auslöst. Diese Methode fand bei Spielzeugautos und -robotern Anwendung. Das Steuerungsprogramm schnitt man mit einer Schere in einen Kartonstreifen, und der Roboter bewegte sich dem gezackten Rand entsprechend.

Andere Roboter waren mit Einrichtungen versehen, die sie mittels elektromechanischer Relais auf einem vorgegebenen Weg hielten. Diese mechanischen Steuerungskontrollen werden nur selten eingesetzt, da die Bauteile teuer sind und nicht präzise genug arbeiten.

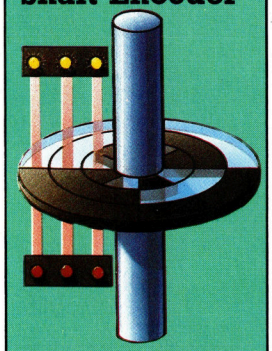
Eine der heute weit verbreiteten Methoden ist, den Roboter über eine spezielle Kontaktspur zu steuern. Dies ähnelt dem bei Modellrennautos angewandten Verfahren, bei dem sich unter dem Auto ein Stift befindet, der in der Streckenführung läuft. Die für Roboter am häufigsten benutzten Methoden sind jedoch die Steuerung über eine Linie am Boden und die Steuerung mittels Draht.

Ein oder zwei Lichtsensoren?

Roboter können einer Linie mit Hilfe eines Lichtsensors (das ist üblicherweise entweder eine Fotozelle oder ein Infrarot-Sensor) folgen. Der Sensor erkennt, ob sich der Roboter über „hellem" oder „dunklem" Boden befindet. Ist die Bodenfarbe dunkel, die Linie aber hell, wird die Sensorausgabe immer dann am stärksten sein, wenn sich der Sensor direkt über der Linie befindet. Wenn der Roboter also immer der Strecke folgt, die den stärksten elektrischen Impuls auslöst, bleibt er stets auf der gewünschten Linie.

Diese Technik birgt ein Problem: Was tut der Roboter, wenn der sensorische Impuls aufhört

Shaft Encoder



Mit einem Shaft Encoder ist es einem Roboter möglich, die zurückgelegte Strecke festzustellen. Dazu wird gemessen, wie weit sich die Achsen seiner Räder gedreht haben. Das Gerät besteht aus einer kalibrierten Scheibe. Die Platte ist in mehrere konzentrische Ringe unterteilt, deren Sektoren durchsichtig oder lichtundurchlässig sind. Durch eine Lichtquelle und eine Fotozelle für jeden Ring wird die exakte Achsenumdrehung gemessen. Die Illustration zeigt einen Shaft Encoder mit drei Ringen, womit die Binärzahlen von 000 bis 111 chiffriert werden können.

und somit anzeigt, daß die Linie verlassen wurde? Ist der Roboter nur mit einem Sensor ausgestattet, wird er sich so lange im Kreis bewegen, bis der Sensorimpuls wieder zunimmt. Erst dann kann er sich in der gewünschten Richtung weiterbewegen. Dieses Verfahren ist nicht so zufallsabhängig, wie es scheint. Bewegte sich der Roboter beispielsweise nach links, als der Sensorimpuls geringer wurde, ist eine Bewegung nach rechts logisch, um die Linie wiederzufinden. Daher ist für den Roboter „anzunehmen", daß die ursprünglich vorgesehene Richtung sich irgendwo zwischen dem (Links)-Kurs, dem er vor dem Abkommen von der Linie folgte und dem (Rechts)-Kurs, dem er bei der Suche nach dem richtigen Kurs folgte, befindet.

Ein System, das den Zeitverlust beim Abkommen vom Weg minimiert, muß mit zwei Sensoren beidseitig der markierten Linie ausgestattet sein. Befindet sich der Roboter also auf der vorgegebenen Strecke, ist der Impuls der Sensoren gering. Sobald der Roboter von der Linie abweicht, wird der Impuls eines Sensors ansteigen. Somit „weiß" der Roboter sofort, daß und in welche Richtung er sich falsch bewegt hat. Ist der Roboter nach rechts abgewichen, wird der Impuls des linken Sensors stärker und löst damit ein Signal aus, das den Roboter zurück auf die Strecke steuert.

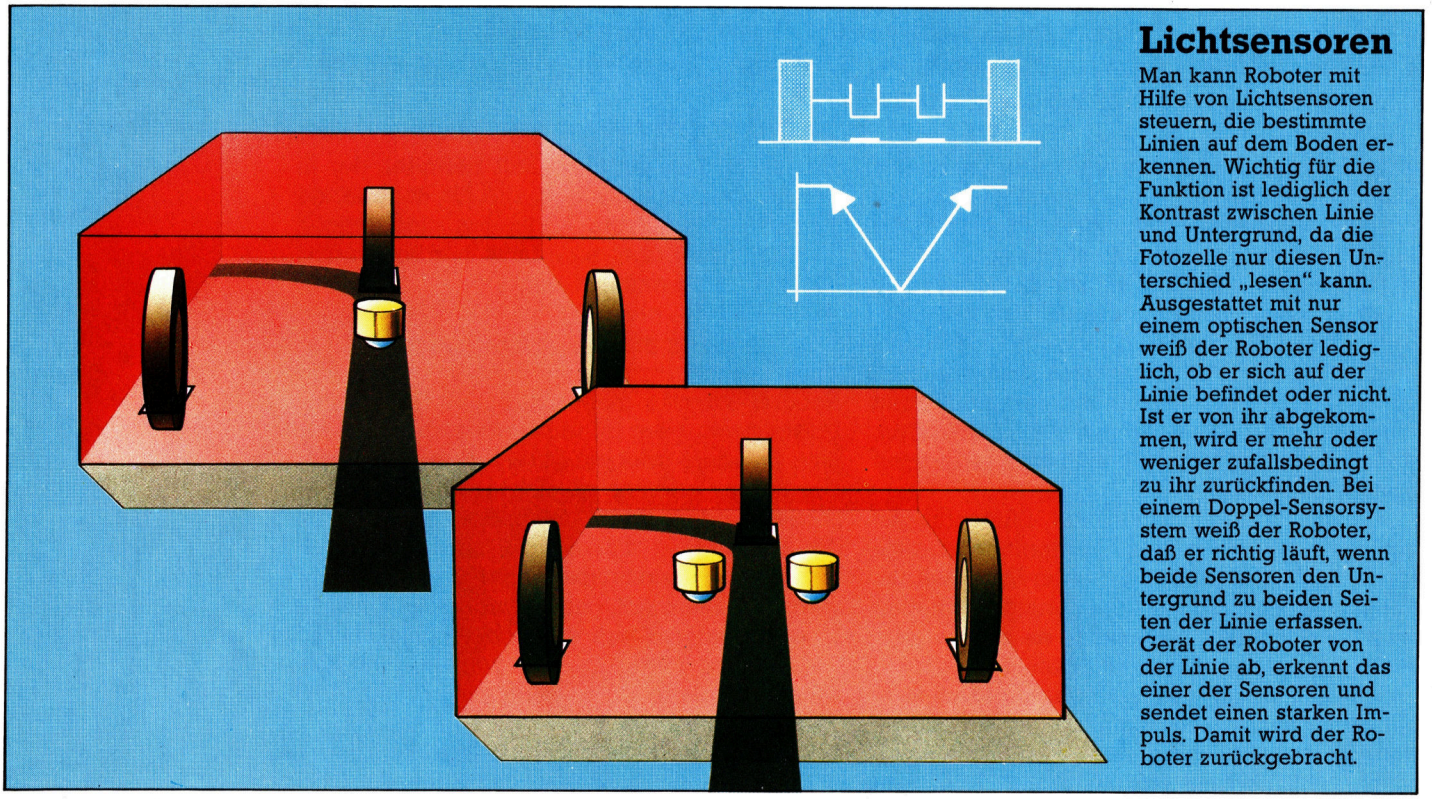
Das System funktioniert immer, gleich ob es sich um eine helle Linie auf dunklem Grund oder den umgekehrten Fall handelt. Wichtig sind allein der Kontrast und das Programm, das dem Roboter vorgibt, was zu tun ist, wenn ein Sensor einen falschen Wert erhält.

Eine andere Art, den Roboter zu steuern, basiert auf einem feinen, im Boden befindlichen Draht, der vom Roboter durch einen Stromimpuls erkannt wird. Mit dem Strom wird ein schwaches Magnetfeld am Draht erzeugt, das vom Sensor gelesen wird. Der dazu erforderliche Sensor ist einfach aufgebaut. Das Magnetfeld erzeugt in einer kleinen Spule eine Minimalspannung, die dort verstärkt werden kann und ähnlich wie Fotozellen reagiert. Industrieroboter werden überwiegend auf diese Art gesteuert, wenngleich auch eine Kontraststeuerung funktionieren würde. Durch Schmutz auf dem Boden könnte diese jedoch gestört werden.

Fernsteuerung

Auch die Steuerung eines Roboters durch einen Menschen ist denkbar. Diese Methode ist naheliegend, wenn Arbeiten von einem Roboter durchzuführen sind, die auch ein Mensch verrichten könnte, die Umstände dies aber verbieten: etwa bei der Bombenentschärfung, beim Umgang mit gefährlichen Chemikalien und radioaktiven Stoffen oder bei der Arbeit in Gebieten, die zu heiß, zu kalt oder generell zu gefährlich sind. Einer der bekanntesten Roboter dieser Art ist der sowjetische Lunokhod 1, der 1970 von der Sonde Luna 16 auf dem Mond abgesetzt wurde. Es handelte sich dabei um einen Roboter auf Rädern, der die Mondoberfläche, von der Erde ferngesteuert, untersuchte.

Die Steuerung eines solchen Roboters unterscheidet sich kaum von der eines ferngesteu-



Lichtsensoren

Man kann Roboter mit Hilfe von Lichtsensoren steuern, die bestimmte Linien auf dem Boden erkennen. Wichtig für die Funktion ist lediglich der Kontrast zwischen Linie und Untergrund, da die Fotozelle nur diesen Unterschied „lesen" kann. Ausgestattet mit nur einem optischen Sensor weiß der Roboter lediglich, ob er sich auf der Linie befindet oder nicht. Ist er von ihr abgekommen, wird er mehr oder weniger zufallsbedingt zu ihr zurückfinden. Bei einem Doppel-Sensorsystem weiß der Roboter, daß er richtig läuft, wenn beide Sensoren den Untergrund zu beiden Seiten der Linie erfassen. Gerät der Roboter von der Linie ab, erkennt das einer der Sensoren und sendet einen starken Impuls. Damit wird der Roboter zurückgebracht.

erten Modellflugzeuges. Das Funksignal kann sowohl analog (also entsprechend der erforderlichen Bewegung des Roboters unterschiedlich stark) als auch digital sein. Beim digitalen Signal werden die Bewegungsbefehle über ein Bitmuster vermittelt. Analoge Signale sind gegenüber den digitalen störanfälliger, da es mehrere Faktoren gibt, die die Signalstärke bei der analogen Übertragung beeinflussen. Das wird deutlich, wenn man beispielsweise versucht, eine entfernte Rundfunkstation zu unterschiedlichen Tageszeiten und Wetterbedingungen zu empfangen. Gleiches gilt für die Robotersteuerung.

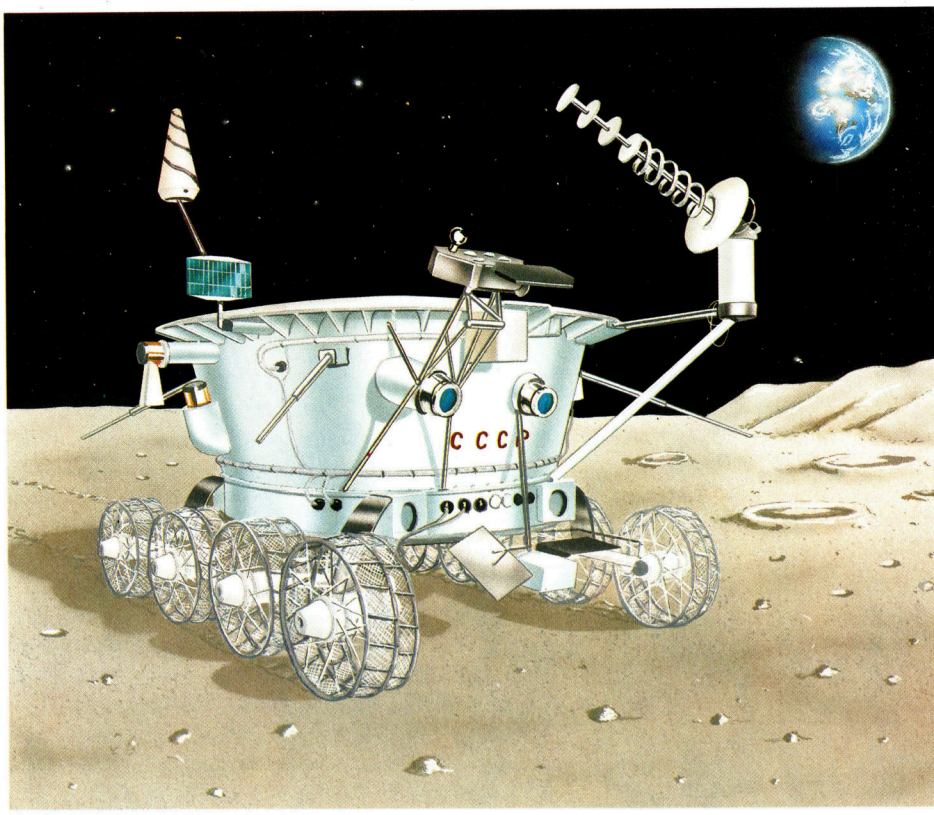
Probleme gibt es natürlich auch bei der digitalen Steuerung, sei es, daß durch Störungen Bits verlorengehen oder daß Bits eingefügt werden, wo sie nicht hingehören. Um das zu vermeiden, wiederholt man die Befehle an den Roboter. Erst wenn der Roboter mehrfach einen gleichlautenden Befehl empfangen hat, führt er ihn aus.

Eine noch weiter entwickelte Technik bedient sich der sogenannten „Schleife", bei der der Roboter das empfangene Signal zur Kontrolle zurückgibt. Man könnte das als Dialog zwischen Sender und Roboter bezeichnen. Ein Beispiel: Der Sender befiehlt „Geh vorwärts", worauf der Roboter zurückfragt „Sagtest du: Geh vorwärts?" und der Sender bestätigt „Ja". Erst jetzt führt der Roboter den Befehl aus. Damit lassen sich verhängnisvolle Fehler, etwa beim Transport radioaktiven Mülls oder beim Durchfahren eines Mondkraters, vermeiden.

Rückkopplungs-Systeme

Befindet sich der den Roboter steuernde Mensch in der Nähe, kann man auf diese Methoden verzichten und ihn über eine direkte Drahtverbindung steuern. Tunlichst wird man mehrere Drähte verwenden, so wie man bei der Fernsteuerung eines Flugzeugmodells ja auch verschiedene Kanäle hat. Beim Roboter allerdings benutzt man mehrere Drähte, um statt serieller parallele Kommunikation führen zu können. Das bedeutet, daß eine Reihe von Bits gleichzeitig über mehrere Drähte ausgesendet wird. Parallele Übertragung erlaubt schnellere Kommunikation mit dem Roboter als serielle. Noch wichtiger ist aber, daß die meisten Computer einen parallelen Port haben, womit ein bequemer Weg zur Steuerung mittels Computertastatur gegeben ist.

Erfolgt die Robotersteuerung durch einen am Computer sitzenden Menschen in Sichtweite, gibt es im Prinzip keinen Unterschied zwischen den Kontrollarten Roboter via Mensch und Roboter via Computer. Das ist einleuchtend, da der Bediener ja wie beim funkferngesteuerten Modellflugzeug Fehler sieht und sofort korrigieren kann. Ist der Roboter aber weiter entfernt (sei es nun auf dem Mond oder auch nur im Nebenzimmer) oder erfolgt die Robotersteuerung über ein Computerprogramm statt über Echtzeit-Eingabe, muß der Roboter schon intelligenter konzipiert sein.



Der Lunokhod 1 der UdSSR wurde 1970 auf dem Mond gelandet, um Informationen über Oberflächenbeschaffenheit und die Atmosphäre zu sammeln. Er war kein echter Roboter, da er von der Erde ferngesteuert wurde. Dank seiner umweltunabhängigen Konstruktion konnte eine größere wissenschaftliche Nutzlast mit dem Raumschiff transportiert werden, als es mit Kosmonauten möglich gewesen wäre, denn deren Lebenserhaltungssysteme wären schwer gewesen. Wie bei allen im Weltraum befindlichen ferngesteuerten Objekten gab es auch bei Lunokhod ein Problem: Zwischen der Sendung vom Mond und dem Empfang des Kontrollsignals von der Erde vergingen drei Sekunden.

Vor allem wird hier eine Rückkopplungsmöglichkeit benötigt. Diese befähigt das System, sein Tun zu korrigieren, indem Bezug zu dem hergestellt wird, was bereits erledigt worden ist und was noch getan werden soll. Ein Beispiel für diesen Referenzrahmen: Soll ein Roboter drei Meter direktgesteuert laufen, wird einfach der Bewegungsbefehl erteilt, das Durchmessen der Entfernung überprüft und nach erfolgtem Laufen der Roboter angehalten. Das ist durch Blickkontakt möglich. Man sieht, welche Strecke er bereits gelaufen ist und wie weit er noch gehen soll, und kann ihn sofort korrigieren.

