Einsteigen - Verstehen - Beherrschen

DM 3,80 öS 30 sfr 3,80

computer kurs

Drucker und Zeichner

Büro-Programme

Roboter lernen laufen

Acorn Electron und IBM PC

Funktion der CPU

Heft 19

Ein wöchentliches Sammelwerk

computer kurs

Heft 19

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.

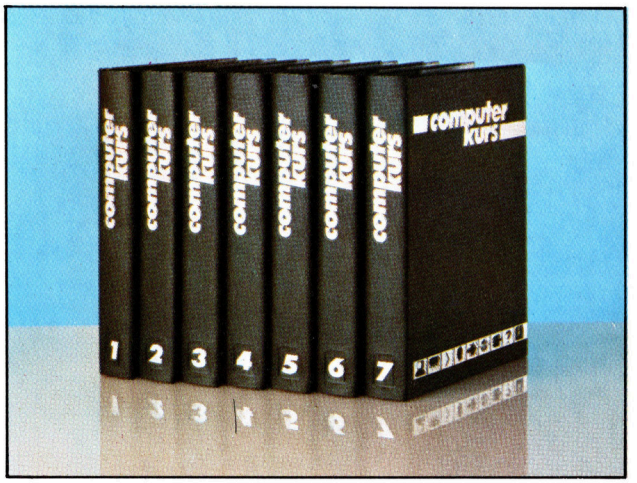
Schweiz: Der Sammelordner kostet sfr 15. Bitte wenden Sie sich an Ihren Kiosk; dort werden Sie jederzeit die gewünschten Exemplare erhalten.

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

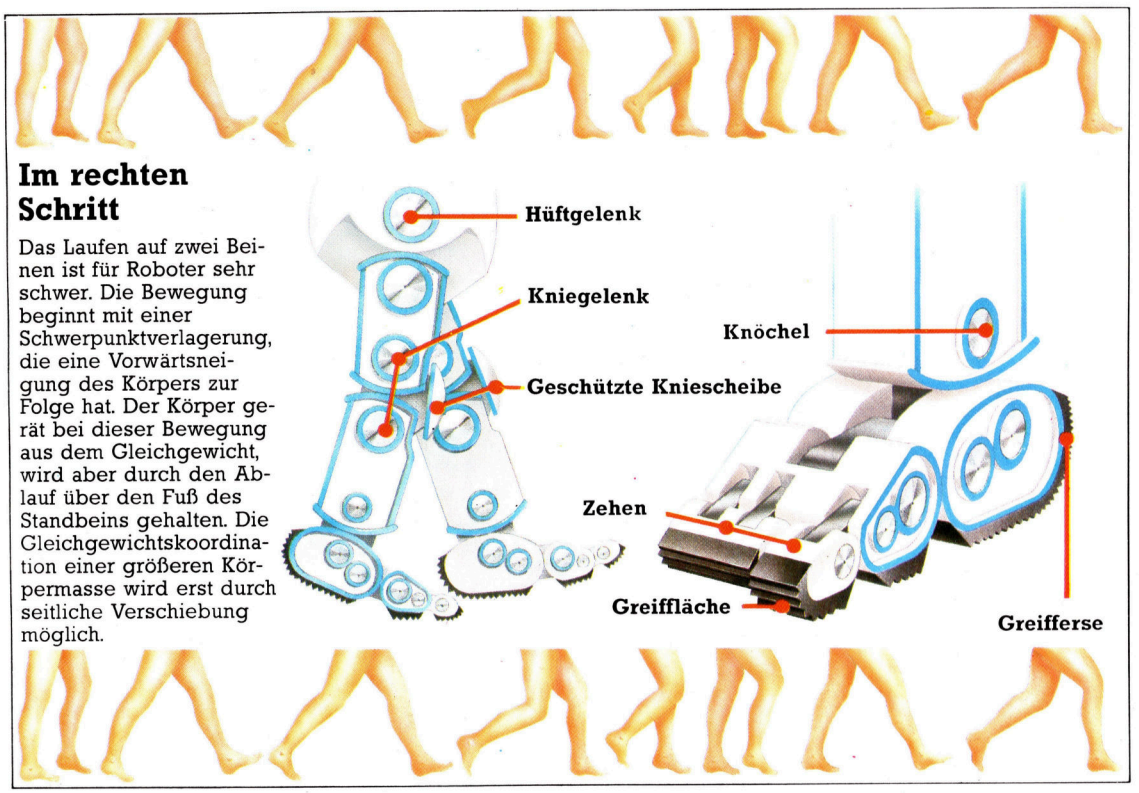
Schritt für Schritt

Die Roboter-Entwicklung von Science-fiction-Fantasiegebilden zu „metal collar"-Arbeitern an Fließbändern wurde bereits aufgezeigt. Nachstehend eine Darstellung der Kontroll- und Fortbewegungsmöglichkeiten von Robotern.