All dies muß der Roboter allein tun, wenn es keine menschliche Kontrolle gibt. Der einer Bodenmarkierung folgende Roboter erhält seine Rückkopplung durch die Linie auf dem Boden. Und ebenso braucht auch der computergesteuerte Roboter eine Rückkopplung. Diese wird durch einen sogenannten „Shaft Encoder" erzeugt, eine Scheibe, die sich auf den Achsen der Roboter-Räder befindet und mit der man sehr genau messen kann, wie weit sie sich gedreht haben. Hat der Computer also an den Roboter den Befehl geschickt, sich drei Meter vorwärts zu bewegen, beginnt der Roboter mit der Ausführung der Bewegung und registriert gleichzeitig mittels der Signale, die von den Shaft Encodern kommen, wie weit er sich bewegt hat. Der Roboter geht so lange weiter, bis er das Ziel erreicht hat, an dem er halten soll.

Zeit und Bewegung

Das Sortieren eines Datenfeldes in BASIC kann eine sehr zeitintensive Angelegenheit sein, doch läßt sich so die Geschwindigkeit beim nachfolgenden Suchen spezieller Verzeichnisse erheblich erhöhen.

Bis jetzt haben wir den größten Teil des Programmcodes entwickelt, um Eingaben in die Adreßbuch-„Datenbank" vornehmen zu können. Die notwendigen Routinen zum Speichern der Eingaben auf Cassette oder Diskette fehlen jedoch noch. Worauf wir bisher verzichtet haben, war eine vernünftige Routine zur Generierung des bereits früher erwähnten MODFLD$-Feldes.

Das vollständige Programm wird in diesem Teil des Kurses gezeigt. Als erstes werden in den Zeilen 10250 bis 10330 alle Zeichen in Großbuchstaben umgewandelt und dann alle Zeichen des Strings daraufhin untersucht, ob es sich um eine Leerstelle handelt oder nicht. Die Position der letzten Leerstelle wird in der Variablen S abgelegt.

Mit den Zeilen 10400 bis 10420 werden alle Zeichen nacheinander aus dem „Großbuchstaben"-String in VNAM$ übertragen, sofern sie ASCII-Werte größer als 64 haben. Dadurch werden Punkte (ASCII 46), Apostrophe (ASCII 39), Leerstellen (ASCII 32) und andere Satzzeichen eliminiert. Mit den Zeilen 10450 bis 10470 wird dasselbe für die Zeichen nach der letzten Leerstelle durchgeführt, und diese werden in FAMNAM$ übertragen. Wenn N$ nur ein einzelnes Wort enthält, beispielsweise KARIN, erhält die Variable S den Wert 0, und alle Zeichen werden in FAMNAM$ übertragen.

Datensätze sortieren

Die Zeilen 10490 bis 10500 werden gebraucht, um die in dieser Routine verwendeten String-Variablen vor einer erneuten Verwendung wieder auf Null zu setzen. Dies ist ein Punkt, an dem man darauf achten sollte, ob Konstruktionen wie LET X$=X$+Y$ verwendet werden. Vergißt man, die Variablen zu löschen, ist das Resultat eine unerwünschte immer größer werdende Zeichenkette. Beachten Sie deshalb, daß WAHL in der ADDVER-Routine auf 0 gesetzt wird.

Such-Routinen

Nun brauchen wir eine Möglichkeit, die Datei auf Cassette oder Diskette zu speichern. Der einfachste Weg wäre, alle Verzeichnisse in der Reihenfolge, wie sie gerade sind, in die Datei zu schreiben. Der Hauptnachteil dieser Lösung zeigt sich, wenn ein bestimmtes Verzeichnis innerhalb der Datei gesucht wird: Wenn man nicht genau weiß, wo das gesuchte Verzeichnis abgelegt wurde, bleibt nur der Weg, alle Verzeichnisse der gesamten Datei zu durchsuchen, bis der Vergleich mit dem Suchbegriff zutrifft. Unglücklicherweise sind sowohl das Sortieren als auch das Suchen sehr zeitintensive Vorgänge. Die Frage ist, welchem man eine Priorität einräumt. Wir haben festgelegt, daß in dem Adreßbuch öfter nachgeschlagen wird, als neue Verzeichnisse eingegeben (oder vorhandene modifiziert) werden. Deshalb ist sicherzustellen, daß die Datei vor dem Abspeichern immer zuerst sortiert wird.

Mit diesem Hintergedanken wird eine Variable mit dem Namen VMOD zum Setzen eines Flags verwendet. Sie kann einen von zwei möglichen Werten beinhalten: 0 oder 1. Ursprungswert ist 0, um anzuzeigen, daß während der Programmausführung kein Verzeichnis modifiziert wurde. Jeder Arbeitsvorgang, der die Datei in irgendeiner Form verändert – beispielsweise das Hinzufügen eines neuen Verzeichnisses – bewirkt, daß VMOD den Wert 1 erhält. Die Routine ENPROG beispielsweise (zum Speichern der Datei und zum Beenden des Programmes) überprüft VMOD in Zeile 11050. Ist der Wert 0, so ist kein Sortieren und Speichern notwendig, da man davon ausgehen kann, daß die Datei bereits in der entsprechenden Form auf der Cassette bzw. Diskette gespeichert ist. Andere Routinen, wie z. B. die Such-Routine, überprüfen ebenfalls VMOD. Enthält VMOD den Wert 0, so kann der Suchvorgang (oder eine andere Operation) fortgesetzt werden. Beträgt der Wert von VMOD 1, wird die entsprechende Routine zuerst einmal die Sortier-Routine aufrufen. Ist die gesamte Datei neu sortiert worden, wird der Wert von VMOD wieder auf 0 gesetzt.

Die Sortier-Routine mit dem Namen *SRTVER* setzt in Zeile 11320 nach Sortierung aller Verzeichnisse VMOD auf 0. Bevor wir uns jetzt *ENPROG" ansehen, wollen wir uns ein paar Gedanken über *SRTVER* machen. Diese Routine arbeitet nach der sehr einfachen Sortiertechnik „Bubble Sort". Es gibt viele Möglichkeiten zum Sortieren, von denen ein „Bubble Sort" die einfachste und jedoch langsamste ist. Die fortschrittlicheren Sortier-Routinen sind meist erheblich schwerer zu verstehen als die in diesem Programm verwendete. Die Notwendigkeit einer besseren Sortier-Rou-

```
  10 REM "HAUPTPROGRAMM"
  20 REM *INITIL*
  30 GOSUB 1000
  40 REM *BGRUES*
  50 GOSUB 3000
  60 REM *AUWAHL*
  70 GOSUB 3500
  80 REM *AUSFUH*
  90 GOSUB 4000
 100 IF WAHL<> 9 THEN 60
 110 END
1000 REM *INITIL* UNTERROUTINE
1010 GOSUB 1100:REM *CREARR*
     (DEFINIERE BEREICHE)
     UNTERROUTINE
1020 GOSUB 1400:REM *LSINDT*
     (LESE DATEI EIN) UNTER-
     ROUTINE
1030 GOSUB 1600:REM *SETFLG*
     (SETZE FLAGS) UNTER-
     ROUTINE
1040 REM
1050 REM
1060 REM
1070 REM
1080 REM
1090 RETURN
1100 REM *CREARR* (DEFINIERE
     BEREICHE) UNTERROUTINE
1110 DIM NAMFLD$(50)
1120 DIM MODFLD$(50)
1130 DIM STRFLD$(50)
1140 DIM STDFLD$(50)
1150 DIM STAFLD$(50)
1160 DIM TELFLD$(50)
1170 DIM INDFLD$(50)
1180 REM
1190 REM
1200 REM
1210 LET GROSS=0
1220 LET VMOD=0
1230 LET SVED=0
1240 LET CURR=0
1250 REM
1260 REM
1270 REM
1280 REM
1290 REM
1300 RETURN
1400 REM *LSINDT* UNTER-
     ROUTINE
1410 OPEN "I", #1, "ADBK. DAT"
1420 INPUT #1, TEST$
1430 IF TEST$ = "@ERST" THEN
     GOTO 1540:REM
     SCHLIESSEN UND RETURN
1440 LET NAMFLD$(1)=TEST$
1450 INPUT #1, MODFLD$(1),
     STRFLD$(1), STDFLD$(1),
     STAFLD$(1), TELFLD$(1)
1460 INPUT #1, INDFLD$(1)
1470 LET GROSS=2
1480 FOR L=2 TO 50
1490 INPUT #1, NAMFLD$(L),
     MODFLD$(L), STRFLD$(L),
     STDFLD$(L), STAFLD$(L)
1500 INPUT #1, TELFLD$(L),
     INDFLD$(L)
1510 LET GROSS=GROSS+1
1520 IF EOF(1)=-1 THEN LET L=50
1530 NEXT L
1540 CLOSE #1
1550 RETURN
1600 REM *SETFLG* UNTER-
     ROUTINE
1610 REM SETZT FLAGS NACH
     *LSINDT*
1620 REM
1630 REM
1640 IF TEST$ = "@ERST" THEN
     LET GROSS=1
1650 REM
1660 REM
1670 REM
1680 REM
1690 RETURN
3000 REM *BGRUES* UNTER-
     ROUTINE
3010 PRINT CHR$(12):REM
     LOESCHT BILDSCHIRM
3020 PRINT
3030 PRINT
3040 PRINT
3050 PRINT
3060 PRINT TAB(13);"*WILL-
     KOMMEN ZUM*"
3070 PRINT TAB(11);"*ADRESS-
     BUCH-PROGRAMM*"
3080 PRINT TAB(8);"*DES
     COMPUTER-KURSES*"
3090 PRINT
3100 PRINT TAB(1);"(DRUECKE DIE
     LEERTASTE, UM FORTZU-
     FAHREN)"
3110 FOR L=1 TO 1
3120 IF INKEY$ <>" " THEN L=0
3130 NEXT L
3140 PRINT CHR$(12)
3150 RETURN
3500 REM *AUWAHL* UNTER-
     ROUTINE
3510 REM
3520 IF TEST$="@ERST" THEN
     GOSUB 3860:REM *ERSTLA*
     UNTERROUTINE
3530 IF TEST$="@ERST" THEN
     RETURN
3540 REM "CHMENU"
3550 PRINT CHR$(12)
3560 PRINT "WOLLEN SIE"
3570 PRINT
3580 PRINT
3590 PRINT
3600 PRINT "1. VERZEICHNIS
     DURCH NAMEN SUCHEN"
3610 PRINT "2. NAMEN DURCH
     TEIL EINES NAMENS
     SUCHEN"
3620 PRINT "3. VERZEICHNISSE
     NACH STADTANGABEN
     SUCHEN"
3630 PRINT "4. VERZEICHNIS-
     LISTE DURCH INITIALEN"
3640 PRINT "5. LISTE ALLER
     VERZEICHNISSE"
3650 PRINT "6. HINZUFUEGEN
     EINER ADRESSE"
3660 PRINT "7. AENDERN EINER
     ADRESSE"
3670 PRINT "8. LOESCHEN EINER
     ADRESSE"
3680 PRINT "9. PROGRAMM
     BEENDEN UND DATEN
     SPEICHERN"
3690 PRINT
3700 PRINT
3710 REM "INWAHL"
3720 REM
3730 LET L=0
3740 LET I=0
3750 FOR L=1 TO 1
3760 PRINT "WAEHLEN SIE (1-9)"
3770 FOR I=1 TO 1
3780 LET A$=INKEY$
3790 IF A$="" THEN I=0
3800 NEXT I
3810 LET WAHL=VAL(A$)
3820 IF WAHL < 1 THEN L=0
3830 IF WAHL > 9 THEN L=0
3840 NEXT L
3850 RETURN
3860 REM *ERSTLA* UNTERROU-
     TINE (STELLT MELDUNG
     DAR)
3870 LET WAHL=6
3880 PRINT CHR$(12): REM
     LOESCHT BILDSCHIRM
3890 PRINT
3900 PRINT TAB(8); "ES SIND
     KEINE VERZEICHNISSE"
3910 PRINT TAB(8); "IN DER DATEI
     SIE MUESSEN"
3920 PRINT TAB (8); "MIT DER
     EINGABE EINES VERZEICH-
     NISSES BEGINNEN"
3930 PRINT
3940 PRINT TAB (5); "(LEERTASTE
     UM FORTZUFAHREN)"
3950 FOR B=1 TO 1
3960 IF INKEY$<>" " THEN B=0
3970 NEXT B
3980 PRINT CHR$(12):REM
     LOESCHT BILDSCHIRM
3990 RETURN
4000 REM *AUSFUH* UNTER-
     ROUTINE
4010 REM
4020 IF WAHL = 6 THEN GOSUB
     10000
4030 REM
4040 REM 1 IST *FNDVER*
4050 REM 2 IST *FNDNMS*
4060 REM 3 IST *FNDSTD*
4070 REM 4 IST *FNDINT*
4080 REM 5 IST *LSTVER*
4090 IF WAHL = 6 THEN GOSUB
     10000
4100 REM 7 IST *MODVER*
4110 REM 8 IST *LOEVER*
4120 IF WAHL = 9 THEN GOSUB
     11000
4130 REM
4140 RETURN
10000 REM *ADDVER* UNTER-
      ROUTINE
10010 PRINT CHR$(12):REM
      LOESCHE BILDSCHIRM
10020 INPUT "GIB NAMEN
      EIN";NAMFLD$(GROSS)
10030 INPUT "GIB STRASSE
      EIN";STRFLD$ (GROSS)
10040 INPUT "GIB STADT EIN";
      STDFLD$(GROSS)
10050 INPUT "GIB STAAT EIN";
      STAFLD$ (GROSS)
10060 INPUT "GIB TELEFONNUM-
      MER EIN"; TELFLD$ (GROSS)
10070 LET VMOD=1: REM "VER-
      ZEICHNIS MODIFIZIERT"
      FLAG SET
10080 LET INDFLD$ (GROSS) =
      STR$ (GROSS)
10090 LET TEST$=""
10100 GOSUB 10200: REM
      *MODNAM*
10110 LET WAHL=0
10120 LET GROSS=GROSS+1
10130 REM
10140 REM
10150 RETURN
10200 REM *MODNAME* ROUTINE
10210 REM WANDELT INHALT VON
      NAMFLD$ IN GROSSBUCH-
      STABEN UM,
10220 REM ENTFERNT UNNOETIGE
      ZEICHEN UND SPEICHERT IN
      DER REIHENFOLGE:
10230 REM FAMILIENNAME +
      LEERSTELLE + VORNAMEN
      IN MODFLD$
10240 REM
10250 LET N$=NAMFLD$(GROSS)
10260 FOR L=1 TO LEN (N$)
10270 LET TEMP$=MID$ (N$, L, 1)
10280 LET T=ASC(TEMP$)
10290 IF T >=97 THEN T=T-32
10300 LET TEMP$=CHR$(T)
10310 LET P$=P$+TEMP$
10320 NEXT L
10330 LET N$=P$
10340 REM LOKALISIERT LETZTE
      LEERSTELLE
10350 FOR L=1 TO LEN(N$)
10360 IF MID$(N$,L,1)=" " THEN
      S=L
10370 NEXT L
10380 REM ENTFERNT UNNOETIGE
      ZEICHEN UND
10390 REM SPEICHERT VORNAMEN
      IN VNAM$
10400 FOR L=1 TO S-1
10410 IF ASC(MID$(N$, L, 1))>64
      THEN VNAM$=VNAM$+MID$
      (N$, L, 1)
10420 NEXT L
10430 REM ENTFERNT UNNOETIGE
      ZEICHEN UND
10440 REM SPEICHERT FAMILIEN-
      NAMEN IN FAMNAM$
10450 FOR L=S+1 TO LEN(N$)
10460 IF ASC(MID$(N$,L,1))>64
      THEN FAMNAM$=
      FAMNAM$+MID$(N$,L,1)
10470 NEXT L
10480 LET MODFLD$(GROSS)=
      FAMNAM$+" "+VNAM$
10490 LET P$="":LET N$="":LET
      FAMNAM$="": LET
      VNAM$=""
10500 LET P$="":LET N$="":LET
      FAMNAM$="":LET
      VNAM$=""
10510 RETURN
11000 REM *ENPROG* UNTER-
      ROUTINE
11010 REM SORTIERT UND
      SPEICHERT
11020 REM DIE DATEI WENN
      IRGENDEIN VERZEICHNIS
11030 REM MODIFIZIERT WURDE
      (VMOD=1)
11040 REM
11050 IF VMOD=0 THEN RETURN
11060 REM
11070 GOSUB 11200: REM
      *SRTVER*
11080 REM
11090 GOSUB 12000: REM
      *SPEVER*
11100 RETURN
11200 REM *SRTVER* UNTER-
      ROUTINE
11210 REM SORTIERT ALLE VER-
      ZEICHNISSE NACH MODFLD$
11220 REM IN ALPHABETISCHER
      REIHENFOLGE UND AKTUALI-
      SIERT INDFLD$
11230 REM
11240 REM
11250 LET S=0
11260 FOR L=1 TO GROSS-2
11270 IF MODFLD$(L)>MODFLD$
      (L+1) THEN GOSUB 11350
11280 NEXT L
11290 IF S=1 THEN 11250
11300 REM
11310 REM
11320 LET VMOD=0: LOESCHT
      "VERZEICHNIS MODIFIZIERT"
      FLAG
11330 REM
11340 RETURN
11350 REM *VTAVER* UNTER-
      ROUTINE
11360 LET TNAMFD$ =
      NAMFLD$(L)
11370 LET TMODFD$ =
      MODFLD$(L)
11380 LET TSTRFD$ = STRFLD$(L)
11390 LET TSTDFD$ = STDFLD$(L)
11400 LET TSTAFD$ = STAFLD$(L)
11410 LET TTELFD$ = TELFLD$(L)
11420 REM
11430 LET NAMFLD$(L) =
      NAMFLD$(L+1)
11440 LET MODFLD$(L) =
      MODFLD$(L+1)
11450 LET STRFLD$(L) =
      STRFLD$(L+1)
11460 LET STDFLD$(L) =
      STDFLD$(L+1)
11470 LET STAFLD$(L) =
      STAFLD$(L+1)
11480 LET TELFLD$(L) =
      TELFLD$(L+1)
11490 LET INDFLD$(L) = STR$(L)
11500 REM
11510 LET NAMFLD$(L+1) =
      TNAMFD$
11520 LET MODFLD$(L+1) =
      TMODFD$
11530 LET STRFLD$(L+1) =
      TSTRFD$
11540 LET STDFLD$(L+1) =
      TSTDFD$
11550 LET STAFLD$(L+1) =
      TSTAFD$
11560 LET TELFLD$(L+1) =
      TTELFD$
11570 LET INDFLD$(L+1) =
      STR$(L+1)
11580 LET S=1
11590 REM
11600 RETURN
12000 REM *SPEVER* UNTER-
      ROUTINE
12010 REM
12020 REM
12030 OPEN "0", #1, "ADBK. DAT"
12040 REM
12050 FOR L=1 TO GROSS-1
12060 PRINT #1, NAMFLD$(L);",";
      MODFLD$(L);",";STRFLD$
      (L);",";STDFLD$(L)
12070 PRINT #1, STAFLD$(L);",";
      TELFLD$(L);","; INDFLD$(L)
12080 NEXT L
12090 REM
12100 REM
12110 REM
12120 REM
12130 CLOSE #1
12140 REM
12150 RETURN
12500 REM *DTGROS* UNTER-
      ROUTINE
12510 IF NAMFLD$(L)=""THEN
      LET L=50
12520 IF NAMFLD$(L)="" THEN
      RETURN
12530 LET GROSS=GROSS+1
12540 REM
12550 REM
12560 RETURN
```

BASIC-Dialekte

Bevor Sie das Adreßbuch-Programm starten, müssen Sie die Namensfeld-Datei auf der Cassette anlegen. Das folgende Programm führt dies durch.

```
10 REM PROGRAMM ZUM ANLEGEN
   VON NFLD AUF CASSETTE
20 DIM Z$(1,30)
30 LET Z$(1)="@ERST"
40 SAVE "NFLD" DATA Z$( )
50 STOP
```

Wenn das Programm stoppt, spulen Sie das Band zurück und tippen VERIFY „NFLD" DATA Z$(), um das Abspeichern zu überprüfen.

Im folgenden sehen Sie die Spectrum-Version von Teilen des Hauptprogramms. Beachten Sie, daß Zeilennummern größer als 9999 vom Spectrum nicht verarbeitet werden können. Wir haben sie hier beibehalten, um Ihnen einen leichteren Vergleich zum Hauptlisting zu ermöglichen. Ändern Sie die Zeilennummern entsprechend, und denken Sie auch daran, die GOSUBs anzupassen.

```
1100 REM *CREARR*
1110 DIM N$ (50,30)
1120 DIM M$ (50,30)
1130 DIM S$(50,30)
1140 DIM T$ (50,15)
1150 DIM C$ (50,15)
1160 DIM R$(50,15)
1170 DIM X$(50,30)
1180 DIM B$ (30)
1190 DIM Z$(30)
1250 LET Z$="@ERST"
 •
1400 REM *LSINDT* UR
1410 LOAD „NFLD" DATA N$( )
1420 IF N$(1)=Z$ THEN LET
     Q$=Z$:RETURN
1430 LOAD "MFLD" DATA M$( )
1440 LOAD "SFLD" DATA S$( )
1450 LOAD "TFLD" DATA T$( )
1460 LOAD "CFLD" DATA C$( )
1470 LOAD "TELFLD" DATA R$( )
1480 LOAD "INDFLD" DATA X$( )
1490 REM *DTGROS"
1500 GOSUB 12500
 •
1540 RETURN
 •
1640 IF Q$=Z$ THEN LET
     GROSS=1
 •
3520 IF Q$=Z$ THEN GOSUB
     3860:RETURN
 •
3810 LET WAHL=CODE A$-48
 •
10090 LET QS=""
 •
10200 REM *MODNAM* UR
 •
10250 LET D$=N$(GROSS):LET
     P$=""
10260 FOR L=1 TO LEN (D$)
10270 LET A$=D$(L)
10280 LET T=CODE A$
10290 IF T>=97 THEN LET T=T-32
10300 LET A$=CHR$ T
10310 LET P$=P$+A$
10320 NEXT L
10330 LET D$=P$:LET P$="":LET
     A$="": LET T=LEN (D$)
10340 REM LOKALISIERE LETZTE
     LEERSTELLE
10350 FOR L=1 TO T
10360 IF D$(L)=" " THEN LET
     S=L:LET L=T
10370 NEXT L
10380 REM ENTFERNE UNNOETIGE
     ZEICHEN
```

tine hängt auch wesentlich von der Anzahl der zu sortierenden Datensätze ab. Die „Zeitkomplexität" eines „Bubble Sort" wie in diesem Programm ist n^2. Mit anderen Worten erhöht sich die zum Sortieren der Daten benötigte Zeit im Quadrat zur Gesamtanzahl der Datensätze. Aufgrund dieser Rechenbasis ergibt sich bei 1000 Datensätzen eine Sortierzeit von über 16 Minuten.

Begrenztes Verzeichnis

Dieses Programm ist jedoch auf maximal 50 Verzeichnisse begrenzt, so daß es mit dem „Bubble Sort" kaum ernsthafte Zeitprobleme geben dürfte. Zu einem späteren Zeitpunkt wird die Technik der dynamischen Dateien aufgezeigt, mit denen sich fast unbegrenzt viele Verzeichnisse verwalten lassen.

Die sortierten Daten sind die Zeichenketten in MODFLD$(L) und MODFLD$(L+1). Verzeichnisse werden in ihrer Reihenfolge nur dann vertauscht, wenn MODFLD$(L) größer ist als MODFLD$(L+1), und das Index-Feld (welches zur Zeit nicht benutzt wird) wird in den Zeilen 11490 bis 11570 aktualisiert. Jedes Mal, wenn zwei Verzeichnisse vertauscht werden, wird die Variable S auf den Wert 1 gesetzt. Wenn die Sortier-Routine Zeile 11290 erreicht, überprüft sie den Wert der Variablen S. Ist der Wert 1, so wird die Routine erneut ausgeführt, und alle Verzeichnisse werden nochmals miteinander verglichen. Sind alle Verzeichnisse in der richtigen Reihenfolge, so beinhaltet S den Wert 0, und die Routine setzt den Wert VMOD auf 0. Danach folgt der Rücksprung zum Hauptprogramm.

Modifizierte Eintragungen

Die ENPROG-Routine beginnt ab Zeile 11000. Als erstes wird überprüft, ob ein Verzeichnis während des bisherigen Programmlaufs modifiziert wurde (Zeile 11050 : IF VMOD=0 THEN RETURN). Wenn an der Datei keine Modifikationen vorgenommen wurden, so braucht auch nicht neu abgespeichert zu werden. Die Routine kehrt zum Hauptprogramm zurück. So gelangt das Programm in Zeile 100, wo der Wert der Variablen WAHL überprüft wird. Beträgt der Wert von WAHL 9 (= Datei sichern und Programm beenden), springt das Hauptprogramm einfach zu Zeile 110 und führt den END-Befehl aus.

Wenn das Programm bei seiner Überprüfung in Zeile 11050 feststellt, daß VMOD den Wert 1 hat, bedeutet das, daß ein oder mehrere Verzeichnisse in irgendeiner Form modifiziert wurden und sich somit eventuell die Reihenfolge verändert hat. In diesem Fall ruft die *ENPROG*-Routine die Sortier-Routine auf (Zeile 11070), und, nachdem alle Verzeichnisse neu sortiert sind, speichert dann die Datei auf Cassette oder Diskette.

PRINT # und INPUT

Die Speicher-Routine (*SPEVER*) wird in Zeile 11090 aufgerufen. Die Routine selbst beginnt ab Zeile 12000. Da *SPEVER* im Hauptprogramm-Listing in Microsoft-BASIC geschrieben ist, sollten Sie sich vor dem Abtippen im Kasten „BASIC-Dialekte" informieren. In Zeile 12030 wird die ADBK. DAT Daten-Datei

```
10390 REM SPEICHERE VORNAMEN
      IN P$
10400 FOR L=1 TO S-1
10410 IF CODE (D$(L))›64 THEN LET
      P$=P$+D$(L)
10420 NEXT L
10430 REM ENTFERNE UNNOETIGE
      ZEICHEN
10440 REM SPEICHERE FAMILIEN-
      NAMEN IN A$
10450 FOR L=S+1 TO LEN (D$)
10460 IF CODE (D$(L))›64 THEN LET
      A$=A$+D$(L)
10470 NEXT L
10480 LET M$(GROSS)=A$+" "+P$
10490 LET P$=" ":LET A$""
10510 RETURN
```

ANMERKUNG: Wegen der Form des String-Handlings des Spectrum teilt die hier gezeigte Routine den Namen an der ersten und nicht an der letzten Leerstelle

```
12000 REM *SPEVER* UR
12030 SAVE "NFLD" DATA N$()
12040 SAVE "MFLD" DATA M$()
12050 SAVE "SFLD" DATA S$()
12060 SAVE "TFLD" DATA T$()
12070 SAVE "CFLD" DATA C$()
12080 SAVE "TELFLD" DATA R$()
12090 SAVE "INDFLD" DATA X$()
12150 RETURN
12500 REM *DTGROS* UR
12510 LET GROSS=50
12520 FOR L=1 TO 50
12530 IF N$(L)=B$ THEN LET
      GROSS=L:LET L=50
12540 NEXT L
12560 RETURN
```

ZX 81

Verwenden Sie dieselben Änderungen wie bei der Spectrum-Version mit folgenden Abweichungen (auch hier müssen Sie die Zeilennummern größer als 9999 entsprechend ändern). Fügen Sie Zeile 1255 LET N$(1)=Z$ hinzu. Löschen Sie die Zeilen 1410 bis 1540 sowie 12030 bis 12140 und fügen Sie folgende Zeilen ein:

```
1410 RETURN
12010 PRINT "LEGEN SIE DIE
      CASSETTE EIN, DRUECKEN
      SIE PLAY UND RECORD, UND
      DRUECKEN SIE NEWLINE"
12020 INPUT W$
12030 SAVE "ADDBK"
```

Wenn das Programm einmal gespeichert wurde, starten Sie es mit GOTO 40, nie mit RUN.

EOF

Auf dem Commodore 64 und VC 20 ersetzen Sie Zeile 1520 durch:

```
1520 IF ST AND 64 THEN LET L=50
```

Auf dem Dragon 32 löschen Sie Zeile 1520 und ersetzen sie durch:

```
1485 IF EOF(-1) THEN GOTO 1510
```

Auf dem Acorn B ersetzen Sie sie durch:

```
1520 IF EOF# X THEN LET L=50
```

Das X ist eine numerische Variable, die in der OPENOUT-Anweisung verwendet wird.

geöffnet und ihr die Kanalnummer #1 zugeordnet. In Zeile 12050 wird die Grenze für die Schleife gesetzt, die alle Verzeichnisse der Datei durchzählt. Die obere Grenze ist GROSS-1, nicht GROSS, da der Wert der Variablen GROSS immer um eins größer ist als die Zahl der gültigen Verzeichnisse in der Datei. So wird beim Hinzufügen eines neuen Verzeichnisses kein bereits vorhandenes versehentlich überschrieben.

Das Format der Zeilen 12060 und 12070 bedarf einer näheren Betrachtung. Jedes Feld ist durch ein „ , " getrennt, das auch zur Datei gesendet wird. Dieses Komma wird von den meisten BASIC-Versionen benötigt, da INPUT# und PRINT# in der gleichen Weise arbeiten wie die normalen Anweisungen INPUT und PRINT. Stellen Sie sich die Anweisung INPUT X, Y, Z vor. Dadurch wird z. B. eine Eingabe wie 10, 12, 15 RETURN über die Tastatur erwartet, die die Werte 10, 12 und 15 den Variablen X, Y und Z zuordnen würde. Ohne die Kommata wäre die INPUT-Anweisung nicht in der Lage zu wissen, wo die einzelnen Daten enden. Somit würden alle Daten der ersten Variablen zugeordnet. Ähnlich verhält es sich bei den meisten BASIC-Versionen mit der INPUT#-Anweisung. Auch sie „wüßte" nicht, wo die einzelnen Felder und Verzeichnisse enden, und würde daher versuchen, jede String-Variable mit so vielen Daten wie möglich zu füllen. Da bei den meisten BASIC-Versionen eine String-Variable bis zu 255 Zeichen beinhalten kann, wären sehr schnell alle Daten der Datei zugeordnet, lange bevor die FOR L=1 TO GROSS-1 Schleife beendet worden wäre. Dies würde zu einer INPUT-PASS-END-Fehlermeldung führen. Diese Fehlermeldung gibt an, daß noch eine INPUT-Anweisung erfolgt ist, nachdem alle Daten bearbeitet wurden. Und die String-Variablen (wie beispielsweise NAMFLD$(x)) würden viel mehr Daten enthalten, als sie sollten.

Wenn alle Verzeichnisse in der Datei gespeichert sind (von L=1 TO GROSS-1), kehrt *SPEVER* zu Zeile 90 des Hauptprogramms zurück. In Zeile 100 wird der Wert von WAHL überprüft, um festzustellen, ob die letzte Operation *ENPROG* war oder nicht. Wenn der Wert 9 ist (sichern und beenden), fährt das Programm mit der Ausführung der END-Anweisung in Zeile 110 fort. Hatte Wahl einen anderen Wert, so wird der Programmlauf mit der *AUWAHL*-Routine fortgesetzt und der Anwender kann eine neue Wahl treffen.

Ende der Datei

Abschließend wollen wir noch einen Blick auf die bei Zeile 12500 beginnende *DTGROSS*-Routine werfen. Sie dient als eine mögliche Alternative für die Anweisung in Zeile 1510. Ihre Verwendung hängt davon ab, ob eine Funktion "End OF File" (Ende der Datei) vorhanden ist: IF EOF(1)=-1 THEN LET L=50. Alle BASIC-Versionen haben eine Möglichkeit anzuzeigen, wann das Ende einer Datei erreicht wurde. Dies kann entweder eine spezielle Funktion wie EOF(x) sein oder eine PEEK-Anweisung für eine bestimmte Speicheradresse. Die *DTGROSS*-Routine ab Zeile 12500 ist eine Alternative, falls eine EOF-Funktion nicht verfügbar ist. In einem solchen Fall müßte Zeile 1510 gegen GOSUB 12500 ausgetauscht werden.

Mit Zange und Zinn

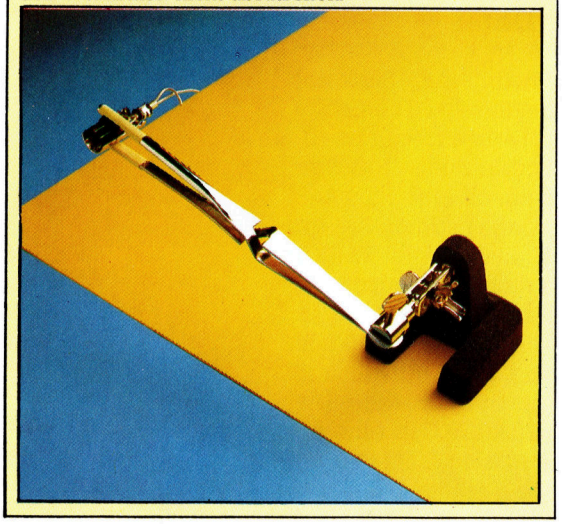
Die Montage und Anpassung von Zusatzgeräten ist beim Computer mit relativ großem Aufwand verbunden. Wie Sie hier Zeit und Geld sparen können, erfahren Sie in diesem Kursus.

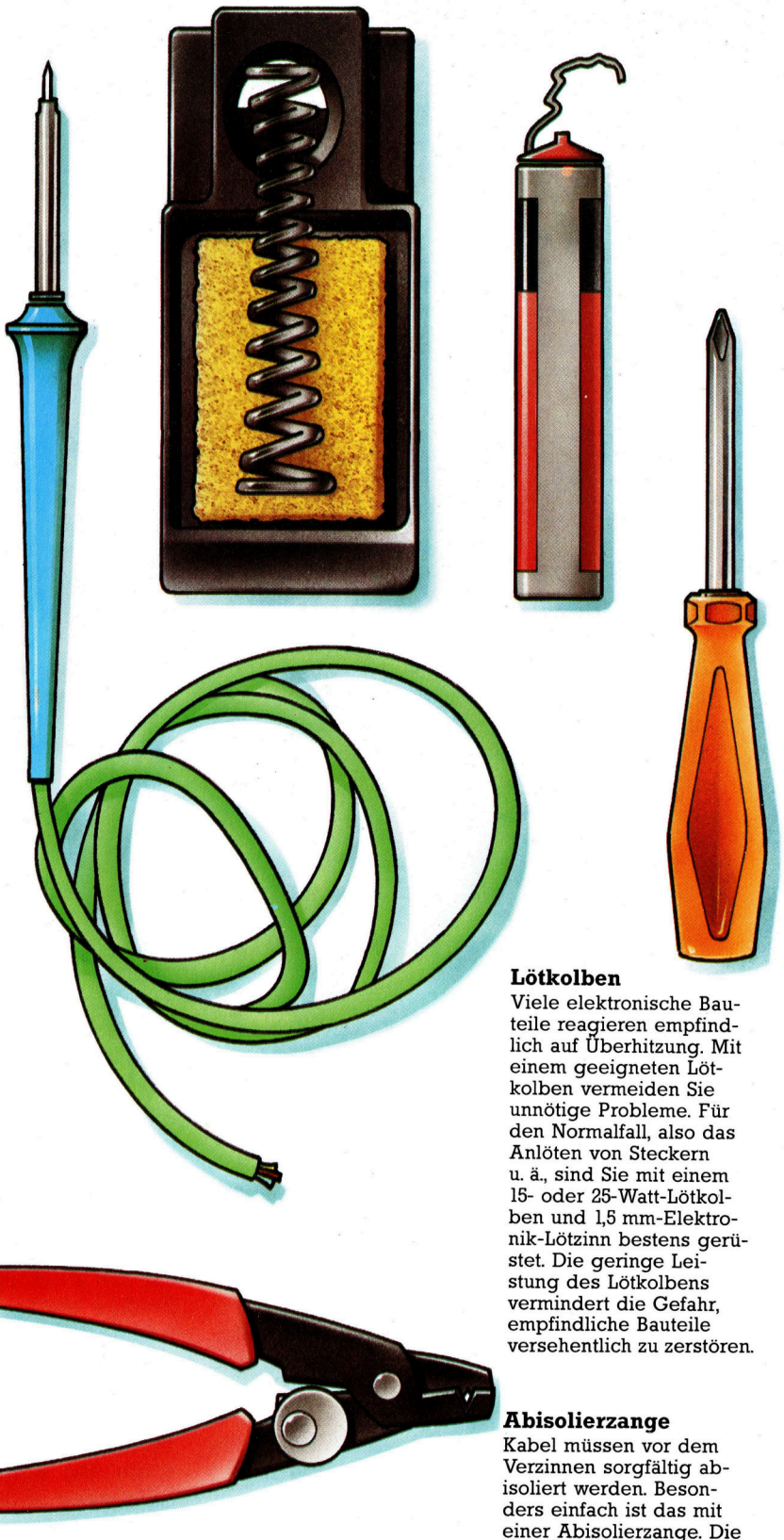
Voraussetzung für elektronische Basteleien ist die genaue Kenntnis des Lötvorganges. Die Verbindung zweier Metallteile durch eine weiche Legierung mit niedrigem Schmelzpunkt nennt man „Weichlöten". In der Elektrotechnik wird als Verbindungsmittel meist Zinn eingesetzt. Die Einzelteile werden auf eine Temperatur oberhalb des Zinn-Schmelzpunktes (280° C) gebracht und zuerst mit dem Lot überzogen. Dann brauchen sie nur noch zusammengehalten werden – beim Abkühlen verfestigt sich das vorher flüssige Zinn.

Sie sollten beim Löten nie mit dem Kolben das heiße Zinn auf den Verbindungspunkt tupfen: Dabei kühlt es schlagartig ab, ohne einen verläßlichen Kontakt herzustellen. Diese „kalten Lötstellen" sind aber leicht zu vermeiden: Zuerst die Kontaktflächen (Drahtenden etc.) erhitzen, dann den Lötkolben wegnehmen und das Lot direkt an der Verbindungsstelle abschmelzen lassen – fertig.

Die dritte Hand

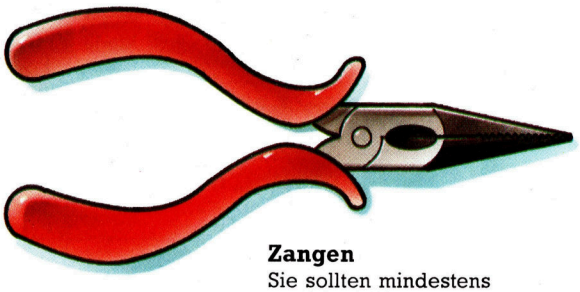
Die Miniaturisierung der Computer schreitet voran – das macht den Umgang mit dem Zubehör gelegentlich zur reinsten Uhrmacher-Arbeit. Preiswerte kleine Klemmen und Pinzetten erleichtern die oft knifflige Arbeit. Klebestreifen – mit der Haftseite nach außen – lassen auch in schwierigen Fällen das Werkstück nicht abrutschen.



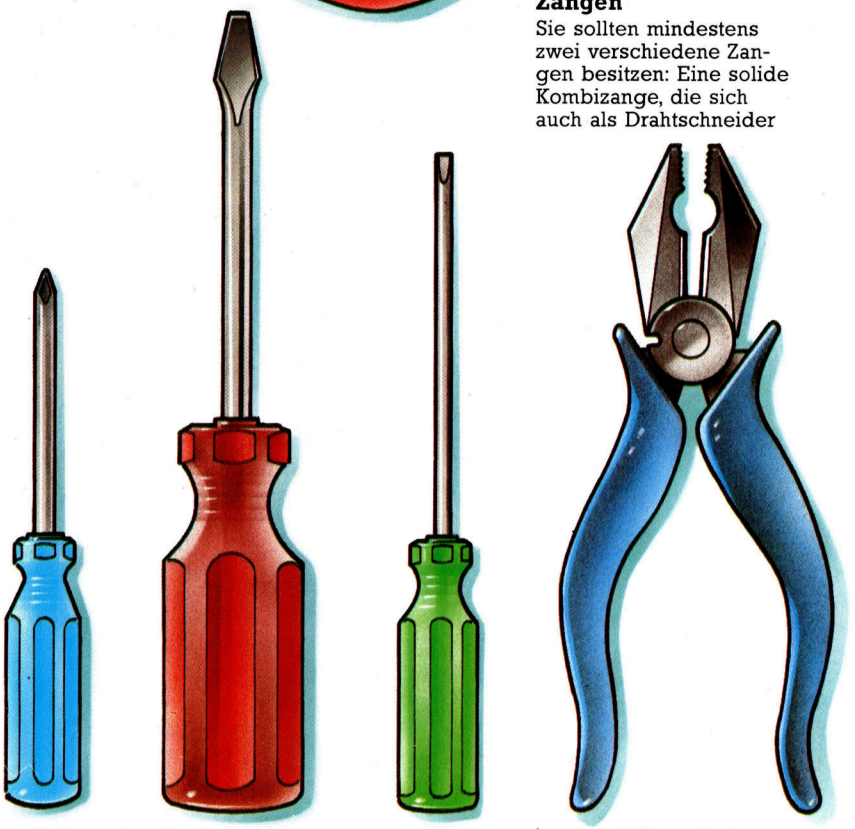


Lötkolben
Viele elektronische Bauteile reagieren empfindlich auf Überhitzung. Mit einem geeigneten Lötkolben vermeiden Sie unnötige Probleme. Für den Normalfall, also das Anlöten von Steckern u. ä., sind Sie mit einem 15- oder 25-Watt-Lötkolben und 1,5 mm-Elektronik-Lötzinn bestens gerüstet. Die geringe Leistung des Lötkolbens vermindert die Gefahr, empfindliche Bauteile versehentlich zu zerstören.

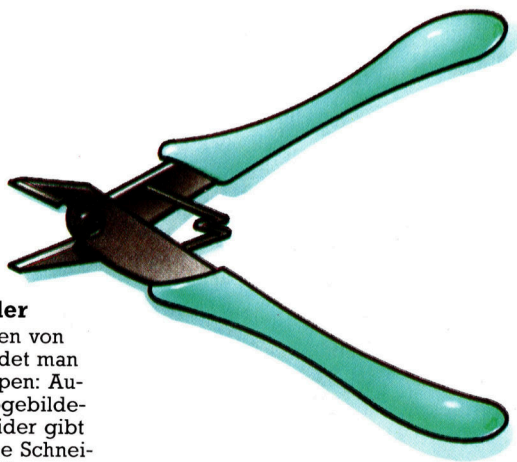
Abisolierzange
Kabel müssen vor dem Verzinnen sorgfältig abisoliert werden. Besonders einfach ist das mit einer Abisolierzange. Die Klinge wird auf die Mantelstärke des Kabels eingestellt. Dadurch geht man sicher, nur den Mantel einzuschneiden – der Draht bleibt intakt.



Zangen
Sie sollten mindestens zwei verschiedene Zangen besitzen: Eine solide Kombizange, die sich auch als Drahtschneider einsetzen läßt, und eine der viel feineren Spitzzangen, mit denen der Zugriff auch bei kleinen Elektronik-Teilchen keinen Schaden anrichtet.

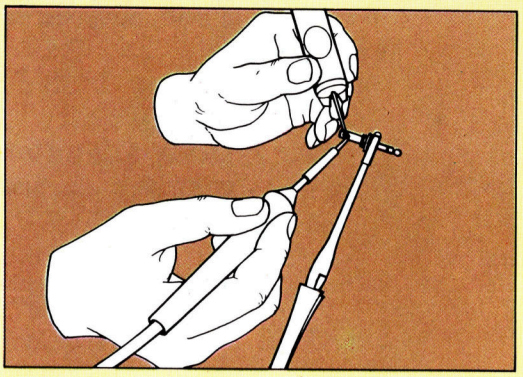


Schraubendreher
Schraubenzieher – die beim Fachmann Schraubendreher heißen – gibt es in zwei unterschiedlichen Ausführungen: Als geraden oder als Kreuzschlitz-Schraubendreher. Bei den Kreuzschlitz-Schraubendrehern gibt es außerdem noch zwei verschiedene Normen, was bei kleinen Größen zum Glück kaum ins Gewicht fällt. Sie sollten sowohl gerade als auch Kreuzschlitzschraubendreher in verschiedenen Breiten zur Hand haben.



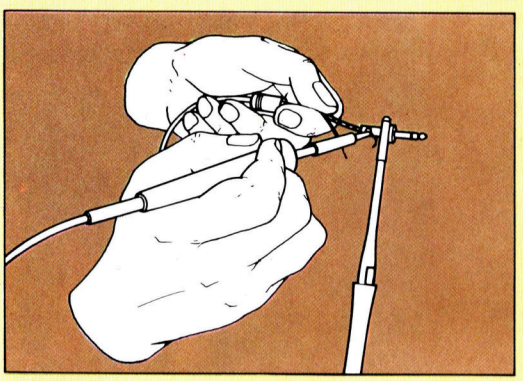
Seitenschneider
Zum Abschneiden von Drähten verwendet man zwei Zangen-Typen: Außer dem hier abgebildeten Seitenschneider gibt es noch spezielle Schneider, bei denen die Schneidklingen im Winkel von 90 Grad zu den Handgriffen stehen.

Stecker anlöten

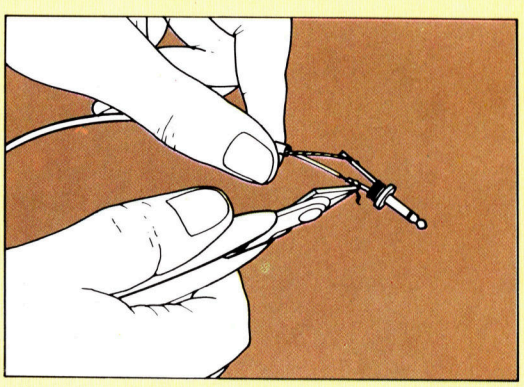


Isolierung entfernen und verzinnen
Vor dem Löten muß zuerst einmal der Kabelmantel entfernt werden. Seien Sie dabei ruhig etwas großzügiger, der Kabelüberschuß kann später abgeschnitten werden. Das Kabel wird nun festgeklemmt und erhitzt. Lötzinn am Draht abschmelzen lassen und mit der Lötkolbenspitze über die gesamte freigelegte Länge des Kabels verteilen.

Die Anschlußfahnen am Stecker verzinnen Sie genau wie das Kabel. Jetzt nur noch die vorbereiteten Teile zusammenhalten und mit dem Lötkolben das Zinn noch einmal zum Schmelzen bringen. Lötkolben wegziehen, beide Teile nicht bewegen und leicht auf die Verbindungsstelle pusten, um das Zinn abzukühlen – so geht es am schnellsten.



Überschuß beschneiden
Jetzt müssen Sie noch die überstehenden Drahtenden wegschneiden. Vorher sollte die Verbindung aber geprüft werden – also ruhig einmal kräftig ziehen, der Draht muß festsitzen. Möglichst kurz abschneiden, damit kein loses Drähtchen einen Kurzschluß herbeiführt – oft die Ursache lästiger Fehler, die man später nur schwer lokalisieren kann.



Daten-Karussell

Floppy-Disk-Laufwerke waren bis vor kurzer Zeit genau wie die sogenannten „Stringy-Floppies“ für das Budget des Heimcomputer-Fans viel zu teuer. Fortschritte in der Technologie machen sie jedoch zunehmend preiswerter.

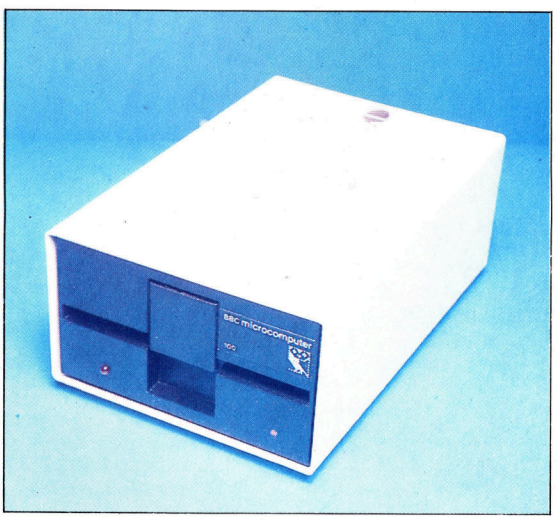
Der Umgang mit Daten ist die eigentliche Aufgabe des Computers. Ohne geeignete Hilfsmittel zum Aufzeichnen von Informationen ist der Rechner jedoch unvollständig: Nach dem Ausschalten sind Daten und Programm spurlos verschwunden. ROM-Cartridges und Cassetten sind als Programm-„Bibliothek“ für den Anfang zwar recht hilfreich, aber wer höher hinaus will, wünscht sich meist die komfortableren Disketten. Lassen Sie uns zunächst einen Blick auf die Alternativen werfen, bevor wir uns den Floppy Disks zuwenden.

Vorsicht, Störung!

Denken Sie daran, daß Floppy Disks immer von Magneten ferngehalten werden müssen. HiFi-Lautsprecher und auch das harmlos erscheinende Telefon enthalten gefährliche Magneten.

Cartridge

Wer selbst programmieren will, kann mit Cartridges wenig anfangen. Die meisten sind mit einem PROM (Programmable Read Only Memory) ausgestattet. Daher dienen sie ausschließlich zum Einlesen von Daten in den Rechner. Es gibt beispielsweise Spiele oder BASIC-Erweiterungen auf Cartridges, die im allgemeinen in Maschinensprache geschrieben sind. Manche enthalten auch die elektrisch löschbaren EEPROMS, die Daten oder Programme ähnlich wie ein RAM aufzeichnen können. Die gespeicherten Informationen bleiben dann auch bei abgeschaltetem Rechner oder herausgenommener Cartridge erhalten.



Bevor dieses Laufwerk mit dem Acorn B betrieben werden kann, muß erst ein DOS-ROM (Disketten-Betriebssystem) montiert werden. Bei „intelligenten“ Laufwerken ist das Betriebssystem schon eingebaut.

Ähnlich arbeiten auch Low Power CMOS-RAMs: Bei diesen werden die Daten durch eine in der Cartridge enthaltene Batterie ständig aufgefrischt und bleiben gespeichert.

Hauptargument gegen EEPROM- und CMOS-RAM-Speicher ist der hohe Preis: Auch eine eher bescheidene Bibliothek dieser Cartridges kostet erheblich mehr als ein entsprechendes Floppy-Disk-Laufwerk.

Cassette

Trotz aller Konkurrenz sind Cassetten nach wie vor die am häufigsten eingesetzten Datenträger. Sie sind nicht nur besonders preiswert, sondern dem Benutzer auch schon aus dem HiFi-Bereich vertraut. Die Anschaffung eines zusätzlichen Abspielgerätes fällt zudem weg, weil für die Datensicherung schon Cassettenrecorder mittlerer Qualität ausreichen. Jedoch können einige Computersysteme, wie zum Beispiel Atari und Commodore, nur mit speziellen Datenrecordern betrieben werden.

Programme und Daten werden auf dem Band in Binärform aufgezeichnet, wobei unterschiedlich hohe Tonfrequenzen jeweils die 0 und die 1 darstellen. Meist wird zuerst der Name eines Datenfiles aufgezeichnet, danach kommen die Daten selbst. Die Übertragung erfolgt Bit für Bit in 1-Byte-Blocks, die zu 256-Byte-Segmenten zusammengefaßt werden. Viele Rechner bieten die Möglichkeit, durch ein Prüfsummen-Verfahren zu kontrollieren, ob die Aufzeichnung ohne Störung verlaufen ist (verify).

Die üblichen Befehle sind SAVE zur Aufzeichnung von Daten und LOAD zum Einspielen zurück in den Computer. Gelegentlich finden sich zusätzliche Befehle für Sonderfunktionen, etwa zur Darstellung eines Inhaltsverzeichnisses, also der Liste der File-Namen.

Leider werden der niedrige Preis und der einfache Umgang auch mit gravierenden Nachteilen bezahlt:

1. In den meisten Fällen muß der Anwender den Recorder per Hand bedienen, also genau im richtigen Augenblick Start- und Stoptaste drücken und die Lautstärkeregelung exakt einstellen.

2. Weil die Daten sequentiell, d. h. aufeinander folgend, gespeichert werden, muß bei der Suche nach einem bestimmten File das Bandzählwerk genau beachtet werden. Oft führt dies zu lästigem Vor- und Zurückspulen, bis die richtige Stelle gefunden ist. Oder der Computer sucht selbst das betreffende File auf

dem Band, wozu es aber einmal ganz durchlaufen muß. Durch die sequentielle Speicherung ist auch der Zugriff auf Daten innerhalb eines Files nicht möglich, ohne das ganze File zu laden – ein großes Handicap des Cassettenrecorders bei der Aufzeichnung größerer Datenmengen, bei denen Einzelzugriff erforderlich ist.

3. Die oben beschriebene Einschränkung macht den Cassettenrecorder zusammen mit einer sehr niedrigen Datenübertragungsrate (meist zwischen 300 und 1200 Bit/s) zu einem äußerst langsamen Speichermedium. Schon ein Programm von fünf KByte Länge erfordert zur Aufzeichnung cirka drei Minuten. Dadurch wird auch die Herstellung der Sicherungskopie zu einem mühsamen Unterfangen.

4. Bänder sind ein unsicherer Datenträger: Nach mehrmaligem Abspielen sind sie oft durch Abrieb beschädigt und unlesbar.

5. Wegen der bei jedem Hersteller unterschiedlichen Eigenheiten der Recorder können die mit dem einen Gerät aufgezeichneten Daten manchmal nicht mit einem anderen Apparat in den Speicher geladen werden. Und leider sind auch Bandrisse noch immer keine Seltenheit.

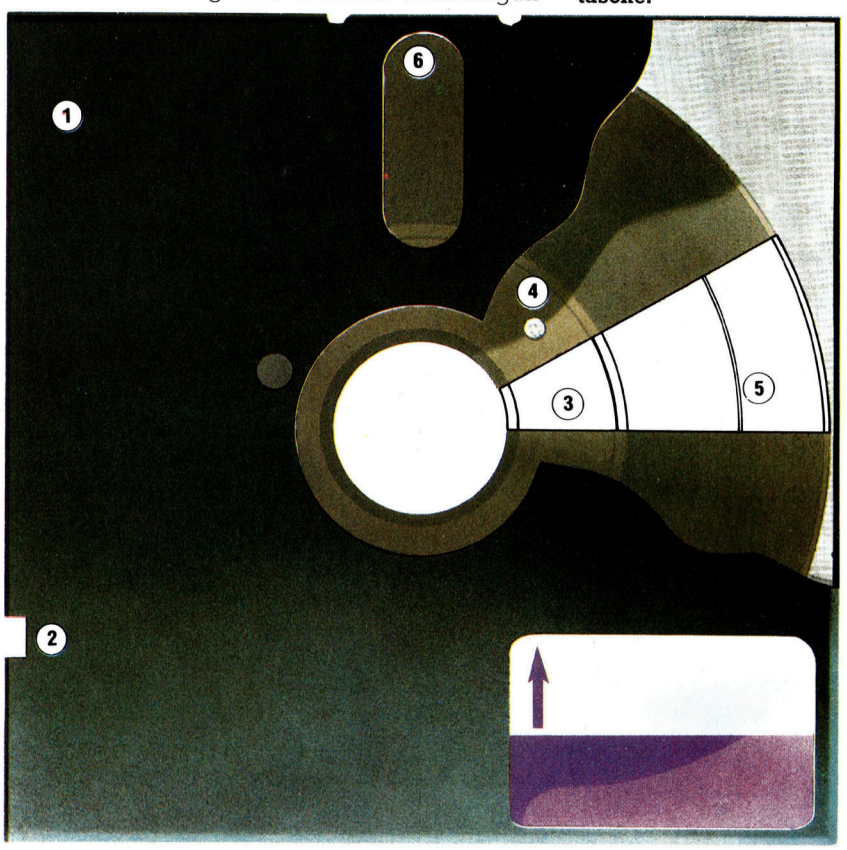
Floppy Disk

Auch Floppy Disks sind natürlich, verglichen mit Festspeichern und Tonbandcassetten, nicht ohne Nachteile. Floppy-Laufwerke sind kompliziert gebaut, empfindlich – und teuer. Unter 500 Mark ist kaum etwas zu machen. Auch die Disketten selbst kosten noch einmal ungefähr acht Mark. Dafür erhält der Anwender allerdings auch ein verläßliches und schnelles Speichermedium, das sich für große Datenmengen eignet und zwischen 50- und 100mal schneller ist als ein Cassettengerät.

Diskettenlaufwerke sind nur mit einem speziellen Betriebssystem (Disk Operating System – DOS) funktionsfähig, das die Verteilung der Informationen auf den einzelnen Spuren (Tracks) der Diskette steuert. Allgemein sind 35 und 80 Spuren pro Plattenseite üblich, wobei jede Spur noch einmal in einzelne Sektoren unterteilt ist. Jeder Sektor enthält einen Datenblock (üblicherweise 256 Bytes).

Das Disketten-Betriebssystem „weiß", wo die einzelnen Informationen auf der Diskette gespeichert sind. Das BAM (Block Aviability Map) beinhaltet ein Verzeichnis der schon benutzten und noch unbelegten Blocks. Dazu dient entweder ein besonderer Bereich auf der Diskette selbst oder im Speicher des Rechners. Normalerweise befindet sich das BAM, das außerdem auch noch ein Verzeichnis der Filenamen, Filetypen und die jeweiligen Anfangs- und Endadressen enthält, auf der inneren Spur der Floppy Disk. Das Disketten-Betriebssystem kann so in Zusammenarbeit mit dem BAM den Schreib/Lesekopf immer richtig positionieren. Die Speicherung der Informationen auf einzelnen Spuren und Sektoren ermöglicht beim Diskettenlaufwerk den exakten Zugriff auf einzelne Files ohne Angabe der Startadresse. Wenn nötig, lassen sich die Daten Bit für Bit lesen bzw. überschreiben. Sie können Disk-Laufwerke grob unterscheiden: nach ihrer Speicherkapazität (meist zwischen 100 und 400 KBytes), nach der Lesegeschwindigkeit und nach den Möglichkeiten, die für die Beeinflussung des Schreib/Lesevorganges über das interne Betriebssystem (DOS) vorhanden sind.

Disketten bestehen aus Mylar oder anderen flexiblen Kunststoffolien. Die Oberfläche ist mit einem Metalloxid beschichtet, das magnetisiert werden kann. Die Scheibe steckt in einer Schutztasche. Wenn sie benutzt wird, liegt der Schreib/Lesekopf des Laufwerkes über dem Schlitz in der Schutztasche.



1. Schutztasche
2. Schreibschutz
3. Sektor
4. Fabrikations-Kennzeichen
5. Spur
6. Schreib/Lese-Schlitz

Als „intelligentes" Laufwerk bezeichnet man Geräte, deren Disketten-Betriebssystem in ein eingebautes ROM integriert ist, das zusammen mit dem ebenfalls eingebauten Prozessor den Schreib-/Lese- und Überwachungsvorgang unabhängig vom Computer bewältigt. Das laufende Programm wird in diesem Fall nicht durch die Funktion des Diskettenlaufwerks unterbrochen.

Sehr verbreitet ist die Methode, das Disketten-Betriebssystem nach dem Einschalten des Computers automatisch von der Floppy Disk in das Computer-RAM zu laden. Bei der dritten Möglichkeit ist das DOS ein Teil des Computer-Betriebssystems. Diesen Weg schlägt Sinclair bei seinen Rechnern ein. Von Acorn stammt ein DOS namens „Disk Filing", das beim Acorn B beschränkte Laufwerkskontrolle ermöglicht und bereits Funktionen wie LOAD, SAVE und CAT beinhaltet. Selbst das Formatieren von Disketten ist direkt möglich.

Listenverarbeitung

In diesem Teil des LOGO-Kurses erfahren Sie mehr über Listenverarbeitung, einen der wichtigsten und vielseitigsten Arbeitsbereiche, den die Sprache bietet.

Der Ausdruck „Liste" wird in LOGO für Elemente benutzt, die in eckigen Klammern stehen, zum Beispiel stellt [LONDON PARIS ROM] eine Liste dar. In den vorangegangenen Folgen des Kurses wurden Listen bereits häufig verwendet, und zwar mit unterschiedlichen Anwendungen. Etwa für die Definition eines Quadrates – REPEAT 4 [FD 50 RT 90]. Die Liste enthält hier die Anweisungen für die Vorwärts- und Drehbewegungen. Ähnlich auch bei der Zeile MAKE „INP REQUEST, wo der Variablen INP die über die Tastatur eingetippten Eingaben in Form einer Liste zugewiesen werden.

Eine Liste kann verschiedene Elemente beinhalten. [[KRAUTSALAT] MIT SPECK] ist eine Liste, deren erstes Element wiederum eine Liste darstellt, nämlich [(KRAUTSALAT)].

Bis zu diesem Zeitpunkt wurde im LOGO-Kurs nur mit einzelnen Zahlen oder Worten gearbeitet. Will man jedoch mehrere Objekte gleichzeitig behandeln, so müssen diese in einzelnen Gruppen zusammengefaßt werden. Der Weg dafür ist die Verwendung von Listen, da sich diese vielseitig einsetzen lassen.

Die ersten wichtigen Anweisungen für die Manipulation von Listen lauten FIRST und BUTFIRST. FIRST [LONDON PARIS ROM] gibt immer das erste Element einer Liste aus, in diesem Fall LONDON. BUTFIRST [LONDON PARIS ROM] druckt dagegen PARIS ROM aus; also sämtliche Listenelemente mit Ausnahme des ersten.

Nun folgt eine Prozedur, bei der die Elemente der Liste untereinander ausgegeben werden:

```
TO PRINTOUT :LIST
  PRINT FIRST :LIST
  PRINTOUT BUTFIRST :LIST
END
PRINTOUT [LONDON PARIS ROM] ergibt:
LONDON
PARIS
ROM
```

Der erste Befehl druckt das erste Element aus und ruft anschließend PRINTOUT auf, der den Rest der Arbeit erledigt. Sobald die vorhandenen Daten ausgegeben sind, stellt der Computer eine Fehlermeldung dar, da sich die Prozedur weiterhin selbst aufruft, jedoch keine Daten findet. Doch auch dafür gibt's eine Lösung:

```
TO PRINTOUT :LIST
  IF EMPTY? :LIST THEN STOP
  PRINT FIRST :LIST
  PRINTOUT BUTFIRST :LIST
END
```

EMPTY? prüft, ob sich noch Elemente in der Liste befinden bzw. ob es sich um eine „leere Liste" handelt. Einige LOGO-Versionen kennen die Abfrage EMPTY? nicht (siehe LOGO-Dialekte). Mit dieser Prozedur kann die Funktion jedoch imitiert werden:

```
TO EMPTY? :LIST
  IF LIST = [] THEN OUTPUT "TRUE
  OUTPUT "FALSE
END
```

Die Anweisungen LAST und BUTLAST arbeiten ähnlich wie FIRST und BUTFIRST. LAST [LONDON PARIS ROM] gibt ROM aus, während BUTLAST [LONDON PARIS ROM] die Elemente LONDON PARIS darstellt.

Für den ersten Ausflug in die Welt der Listenverarbeitung werden wir einige Wörter einsetzen, die ein Patient auf dem Sofa des Psychoanalytikers sagen könnte:

```
MAKE "WOERTER [MUTTER VATER
    GESCHLECHT MOERDER EIFER-
    SUCHT FEUER MEER TOD TRAUM]
```

Da Sie aus Erfahrung wissen, daß diese Ausdrücke das Interesse eines Psychoanalytikers erwecken, soll nun ein konstanter, aber in der Reihenfolge unterschiedlicher Wortfluß aus diesen Begriffen erzeugt werden. Für diesen Zweck wird eine Zufallsvariable (N) definiert, die jeweils ein Element der Liste zwischen dem ersten und dem Nten (in diesem Fall sind es neun) heraussucht.

```
TO NTE :NO :LIST
  IF :N = 1 THEN OUTPUT FIRST :LIST
  OUTPUT NTE :NO — 1 BUTFIRST :LIST
END
```

Geben Sie jetzt zum Beispiel NTE 1 :WOERTER ein, ist die Bedingung in der ersten Zeile „wahr", und als Ausgabe erscheint das Element FIRST :WOERTER, in diesem Fall also das Wort MUTTER.

Bei der Eingabe NTE 2 :WOERTER dagegen ist das Ergebnis der Abfrage negativ, so daß das erste Listenelement ignoriert und statt dessen das erste der restlichen Liste (VATER) ausgegeben wird.

Das ist die Prozedur, die per Zufallsgenerator Wörter aus der Liste heraussucht:

```
TO ZUFALL :LIST
  OUTPUT NTE ((RANDOM 9) +1) :LIST
END
```

Geben Sie nun ZUFALL :WOERTER ein.

Dieses Programm ist auf neun Wörter begrenzt. Das nächste Beispiel zeigt, wie sich die Länge einer Liste abfragen läßt:

```
TO LAENGE :LIST
  IF EMPTY? :LIST THEN OUTPUT 0
  OUTPUT 1 + LAENGE BUTFIRST :LIST
END
```

Zur Demonstration der Arbeitsweise tippen Sie LAENGE [SCIENCE FICTION] ein. Da es sich nicht um eine leere Liste handelt, ist die erste Abfrage negativ und die Prozedur ergibt 1 + LAENGE [FICTION]. Die nächste Abfrage lautet LAENGE [FICTION] 1 + LAENGE [], womit nun die Bedingung in der ersten Zeile erfüllt wäre (OUTPUT 0). Das endgültige Ergebnis von LAENGE [SCIENCE FICTION] heißt 1+1=2. Bauen Sie jetzt diese Befehle in die zuvor gezeigte Prozedur ein, um die Beschränkung von neun Wörtern aufzuheben:

```
TO ZUFALL :LIST
  OUTPUT NTE ((RANDOM LAENGE :LIST)
    +1) :LIST
END
```

In einigen LOGO-Versionen sind bereits Anweisungen definiert, die diese Abfragen selbständig vornehmen. ITEM macht das, was hier mit NTE erreicht wurde, und COUNT steht für LAENGE. Mit Hilfe dieser Befehle läßt sich die Prozedur wie folgt umschreiben:

```
TO ZUFALL :LIST
  OUTPUT ITEM ((RANDOM COUNT :LIST)
    + 1) :LIST
END
```

Um nun beim Psychoanalytiker-Programm eine Zufalls-Auswahl zwischen zehn Begriffen treffen zu können, geben Sie ein: REPEAT 10 [PRINT ZUFALL :WOERTER]

Nachdem wir feststellten mußten, daß unser Psychoanalytiker nicht durch einzelne Wörter zu beeindrucken war, sollen jetzt Sätze mit Hilfe des Zufallsgenerators erzeugt werden.

```
TO GEDICHT1 :LAENGE
  IF :LAENGE = 0 THEN PRINT "STOP
    (PRINT1 " ' ' ZUFALL :WOERTER)
  GEDICHT1 :LAENGE —1
END
```

Das Lied der Suppenschildkröte

Ein Auszug aus dem Lied der Falschen Suppenschildkröte aus dem Buch „Alice im Wunderland" von Lewis Carroll. Rhythmische Verse können leider nicht mit Hilfe der Listenverarbeitung von einem Computer erzeugt werden.

"Bitte, geh doch etwas schneller!" sprach der Weißfisch zu der Schnecke.
"Hinter uns – dreh dich nicht um – krabbelt zwickzwack eine Zwecke.
Sieh! Die Schildkröt und der Hummer laufen schon aufs Ufer zu!
Und sie warten schon am Strande – sagst du mir das Tänzchen zu?
Willst du, magst du, willst du, magst du, sagst du mir das Tänzchen zu?
Willst du, magst du, willst du, magst du, sagst du mir das Tänzchen zu?
Denk dir doch, wie schön das wird, ich kanns genügend kaum erläutern,
Wenn sie uns aufs offne Meer zusammen mit den Hummern schleudern!"
"Zu weit! Zu weit!" die Schnecke spricht und schaut dabei auf ihre Schuh,
Dankt dem Weißfisch allerherzlichst, doch sie sagt den Tanz nicht zu:
Könnt und möcht nicht, könnt und möcht nicht, sagt den Tanz nicht zu,
Könnt und möcht nicht, könnt und möcht nicht, sagt den Tanz nicht zu.

PRINT " ' ' bewirkt, daß zwischen den Wörtern ein Leerzeichen ausgegeben wird. Um einen Satz mit sechs Wörtern zu bilden, geben sie GEDICHT1 6 ein.

Es wäre jedoch sinnvoll, einen Weg zu finden, die Anzahl der Listenelemente variabel zu halten, damit man diese später beliebig erweitern kann und dabei nicht die gesamte Prozedur umschreiben muß. Für diesen Zweck verwendet man die Operation SENTENCE, die zwei Eingaben in eine Liste umwandelt. So gibt SENTENCE "HONIG [MARMELADE SIRUP] nach erfolgter Listen-Zuweisung [HONIG MARMELADE SIRUP] aus.

```
TO ADDWORDS1 :LIST
  MAKE "WOERTER SENTENCE :LIST
    :WOERTER
END
```

Mit dieser Prozedur läßt sich WOERTER erweitern. Ein Problem entsteht, wenn der Variablen WOERTER zuvor noch kein Wert zugewiesen wurde. Der Befehl THING? überprüft dies durch eine Abfrage, wobei wiederum TRUE ausgegeben wird, wenn die Variable bereits einen Wert enthält. Dazu wieder ein Beispiel:

```
TO ADDWORDS1 :LIST
  IF NOT THING? "WOERTER THEN MAKE
    "WOERTER[]
  MAKE "WOERTER SENTENCE :LIST
    :WOERTER
END
```

Unter Verwendung dieser Prozedur und einer veränderten Liste entstand eine merkwürdig anmutende Wortmischung:

```
GESPENST LAUT SPRECHEN GLAENZEND
    PARANOID
    PLANET ERSCHRECKEN DER MIT
    GRUEN GESPENST
    SCHWEBEND PARANOID ROBOTER
    MENSCH FLIEGEN
    SPRECHEN SCHWEBEND LAUT
```

Bei diesem Zufalls-"Text" kann natürlich die Grammatik nicht berücksichtigt werden. Sinnvoller wäre es, wenn man eine Mischung aus Substantiv, Verb, Substantiv erreichen könnte. Dazu müssen mehrere Listen angelegt werden, die verschiedene Wort-Kategorien enthalten. Anschließend werden diese Listen je nach Vorgabe und gewünschter Satzlänge aufgerufen und die Elemente zusammengefügt. Vielleicht finden Sie eine Möglichkeit, dieses zu realisieren.

LOGO-Dialekte

Einige LOGO-Versionen verfügen nicht über die Befehle EMPTY?, ITEM und COUNT. Ersetzen Sie gegebenenfalls

EMPTY? durch EMPTYP
LIST? durch LISTP
PRINT1 durch TYPE

In manchen LOGO-Dialekten ist die Abfrage EQUALP definiert, die zwei Eingaben miteinander vergleicht. Ersetzen Sie das Gleichheitszeichen an den Stellen, wo Listen und Wörter verglichen werden, durch diesen Befehl.

Denken Sie an die unterschiedliche Schreibweise bei IF-Befehlen:

```
IF EMPTYP :LIST
  [OUTPUT 0]
```

Beim Atari-LOGO kann SENTENCE nur in Form von SE eingesetzt werden. ITEM läßt sich hier nicht verwenden.

Abkürzungen

BUTFIRST BF
BUTLAST BL
SENTENCE SE

Übungen

1. Schreiben Sie eine Prozedur, bei der die Elemente in umgekehrter Reihenfolge ausgegeben werden, das heißt das letzte zuerst. Verwenden Sie dabei LAST und BUTLAST.

2. Entwickeln Sie ein Programm, das einzelne Elemente aus der Liste löscht, z. B. soll bei DELETE "SPEISEN [GETRAENKE SPEISEN] als Ausgabe [GETRAENKE] erscheinen.

Lösungen

1. Berechnungen:

```
TO POWER :A :N
  IF NOT ( ( INTEGER :N ) = :N ) THEN PRINT
  [WHOLE NUMBER INDICES ONLY] STOP
  IF :N = 0 THEN OUTPUT 1
  OUTPUT :A * POWER :A :N – 1
END
```

2. Umwandlung in Hexadezimalzahlen:

```
TO HEX.PRINT :NO
  IF :NO < 10 THEN OUTPUT :NO
  IF :NO = 10 THEN OUTPUT "A
  IF :NO = 11 THEN OUTPUT "B
  IF :NO = 12 THEN OUTPUT "C
  IF :NO = 13 THEN OUTPUT "D
  IF :NO = 14 THEN OUTPUT "E
  IF :NO = 15 THEN OUTPUT "F
END

TO HEX :NO
  IF :NO = 0 THEN STOP
  HEX QUOTIENT :NO 16
  PRINT1 HEX.PRINT REMAINDER :NO 16
END
```

3. Ist die Zahl gerade?

```
TO EVEN? :NO
  IF ( ( REMAINDER :NO 2 ) = 0 ) THEN OUTPUT
  "TRUE OUTPUT "FALSE
END
```

4. Anwendung der "Monte-Carlo-Methode":

```
TO MC
  DRAW PU MAKE "IN 0
  MC1 1000 10 100
  ( PRINT [AREA IS] ( :IN ) )
END

TO MC1 :NO :XNO :YNO
  IF :NO = 0 THEN STOP
  RANDOM.POINT :XNO :YNO
  IF INSIDE? THEN MAKE "IN :IN + 1
  MC1 :NO – 1 :XNO :YNO
END

TO RANDOM.POINT :XNO :YNO
  SETXY RANDOM :XNO RANDOM :YNO
END

TO INSIDE?
  IF YCOR < XCOR * XCOR THEN OUTPUT "TRUE
  OUTPUT "FALSE
END
```

Buchführung

Kommerzielle Systeme lassen sich am leichtesten über die Verfolgung des Geldweges verstehen. Auf Kleinbetriebe ausgerichtete Softwarehersteller versuchen, mit diesem Ansatz eine allumfassende Lösung in einem Programmpaket zu bieten.

Anhand von drei Programmen wollen wir die Merkmale eines kommerziellen Paketes untersuchen: „The Quick-Count's Bookkeeping System For The Cash Trader", ein Cassettenprogramm für den Commodore 64, „Accountant" von Compact Accounting Services, für den Acorn B mit zwei Diskettenlaufwerken, und „Microledger" von Lewis Ashley Computers, ein Diskettensystem, das auf dem Apple II läuft.

Ein vollständiges Buchhaltungsprogramm muß einzelne Beträge nach Kostenstellen und Einkunftsart aufteilen können. Geschäftsleute wollen für einen bestimmten Zeitraum nicht nur die Gesamtausgaben oder Einnahmen kennen, sondern brauchen für eine Analyse die einzelnen Komponenten dieser Summen. Sie möchten wissen, wieviel Geld für Miete, Reisen, Büromaterial etc. ausgegeben wurde und können mit Gesamtsummen nur wenig anfangen.

Damit eine Analyse überhaupt möglich ist, muß ein Programm Einkünfte und Ausgaben bestimmten „Kontenklassen" zuordnen können. Ein „Hauptbuch" enthält all diese Zuordnungsmöglichkeiten und bildet daher das Herz jedes Buchhaltungssystems.

Hauptbuch mit 79 Konten

Das Programm „Cash Trader" ist ein gutes Beispiel für den Leistungsumfang und die Grenzen eines Cassettenprogramms. Es besitzt ein Hauptbuch mit 79 verschiedenen Konten und ist nicht mit Diskettensystemen zu vergleichen, die mit Hunderten von Konten arbeiten. Dennoch können Einzelhändler damit im Grunde alle Aspekte ihres Geschäfts erfassen.

Die Konten im Hauptbuch des Cash Traders sind vorgegeben (im Gegensatz zu Accountant und Microledger, deren Kontenrahmen sich frei definieren läßt). „Vorgegeben" bedeutet, daß im Hauptbuch mögliche Kontennummern bereits auf die verschiedenen Kontenklassen verteilt sind. Das Programm „weiß" daher, welche Konten zur Bilanzierung miteinander zu verrechnen sind, und ist für einen Anfänger leichter zu bedienen. Der starre Kontenrahmen und die feste Struktur der Bilanz können aber auch Nachteile haben.

In dem Programm sind die Konten 01 bis 19 für die eingehenden Beträge vom Verkauf und Kauf reiner Handelsgüter vorgesehen. Möglicherweise werden dafür aber weniger Konten benötigt. Beispielsweise könnten im Konto 01 einfach alle „Einnahmen" aus Verkäufen gebucht werden. Eine Summierung entfällt dabei.

In einem anderen Fall benötigt ein Händler vielleicht mehrere Konten für verschiedene Warenarten und möchte diesen die Kontonummern 01 bis 05 zuordnen. Die einzelnen Beträge müßten dann natürlich den entsprechenden Konten angewiesen werden, wobei das Programm die Summierung automatisch ausführt.

Die Konten 20 bis 49 behandeln Gewinn und Verlust. Hier werden die unterschiedlichen Arten von Ausgaben verzeichnet. Die Titel der Konten sind daher auch: Miete, Löhne, Strom,

Arthurs Buchhaltung für diese Woche sieht folgendermaßen aus: Drei Kisten Glen Kyushu Whiskey gegen Bargeld verkauft und 10 Kisten gegen Kreditkarte; ein Scheck über 2000 Mark für „Dienstleistungen" empfangen und den Rolls Royce gegen Bargeld verkauft. Bar bezahlt wurden neue Whiskeykäufe und das Gehalt von Terry, seinem einzigen Angestellten. Weiterhin wurde ein Scheck für einen neuen Wagen ausgestellt.

Bankgebühren, Zinsen, Büromaterial, Werbung und Telefon.

Die Konten 50 bis 79 beziehen sich auf die Bilanz. Sie geben jederzeit einen Überblick über die finanzielle Gesamtsituation des Geschäftes. Darin enthalten sind z. B. Anlagegüter (der Geldwert aller Anlagegüter), die Salden aller Bankkonten, der Mehrwertsteueranteil und die ausstehenden Forderungen (d. h. wieviel Geld die Kunden dem Geschäft noch schulden).

In einem herkömmlichen Buchhaltungssystem werden Käufe und Verkäufe in ein Tagesjournal eingetragen, das die Gesamtsumme aller Einnahmen und Ausgaben auf täglicher oder wöchentlicher Basis anzeigt. Das Menü des Cash Traders sieht folgendermaßen aus:

1. Tageseinnahmen
2. Barausgaben
3. Zahlungen per Bank
4. Journal

Mit der Option 1 werden die Wocheneinnahmen aufgerufen. Der Anwender gibt einen bestimmten Tag an (1–7) und trägt die Summe der Tageseinnahmen ein. Die einzige Unterscheidung, die an dieser Stelle möglich ist, ist die Kennzeichnung einer Tageseinnahme als „Sonder-Feld".

Mit dieser Kennzeichnung lassen sich außerordentliche Einnahmen in der Bilanz gesondert ausweisen. Verkauft z. B. ein Gärtner seinen Lieferwagen, dann ist das eine außerordentliche Einnahme, die die Statistik der Einnahmen verzerren würde, wenn sie nicht besonders gekennzeichnet wäre. Dies ist allerdings die einzige Möglichkeit, mit der Cash Trader Unterscheidungen in den Gesamteinnahmen vornehmen kann. Der Anwender kann jetzt angeben, welchem Konto die Tageseinnahmen zugeordnet werden sollen: dem Bankenkonto, dem Bargeldkonto oder dem Kreditkarten-Konto.

Zahlungen können dem Bargeldkonto oder dem Bankenkonto zugeordnet werden. Eine dreistellige Bezugsnummer kennzeichnet die Art jeder Zahlung, und ein Feld mit sechzehn Zeichen ermöglicht die Angabe des Zahlungsempfängers. Dabei wird automatisch die Mehrwertsteuer berechnet und in das Mehrwertsteuerkonto eingetragen.

Die Programme Microledger und Accountant haben eine völlig andere Struktur. Hauptunterschied ist dabei, daß keine Kontenbereiche vorgegeben sind. Dem Anwender wird bei Accountant eine achtstellige Code-Struktur zur Verfügung gestellt, mit deren Hilfe ein eigener Kontenrahmen eingetragen werden kann. Microledger bietet eine dreistellige Code-Struktur, folgt sonst aber dem gleichen Prinzip.

Accountant baut ebenfalls auf einem Hauptbuch auf, läßt aber mehr Einzelheiten und Kontenklassen zu und bietet zusätzlich Analysemöglichkeiten. Aber auch hier müssen die Einträge – im Gegensatz zu einem vollausgebauten Buchhaltungssystem – teilweise noch zusammengefaßt werden.

Bei einem vollwertigen Buchungssystem kann der Anwender alle Lieferanten und Kunden in einer Hauptdatei führen und dort Einzelheiten wie z. B. ausstehende Rechnungen und Bestellungen vermerken. Dabei lassen sich alle ausstehenden Rechnungen und die einzelnen Zahlungseingänge im Gesamtüberblick darstellen.

Das Programm Accountant bietet diesen Komfort nicht. Wie der Cash Trader akzeptiert er nur tageweise zusammengefaßte Einträge. Da es durch das Diskettensystem aber weitaus mehr Speicher zur Verfügung hat, braucht die Eingabe nicht in allzu komprimierter Form zu erfolgen.

So kann der Anwender statt der Gesamtverkaufssumme eines Tages soviele Beträge wie nötig eingeben. Das System verarbeitet dabei fünf verschiedene Arten: Rechnungen, Kreditverkäufe, Barverkäufe, Bareingänge und vermischte Eingänge. Jeder Eintrag kann eine Beschreibung mit bis zu 16 Zeichen erhalten, wird mit einer laufenden Nummer versehen und läßt sich für die Analyse in eine beliebige Anzahl Summenkonten des Hauptbuches stellen (im Gegensatz zu den drei Möglichkeiten, die der Cash Trader zuläßt).

Das tägliche Kassenbuch kann Kredit-, Bar- und andere Käufe unterscheiden und bietet eine Übersicht über die Beträge und Prozentsätze der Mehrwertsteuer.

Vollwertige Fakturierung

Im Gegensatz zu diesen beiden Systemen bietet Microledger eine vollwertige Fakturierung mit Buchführung. Bis zu 999 unterschiedliche Konten können angegeben werden, wobei jedes Konto mit bis zu fünf Zeilen durch Namen und Adresse des Kunden dokumentiert werden kann. Der Umsatz dieses Kunden sowie die Summe der ausstehenden Beträge werden automatisch gespeichert.

Der Hauptunterschied zwischen Microledger und den beiden anderen Programmen liegt in der Informationsmenge, die über Lieferanten und Kunden von der Software akzeptiert wird. Der Cash Trader faßt diese Informationen in ein oder zwei Summen zusammen. Das Programm Accountant kann zwar einzelne Käufe oder Verkäufe speichern, die Salden einzelner Kunden oder Lieferanten müssen aber per Hand ausgerechnet werden.

In unserem nächsten Artikel werden wir am Beispiel dieser drei Programmpakete untersuchen, wie Werte in die einzelnen Konten eingegeben werden. Dabei gehen wir auch auf die Listen ein, mit denen die Daten einzelner Konten angezeigt und für Kostenanalysen genutzt werden können.

Buchhaltungs-Programme

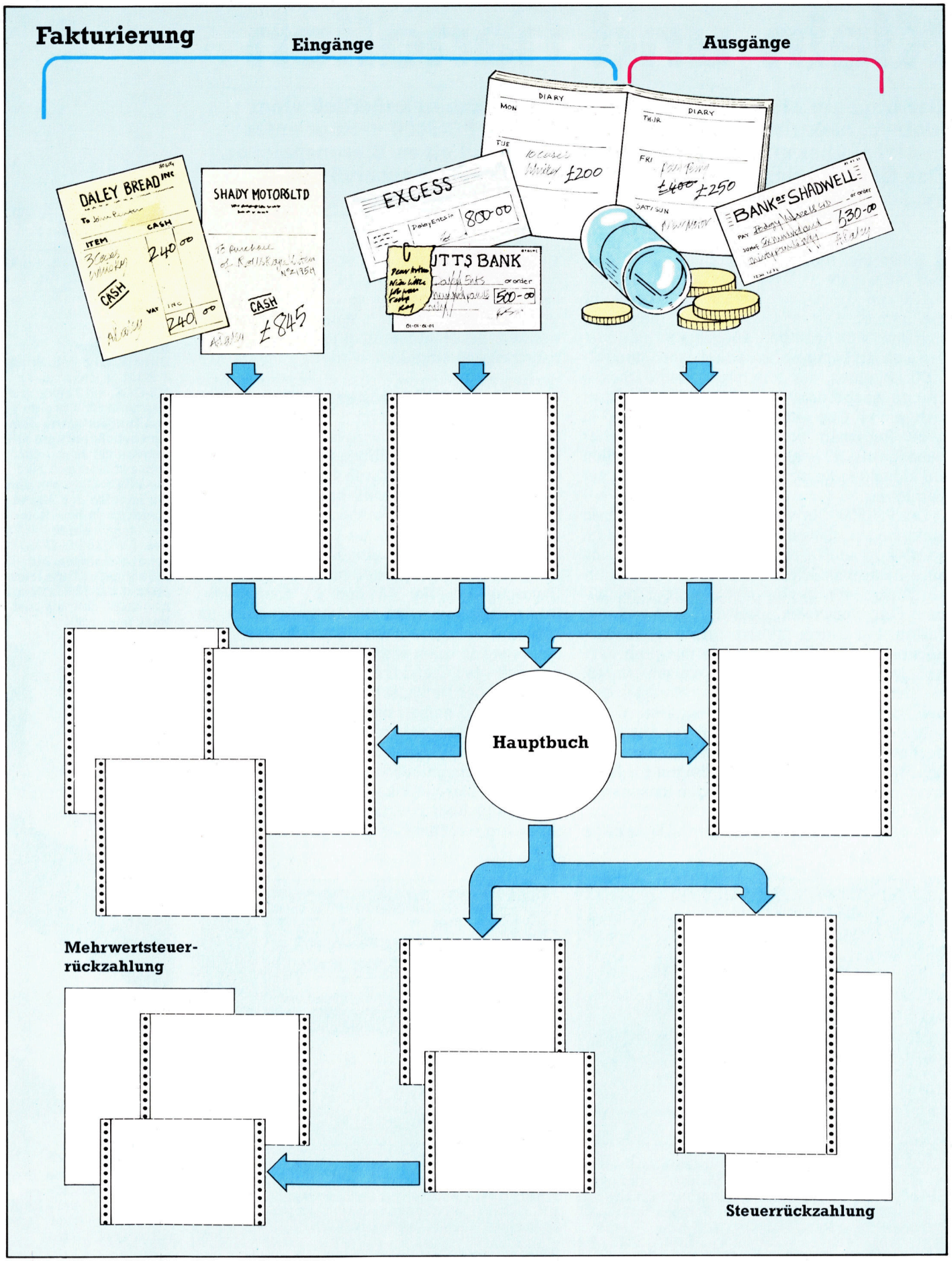
Typische Einnahmen
Alle Eingänge werden nach der Art des Verkaufs (BAR, BANK oder KREDITKARTE) und als „Ware" oder „Spezial" nach der Art des Geschäftsvorgangs eindeutig gekennzeichnet.

Typische Ausgaben
Auch hier werden die Arten der Bezahlung und der jeweilige Geschäftsvorgang angegeben. Die Mehrwertsteuer und die zu belastende Kostenstelle werden ebenfalls genannt.

Summen
Die Summen für Einnahmen und Ausgaben werden automatisch angezeigt.

Hauptbuch
Es gibt 79 Konten, die die Bezeichnungen Einnahmen, Ausgaben, Lagerbestand, Gehälter, Miete, Bankgebühren, Anlagegüter, Bargeld etc. tragen. Für die Bilanz sind diese Konten in Klassen unterteilt, z. B. Umsatzkonto, Gewinn- und Verlustkonto und Bilanzkonten.

Listen
Es gibt eine ganze Anzahl von Listen. Nach Eingabe der Daten interessiert einen Geschäftsmann aber vor allem der Zustand seines Bargeldkontos, des Bankenkontos und die Zwischenbilanz. Die Gesamtbilanz und das Mehrwertsteuerkonto sind nur in größeren Zeiträumen interessant.


Fakturierung
Eingänge
Ausgänge
DALEY BREAD INC
CASH
ITEM
240 00
CASH
VAT INC
240 00
SHADY MOTOR&LTD
CASH
£845
DIARY
MON
TUE
£200
DIARY
THUR
FRI
£400
£250
SAT/SUN
EXCESS
800 00
UTTS BANK
500-00
BANK OF SHADWELL
630-00
Hauptbuch
Mehrwertsteuer-
rückzahlung
Steuerrückzahlung

Klein und handlich

Der tragbare MS-DOS-Computer von Sharp ähnelt äußerlich einer kleinen, elektrischen Schreibmaschine. Der PC-5000 verfügt unter anderem über eine Flüssigkristallanzeige und einen Blasenspeicher. Das Gehäuse des Computers ist speziell für den Anschluß eines zusätzlichen Druckers ausgelegt.

Unter den tragbaren Computern sind derzeit Geräte im Telefonbuchformat wie der HX-20 von Epson und der TRS-80 Modell 100 die Renner. Fast alle Maschinen dieser Art sind mit LCD-Anzeigen ausgestattet und lassen sich mit Batterien betreiben. Der Sharp PC-5000 ist eines der teuersten Geräte dieses Trends. Ausgerüstet mit einem Blasenspeicher statt eines Cassettenlaufwerkes stößt er in neue Regionen vor. Magnetblasenspeicher benötigen nur wenig Elektrizität und stellen auf kleinstem Raum viel Speicherkapazität zur Verfügung.

Der PC-5000 ähnelt einer kleinen, tragbaren elektrischen Schreibmaschine. Unter dem hochklappbaren Gehäusedeckel befindet sich eine Schreibmaschinentastatur mit konturierten Tasten, von denen acht als programmierbare Funktionstasten ausgelegt sind und vier Tasten den Cursor steuern. In den Gehäusedeckel ist ein LCD-Bildschirm integriert. Obwohl die Abdeckhaube recht schwer ist, läßt sie sich in mehreren Stellungen einrasten und der Position des Bedieners anpassen. Oberhalb der Tastatur befinden sich die Anschlußbuchse für den Blasenspeicher und drei Leuchtdioden, die als Warnanzeigen für Netzspannung, leere Batterien und den Blasenspeicher dienen.

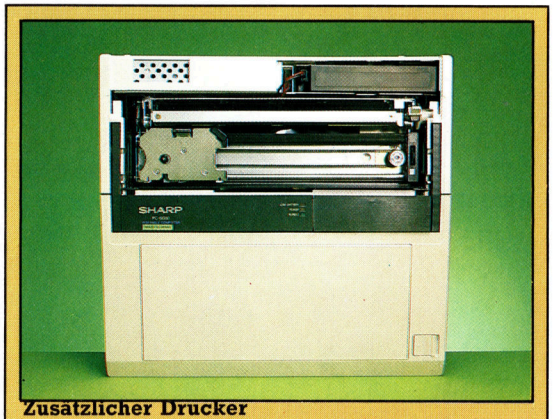
In eine Aussparung hinter der LCD-Anzeige läßt sich ein Thermodrucker einbauen, der als Zusatzgerät angeboten wird. Der Drucker erzeugt ein ausgezeichnetes Schriftbild. Er läßt sich mit Thermo- oder normalem Papier verwenden. Bei letzterem muß jedoch eine Farbbandcassette eingelegt werden.

Viel Speicherplatz

Der PC-5000 stellt dem Anwender einen Arbeitsspeicher von 128 KByte und 192 KByte als ROM zur Verfügung. Darin befindet sich das GW-BASIC von Microsoft, das auch auf dem IBM-PC läuft. Ebenfalls wie bei dem IBM-PC wurden der Intel 8088 als CPU und das Betriebssystem MS-DOS eingesetzt. Der Blasenspeicher läßt sich in der gleichen Form wie Diskettenlaufwerke (A und B) ansprechen. Sharp bietet weiterhin ein Doppellaufwerk für Disketten an, das auf der Rückseite des Gerätes angeschlossen wird. Diese Laufwerke lassen sich über C und D ansprechen, können jedoch nur per Netzanschluß betrieben werden.

Der Sharp enthält mehrere Softwarepakete der Firma Sorcim, darunter SuperCalc, SuperWriter und SuperComm, ein Programm für die Telekommunikation. Die Programme sind in einem Blasenspeicher untergebracht und lassen sich über Funktionstasten aufrufen. Die Belegung der Tasten wird angezeigt.

Doppellaufwerk für Disketten
Dieses Zusatzgerät enthält zwei Diskettenlaufwerke mit je 320 K, die unter MS-DOS laufen. Es wird an die Rückseite des PC-5000 angeschlossen, läuft aber nicht mit Batterien. Fremde Software muß umformatiert werden.

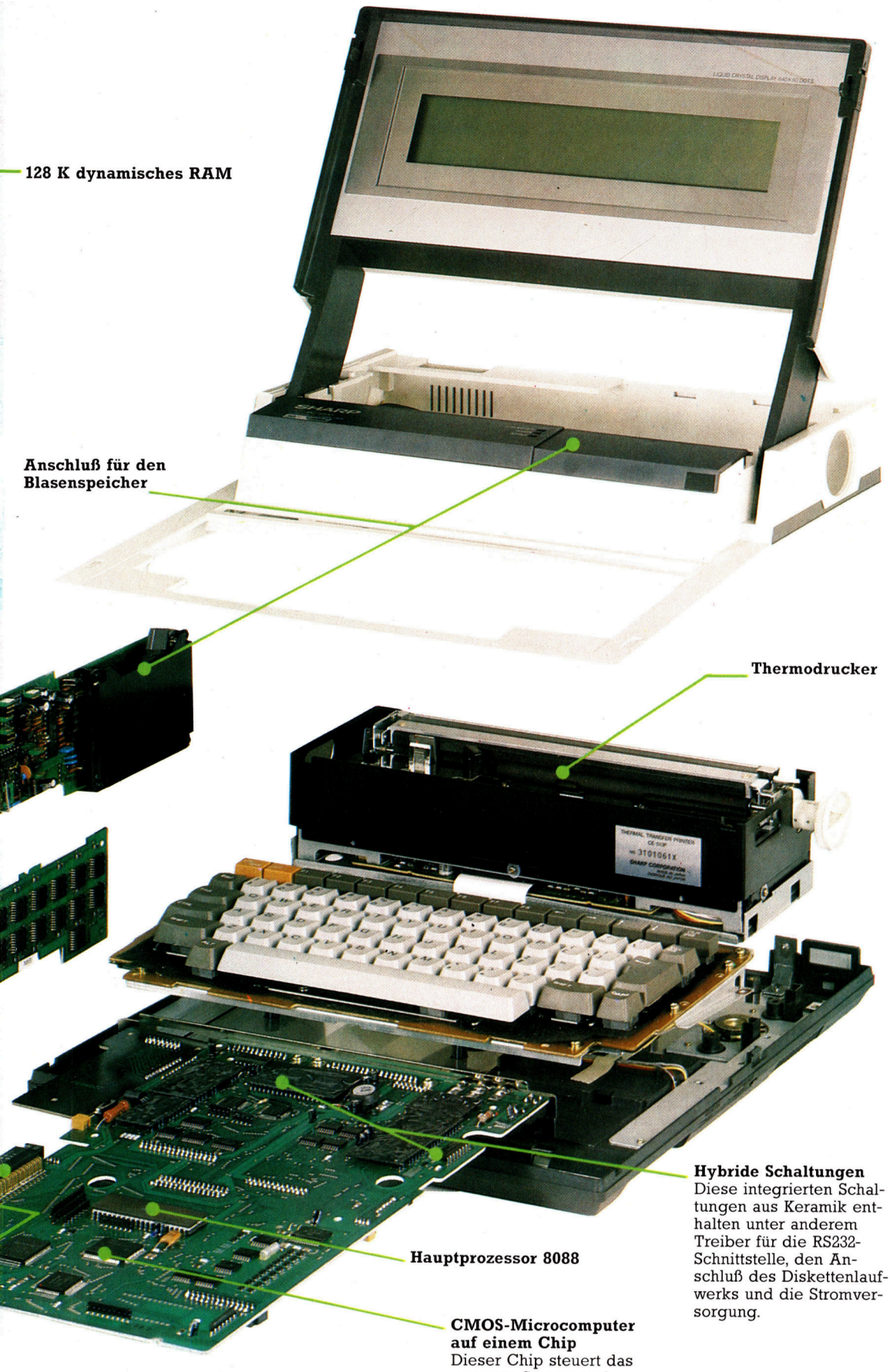


Zusätzlicher Drucker
Von oben ist der zusätzliche Thermodrucker im Gehäuse des PC-5000 zu erkennen. Der Drucker paßt in eine Aussparung im Gerät und ist über ein Flachkabel mit der Systemplatine verbunden. Er druckt 37 Zeichen pro Sekunde.

Steuerplatine für den Blasenspeicher
Diese Platine läßt sich wie ein Diskettenlaufwerk ansprechen, steuert hier jedoch die Ein- und Ausgabe des Blasenspeichers.

ROM-Platine
Diese Platine enthält 192 K ROM, in denen das MS-DOS, die Zeichengeneratoren für Anzeige und Drucker sowie Kommunikationssoftware untergebracht sind. Außerdem befindet sich hier der PISCS-Chip, mit dem die Impulse des Blasenspeichers in eine Sprache umgewandelt werden, die das MS-DOS verarbeiten kann. Mit Hilfe dieses Chips interpretiert das MS-DOS den Blasenspeicher als Laufwerk A und B.

Steckleisten für Erweiterungen
Über diese Steckleisten lassen sich weitere RAM-Module oder ROM-Cartridges anschließen.



Sharp PC-5000

PREIS
ca. 5000 Mark (ohne Drucker)

ABMESSUNGEN
300 x 318 x 90 mm

GEWICHT
5,7 kg mit Drucker

ZENTRALEINHEIT
Intel 16-Bit-8088 und 8-Bit-CMOS

SPEICHERKAPAZITÄT
128 K RAM
192 K ROM

BILDSCHIRM-DARSTELLUNG
Flüssigkristallanzeige, 8 Zeilen mit je 80 Zeichen, grafische Auflösung 640 x 80 Bildpunkte. Eingebauter internationaler Zeichensatz und Grafikzeichen.

SCHNITTSTELLEN
Cassettenanschluß, Externe Diskettenstation, Modemausgang, RS232, Netzanschluß, Steckleisten für RAM und ROM

PROGRAMMIER-SPRACHEN
Microsoft GW-BASIC

TASTATUR
Standard-Schreibmaschinentastatur mit 57 Tasten, 8 Funktionstasten, Cursor-Steuertasten, drei Spezialtasten. Die Funktionstasten sind frei programmierbar, und die jeweilige Belegung kann auf der untersten Bildschirmzeile angezeigt werden.

HANDBÜCHER
Die Handbücher enthalten Informationen über die Inbetriebnahme des PC-5000 und den Gebrauch von MS-DOS. Das BASIC-Handbuch von Microsoft ist für Anfänger zu technisch gehalten.

STÄRKEN
Der PC-5000 verfügt über fast alle Funktionen einer kommerziellen Maschine. Mit dem eingebauten Drucker und den Blasenspeichern ist das Gerät seiner Konkurrenz voraus.

SCHWÄCHEN
Der 5000 ist teurer als vergleichbare Geräte und für seine Größe sehr schwer. Wenn Sie Diskettenbetrieb gewöhnt sind, werden Ihnen die Blasenspeicher eher langsam vorkommen.

Speicheraufbau

In dieser Folge über die Maschinencode-Programmierung zeigen wir, wie ein Computer seine Daten speichert, sein „Gedächtnis" organisiert und auf welche Weise die CPU einzelne Bytes mit Binärzahlen adressiert.

In der ersten Folge haben wir erklärt, wie ein Computer Informationen durch elektrische Impulse speichern kann. Bei dem Beispiel wurde deutlich, wie die vier Schalter und Glühbirnen mit ihren 16 unterschiedlichen Schaltmöglichkeiten die Zahlen 0 bis 15 darstellen können. Mit acht Schaltern und Glühbirnen ergeben sich jedoch 256 individuelle Schaltmuster ($2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 = 256$) für die Zahlen von 0 bis 255.

Der Arbeitsspeicher eines Heimcomputers ist aus Blöcken aufgebaut, die „Bytes" genannt werden und aus je acht Schaltern bestehen. Im allgemeinen kann die CPU nur ein Byte zur Zeit bearbeiten und damit auch nur Zahlen zwischen 0 und 255 addieren, vergleichen oder speichern. Auf den ersten Blick scheint das die Arbeit der CPU stark einzuschränken. Bedenken Sie aber, daß Sie bei der Addition zweier Zahlen (z. B. 63951 + 48770 = ?) auch nur die einzelnen Stellen nacheinander berechnen. Auf genau die gleiche Weise führt die CPU mathematische Aufgaben mit großen Zahlen Byte für Byte durch.

Mit seinen acht Schaltern kann ein Byte eine achtstellige Binärzahl speichern. Jede einzelne Stelle dieser Binärzahl wird „Bit" genannt und stellt die kleinstmögliche Informationseinheit dar. Ein Bit ist entweder AN oder AUS, und eine binäre Stelle ist entweder 1 oder 0.

MSB – Most Significant Bit

Da die einzelnen Bits eines Bytes oft einzeln angesprochen werden, bezeichnet man sie von links nach rechts mit den Zahlen 0 bis 7. Enthält z. B. ein Byte die Binärzahl 00000001, dann heißt es, Bit0 ist 1, oder Bit0 ist AN, oder Bit0 ist gesetzt, und alle anderen Bits sind 0 oder AUS oder CLEAR. In der Binärzahl 01001000 sind daher Bit3 und Bit6 gesetzt, Bit4 ist AUS, Bit7 ist 0, Bit0 ist nicht gesetzt etc. In einem Byte wird Bit0 auch das „niederwertigste Bit" (LSB – Least Significant Bit) genannt, und Bit7 das „höchstwertige Bit" (MSB – Most Significant Bit).

Den Speicher eines Computers kann man sich auch als ein großes Blatt Rechenpapier vorstellen, das acht Kästchen breit und Tausende von Kästchen lang ist. Jede Reihe von acht Kästchen ist ein Byte und jedes Kästchen ist ein Bit innerhalb eines Bytes. Da ein Speicher völlig nutzlos wäre, wenn man die darin enthaltenen Informationen nicht wiederfinden könnte, besitzt jedes Byte eine eigene Bezeichnung: die Adresse. Diese wird nun nicht extra in jeder Kästchenreihe (oder in jedem Byte) geschrieben, sondern ergibt sich aus der Position des Bytes vom Anfang des Speichers an gezählt. Das erste Byte im Speicher hat die Adresse 0, das nächste Byte die Adresse 1, das nächste die Adresse 2 und so weiter. Wenn Sie etwas in das Byte43 schreiben möchten, fangen Sie am Anfang des Speichers (bei Byte0) an, und zählen die Bytes, bis Sie Byte43 erreicht haben.

Offset und Seitenoffset

Die Adresse 43 des Byte43 wird dabei nur von der Position bestimmt – Sie haben von Byte0 aus 43 Bytes abgezählt und erreichten somit Byte43. In den Chips im Inneren des Computers sind die Bytes des Speichers als winzige Blöcke mit je acht Transistorschaltungen (pro Bit eine Schaltung und pro Byte acht Schaltungen) eingeätzt, deren Konstruktion, abgesehen von ihrer Position, identisch ist.

Stellen wir uns den Speicher eines Computers nochmals als Papierstreifen mit einer Breite von acht und einer Länge von Tausenden von Kästchen vor. Dieser Streifen ließe sich an jedem hundertsten Byte (d. h., an der Grenze zwischen dem Byte99 und dem Byte100, zwischen Byte199 und Byte200, Byte299 und Byte300 etc.) zerschneiden. Jeder dieser Teilstreifen wäre jetzt eine Seite mit je 100 Bytes. Seite0 (Page0) fängt bei Byte0 an und geht bis Byte99, Seite1 beginnt bei Byte100 und reicht bis Byte199, und Seite2 reicht von Byte200 bis Byte299 etc. Wenn wir jetzt ein bestimmtes Byte suchen, z. B. das Byte3518, brauchen wir nicht vom Anfang des Speichers an 3518 Bytes abzuzählen, sondern können aus der Adresse des Bytes ersehen, daß es auf der Seite 35 liegen muß. Wir brauchen daher vom Anfang des Speichers nur 35 Seiten abzuzählen und dann vom Anfang dieser Seite 18 Bytes, um Byte3518 zu erreichen.

Eine Aufteilung des Speichers in Seiten ist außerordentlich praktisch, da man jedes Byte leicht finden kann, wenn seine Adresse in zwei Teile zerlegt wird. Die Ziffern von der Hunderterposition aus nach links geben dabei die Sei-

tennummer an und die Ziffern von der Zehnerposition aus nach rechts die exakte Position des Bytes auf dieser Seite. In unserem Beispiel teilen wir die Adresse des Bytes 3518 in zwei Zahlen auf: Seitennummer 35 und Byte Nummer 18 auf einer Seite. Die Zahl 18 wird dabei auch „Offset" oder „Seitenoffset" genannt und bezeichnet den Abstand oder die Distanz einer Adresse zu einer Basisadresse (in diesem Fall ist die Basisadresse Byte3500).

Ein Computer zählt jedoch nicht dezimal, sondern binär. Mit dem Seitensystem muß sich dabei eine bestimmte Seite und ein Offset ebenso leicht wie im Dezimalsystem über die Aufteilung der Adresse finden lassen. Nun wird die Dezimaladresse 99 binär von der Zahl 01100011 dargestellt und dezimal 100 von der Binärzahl 01100100, dezimal 199 = binär 11000111 und dezimal 200 = binär 11001000.

Die Zahl 100 bietet sich im Dezimalsystem als Seitengröße an, da sie ein Vielfaches von 10 ist und sich von anderen Zahlen deutlich unterscheidet. Wenn wir im Binärsystem zählen, müssen wir also eine Einheit wählen, die für dieses System Bedeutung hat. Microcomputer verwenden daher Seitengrößen von 256 Bytes, so daß Seite0 bei Byte0 anfängt und bei Byte255 aufhört, Seite1 bei Byte256 beginnt und bis Byte511 reicht etc. Der Grund für diese Einteilung läßt sich leicht aus der binären Darstellung dieser Zahlen ersehen:

Seite0: Byte00000000 bis Byte11111111
Seite1: Byte100000000 bis Byte111111111

Wie Sie sehen, lassen sich die Zahlen 0 bis 255 mit acht Bits darstellen, die nächste Zahl – 256 – benötigt neun Bits, und mit neun Bits können wir bis 511 zählen. Die nächsthöhere Zahl – 512 – benötigt zehn Bits, mit denen wir bis 1023 zählen können usw. Wenn die Seitengröße also 256 ist und wir im Binärsystem zählen, dann sind die acht Bits auf der rechten Seite der Zahl der Seitenoffset, während sich die Seitennummer aus den Bits ergibt, die von Bit8 an auf der linken Seite der Zahl stehen.

Speicheradressen aus zwei Bytes

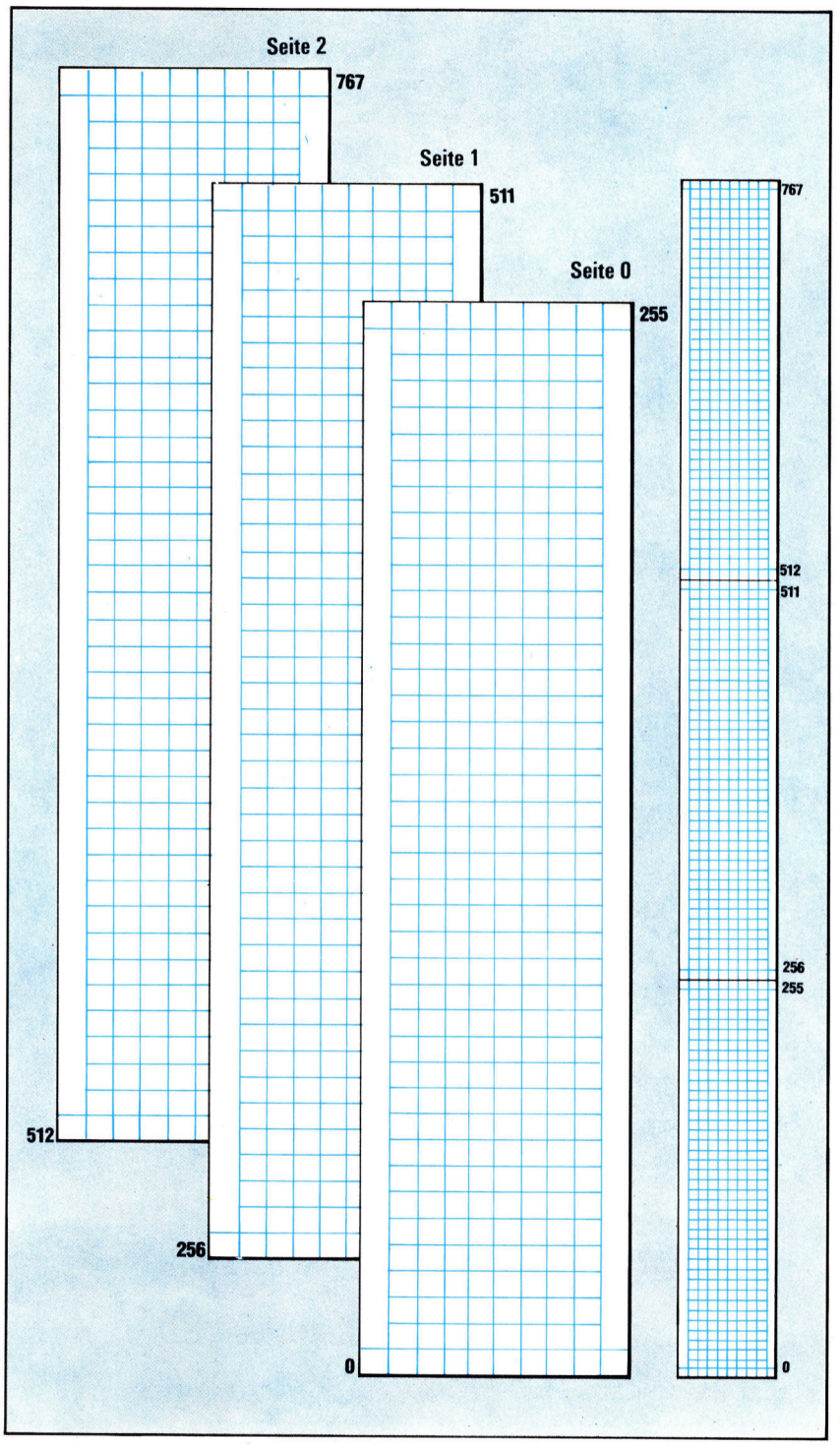
Diese Einteilung mag ein wenig verwirren, da wir früher erwähnt hatten, daß die CPU nur einzelne Bytes verarbeiten kann. Wenn ein Byte nur acht Bits enthält, warum dann Zahlen mit neun oder zehn Bits verwenden? Die Antwort ist einfach: Alle Speicheradressen bestehen aus zwei Bytes, wobei die CPU die beiden Bytes nacheinander bearbeitet. Wenn wir die Seitengrenzen als Zwei-Byte-Zahlen darstellen, wird das gesamte System deutlich:

Seite	0	beginnt bei	00000000 00000000
		endet bei	00000000 11111111
Seite	1	beginnt bei	00000001 00000000
		endet bei	00000001 11111111
Seite	10	beginnt bei	00000010 00000000
		endet bei	00000010 11111111
Seite	11	beginnt bei	00000011 00000000
		endet bei	00000011 11111111

und so weiter.

Aus diesen Beispielen läßt sich leicht ersehen, wie die CPU ein Byte im Speicher über eine Zwei-Byte-Adresse anspricht. Das erste oder links stehende Byte ergibt die Seitennummer und das zweite oder rechte Byte den Offset, die Position der Daten.

Die Seitenadressierung unterteilt den gesamten Speicher in Blöcke oder Seiten von je 256 Bytes Länge. Alle Adressen werden als Zwei-Byte Zahlen dargestellt: Ein Byte gibt die Seitennummer an, während das andere Byte den Abstand vom Anfang dieser Seite (Offset) enthält.



Zahlen-umwandlung

Diese drei Programme für den Commodore 64, den Acorn B und den Spectrum wandeln dezimale Eingaben in die binären Äquivalente um.

Commodore 64

```
10 REM********COMMODORE*********
40 S$="=                ":X$="0123456789ABCDEF"
50 REM    S$ CONTAINS 9 SPACES
60 PRINT CHR$(147)      :REM CLEAR SCREEN
70 PRINT "          TO DISPLAY DECIMAL NUMBER
S"
80 PRINT "      AND THEIR BINARY EQUIVALEN
TS"
90 PRINT:PRINT "    *******ENTER 0 TO QUI
T********":PRINT
100 FOR K=1 TO 1
110 FOR L=1 TO 1
120 INPUT"TYPE ANY POSITIVE WHOLE NUMBER
 ";A$
130 NU=VAL(A$)
140 IF NU=0 THEN PRINT "PROGRAM EXIT":ST
OP
150 IF INT(NU)<>ABS(NU) THEN L=0
160 IF NU>65535 THEN PRINT NU;" IS TOO B
IG":L=0
170 NEXT L
200 NM=NU:H$="":GOSUB 2000
210 PRINT NU;TAB(5);N$;
220 IF RIGHT$(A$,1)="+" THEN GOSUB 4000
230 PRINT H$:PRINT:PRINT
240 K=0:NEXT K
250 END
300 END
1000 REM******BINARY BYTE S/R*******
1010 B$=""
1020 FOR D=8 TO 1 STEP-1
1030 N1=INT(N/2)
1040 R=N-2*N1
1050 B$=MID$(STR$(R),2)+B$
1060 N=N1
1070 NEXT D
1080 RETURN
2000 REM****BINARY CONVERSION S/R***
2010 IF NM<256 THEN N=NM:GOSUB 1000:N$=S
$+B$:RETURN
2020 HI=INT(NM/256):LO=NM-256*HI
2030 N=HI:GOSUB 1000:N$="= "+B$
2040 N=LO:GOSUB 1000:N$=N$+" "+B$
2050 RETURN
3000 REM****HEX BYTE S/R*********
3010 HB=INT(N/16):LB=N-HB*16
3020 B$=MID$(X$,HB+1,1)+MID$(X$,LB+1,1)
3030 RETURN
4000 REM****HEX CONVERSION S/R******
4010 IF NM<256 THEN N=NM:GOSUB 3000:H$="
=    "+B$:RETURN
4020 HI=INT(NM/256):LO=NM-256*HI
4030 N=HI:GOSUB 3000:H$="= "+B$
4040 N=LO:GOSUB 3000:H$=H$+" "+B$
4050 RETURN
```

Acorn B

Folgende Zahlen müssen für den Acorn B geändert werden:

```
60 CLS:@%=5
210 PRINT TAB(0);NU;TAB(5);N$;
1050 B$=STR$(R)+B$
```

Dieses Programm berücksichtigt die Besonderheiten, die der Acorn für die Handhabung von numerischen Werten bietet, nicht. Man könnte das Listing deshalb auch in einer Kurzform schreiben.

Spectrum

```
10 REM*******SPECTRUM*********
40 LET S$="=                ":LET X$="0
123456789ABCDEF"
50 REM    S$ CONTAINS 9 SPACES
60 CLS
70 PRINT "  TO DISPLAY DECIMAL N
UMBERS"
80 PRINT " AND THEIR BINARY EQUI
VALENTS"
90 PRINT:PRINT " *******ENTER 0
TO QUIT********":PRINT
100 FOR K=1 TO 1
110 FOR L=1 TO 1
120 INPUT"TYPE ANY POSITIVE WHOL
E NUMBER ";A$
130 LET NU=VAL(A$)
140 IF NU=0 THEN PRINT "PROGRAM
EXIT":STOP
150 IF INT(NU)<>ABS(NU) THEN LET
 L=0
160 IF NU>65535 THEN PRINT NU;"
IS TOO BIG":LET L=0
170 NEXT L
200 LET NM=NU:LET H$="":GOSUB 20
00
210 PRINT NU;TAB(5);N$;
220 IF A$(LEN A$)="+" THEN GOSUB
 4000
230 PRINT H$:PRINT:PRINT
240 LET K=0:NEXT K
300 END
1000 REM**BINARY BYTE S/R**
1010 LET B$=""
1020 FOR D=8 TO 1 STEP-1
1030 LET N1=INT(N/2)
1040 LET R=N-2*N1
1050 LET B$=STR$(R)+B$
1060 LET N=N1
1070 NEXT D
1080 RETURN
2000 REM**BINARY CONVERS S/R**
2010 IF NM<256 THEN LET N=NM:GOS
UB 1000:LET N$=S$+B$:RETURN
2020 LET HI=INT(NM/256):LET LO=N
M-256*HI
2030 LET N=HI:GOSUB 1000:LET N$=
"= "+B$
2040 LET N=LO:GOSUB 1000:LET N$=
N$+" "+B$
2050 RETURN
3000 REM****HEX BYTE S/R******
3010 LET HB=INT(N/16):LET LB=N-H
B*16
3020 LET B$=X$(HB+1)+X$(LB+1)
3030 RETURN
4000 REM***HEX CONVERS S/R****
4010 IF NM<256 THEN LET N=NM:GOS
UB 3000:LET H$="=    "+B$:RETURN
4020 LET HI=INT(NM/256):LET LO=N
M-256*HI
4030 LET N=HI:GOSUB 3000:LET H$=
"= "+B$
4040 LET N=LO:GOSUB 3000:LET H$=
H$+" "+B$
4050 RETURN
```

Wenn Sie eine Zahl mit einem „+“ am Ende eingeben (beispielsweise 6435+), wird außer dem dezimalen und dem binären Wert auch die entsprechende Hexzahl angezeigt.

Fachwörter von A bis Z

Bandwidth = Bandbreite
Der Begriff „Bandbreite" wird meist in nachrichtentechnischem Zusammenhang gebraucht. Die Bandbreite eines Übertragungskanals (z. B. verdrillte Leitung, Telefonkabel, Laserstrahl oder Funkverbindung) ist der Frequenzbereich, der bei der Übermittlung ohne nennenswerte Signalverzerrung nutzbar ist. Das Telefon arbeitet beispielsweise mit dem Tonfrequenzbereich von 300-3400 Hz. Die Bandbreite beträgt 3,1 kHz. Das genügt für Sprachübertragung, aber nicht für HIFI-Musik.

Die Bandbreite der Verbindung bestimmt die maximale Datenrate beim Verkehr zwischen den einzelnen Geräten. Ein Fernsehkanal belegt einen Frequenzbereich, der für 3000 parallele Ferngespräche ausreichte. Während ein preisgünstiges Telefonmodem eine Datenrate von nur etwa 1200 Baud bewältigt, bietet der Computer die Möglichkeit, bei Verwendung des Kabelfernsehnetzes in Sekundenbruchteilen ein längeres Programm an einen anderen Benutzer übertragen.

Bank Switching = Bankauswahlverfahren
Jeder Microprozessor hat einen vorgegebenen Adreßbereich, der die Anzahl der identifizierbaren Speicherplätze bestimmt. 8-Bit-Zentraleinheiten wie die verbreiteten Bausteine Z80 oder 6502 können gewöhnlich einen Adreßbereich von 64 KByte (Speicherplätze 0 bis 65535) ansprechen. Die neueren 16-Bit-Rechner (auch der Sinclair QL) sind in der Lage, sehr viel mehr Speicherraum zu adressieren, zum Teil mehrere Megabyte.

Eine Speichererweiterung über 64 KByte hinaus ermöglicht bei einem 8-Bit-Prozessor nur das sogenannte „Bank Switching", eine gängige Programmiertechnik bei Rechnern wie dem Commodore 64 oder dem Atari. Obwohl dem Rechner eine festgelegte Speicherkapazität zur Verfügung steht, läßt sich mit Hilfe des Betriebssystems, durch ein internes Umschalten (Switching), der Speicherplatz doppelt nutzen. Ähnlich

Hier werden einzelne Fachausdrücke eingehend behandelt. Da bei der Kommunikation mit dem Computer meist die englische Sprache verwendet wird, werden hier zunächst die englischen Begriffe genannt, dann die deutsche Übersetzung. In den Gesamtindex werden sowohl deutsche als auch englische Stichwörter aufgenommen, damit Sie es leichter haben, das von Ihnen Gesuchte zu finden.

funktioniert auch das „paging", wobei der Rechner zum Beispiel bei der Bildwiedergabe veranlaßt wird, je nach Wunsch etwa den halben oder den doppelten Bereich des Bildschirm-RAMs für die Darstellung zu nutzen. Das Bank Switching können Sie sich so vorstellen: Die „memory map" des Computers ist eine senkrechte Liste. Zeilenweise werden nun RAM-„Streifen" hin- und hergeschoben, bis sich die benötigten zusätzlichen Speicherbereiche in der „memory map" befinden.

Bar Codes = Strichcodes
Strichcodes kennt wohl jeder – und sei es nur von Verpackungsetiketten.

Strichcodes werden nicht nur auf Preisschildern verwendet. Bei dieser Orgel von Casio sind die Noten auf dem Papier als Strichcodes verschlüsselt und können mit einem Lichtstift in den Speicher eingelesen werden.

Diese Form der Verschlüsselung von Digitalwerten ist mit Hilfe eines optischen Lesestiftes elektronisch lesbar. – Er wird einfach darüber hinweg geführt.

Strichcodes werden aber nicht nur bei Preis- und Produkt-Etiketten verwendet. Sie sind z. B. auch zur Noteneingabe bei Synthesizern (wie bei den Casio-Orgeln) zu gebrauchen, und Hewlett-Packard verwendete Strichcodes erstmals zum Programmeinlesen vom Papier in den Rechner.

Schwierigkeiten bereiten bei der Abtastung der Strichcodes die verschiedenen Geschwindigkeiten, in denen der Lesestift über die Codes geführt wird.

Base = Basis
Beim vertrauten Dezimalsystem wird meist vergessen, daß es ein „Stellenwertsystem" darstellt. Dessen „Basis" 10 ist jedoch für die Rechnertechnik weniger geeignet als die Basiszahlen 2 (Dual- oder Binärsystem) oder 16 (Hexadezimal-, kurz: Hex-System).

Die Basis eines Zahlen-Systems kann beliebig groß sein; bei der Zeitangabe in Minuten und Sekunden z. B. wird die Basis 60 benutzt (Sexagesimal-System). Wird die Basis 10 überschritten, müssen zur Ergänzung der Ziffern 0–9 neue Symbole gefunden werden – im Hex-System werden die „Ziffern" 10–15 durch die Buchstaben A–F dargestellt. Während sich im normalen Gebrauch das Dezimalsystem durchsetzen konnte, arbeiten Rechner im Dualsystem, da elektrisch zwei stabile Zustände wesentlich besser realisierbar sind.

Bildnachweise
533: Chris Stevens
534, 538, 546, 548, 556, 557, U3: Ian McKinnell
535: Chris Stevens, Ian McKinnell
539: Marcus Willy
540, 559: Kevin Jones
541: Steve Cross
547: Michael Brownlow, Kevin Jones
549: David Weeks
551: Sir John Tenniel
553: Marcus Wilson-Smith
555: Tony Duncan Smith

++ Vorschau +++ Vorschau +++ Vorschau +++

computer kurs Heft 21

Schatzsuche

In einem verlassenen Bergwerk stürzt sich „Willy" ins Abenteuer. Jede der 20 Höhlen gibt den Weg erst frei, wenn er vier Schlüssel gefunden hat . . .

Elektronische Schüsse

Der Computer errechnet die Position der Pistole beim Abschuß aus den Daten einer Fotozelle.

Scharfe Kiste

Keyboard, Maus, Monitor mit eingebautem Diskettenlaufwerk und Tragekoffer des Macintosh wiegen zusammen nur 12 Kilo.

+++ Greifarme +++ LOGO-Abenteuerspiel +++ Computer-Historie +++ Platine-Einbau beim ZX81 +++ BASIC +++ Kostenanalyse per Rechner +++