Lange bevor ein Kind die ersten Schritte macht, ist es imstande, Gegenstände zu greifen, und es kann seine Intelligenz auf vielerlei Art unter Beweis stellen. Laufen ist dagegen eine Fertigkeit, die gelernt sein will, bevor sie automatisch erfolgt.

Auch Roboter können „Gehen" lernen. Die dabei verwendeten Techniken unterscheiden sich jedoch wesentlich von dem Bewegungsablauf beim Menschen. Roboter können mit künstlichen Beinen ausgestattet sein, die – ähnlich wie beim menschlichen Gang – vor- und zurückschwingen. Viele Robotermodelle bewegen sich jedoch auf Rollen – um unerwünschte Rückwärtsbewegungen zu verhindern, sind diese mit einer Sperre ausgestattet. Ein solcher Roboter folgt beim „Laufen" einer vorprogrammierten Sequenz. Nachteil dieser Methode: Es fällt schwer, den Roboter zu steuern. Er bewegt sich lediglich vorwärts, und seine Bewegungen sind ungenau.

Eine bessere Lösung wäre es, den menschlichen Bewegungsablauf zu kopieren. Die Beine werden abwechselnd gehoben, statt die Gliedmaßen nur in begrenztem Bogen zu schwingen. Hauptproblem hierbei ist aber das Halten der Balance. Verschiedene Möglichkeiten wurden getestet: So die seitliche Neigung des Roboterkorpus oder gar die komplette Neigung des Rumpfes, zur Verlagerung des Schwerpunktes auf das Bein, auf dem das Gewicht lastet. Könnte ein solches System entwikkelt werden, gäbe es tatsächlich laufende Roboter. Theoretisch ließe sich ein Roboter konstruieren, der Treppen steigen oder gar Tee servieren könnte. In der Praxis aber sieht das anders aus, wenngleich ein treppensteigender Roboter im Bereich des Möglichen liegt. Das Problem dabei ist, daß der Roboter „wissen" müßte, wann er die jeweils oberste Stufe erreicht hat. Die Entwicklung einer solchen zusätzlichen Sensor-Einrichtung ist jedoch nur schwer zu verwirklichen. Alternativ dazu hat man Roboter konstruiert, die sich mittels Laufketten fortbewegen. Damit können sich Roboter auch auf unebenem Boden fortbewegen.



Im rechten Schritt

Das Laufen auf zwei Beinen ist für Roboter sehr schwer. Die Bewegung beginnt mit einer Schwerpunktverlagerung, die eine Vorwärtsneigung des Körpers zur Folge hat. Der Körper gerät bei dieser Bewegung aus dem Gleichgewicht, wird aber durch den Ablauf über den Fuß des Standbeins gehalten. Die Gleichgewichtskoordination einer größeren Körpermasse wird erst durch seitliche Verschiebung möglich.

Laufketten sind stabil und leicht anzutreiben, weisen aber zwei Nachteile auf. Da die meisten Roboter klein sind, müßten die Laufketten entsprechend dimensioniert werden. Größere Hindernisse wären deshalb schwer oder gar nicht zu überwinden. Ein Panzer bewältigt ob seiner Größe und seines Gewichtes fast jede Barriere. Befindet sich der Schwerpunkt des Panzers jedoch außerhalb des Kettenbereichs, kippt er um. Gleiches würde einem Roboter widerfahren, wenn er sich auf zu steilem Untergrund bewegte.

Problem: Steuerung

Ein weiterer Nachteil ist die ungenaue Steuerungsmöglichkeit von Laufketten. Eine Richtungsänderung erfolgt, indem die eine Laufkette gebremst wird, während sich die andere weiterbewegt. Ein Roboter (zum Beispiel der Panzer in der untenstehenden Abbildung) dreht sich also innerhalb eines Bogens. Im Bewegungsablauf kann es vorkommen, daß eine Laufkette rutscht. Das hat zur Folge, daß die gewünschte Positionsänderung nicht erreicht wird. Steuert ein Mensch den Panzer, kann der gewünschte Zielort erreicht werden, indem die Lenkung korrigiert wird. Die Richtungskorrektur bei einem Roboter ist schwieriger.

Um einen Roboter genau zu steuern, sind exakte Anweisungen erforderlich, die jede Kursänderung ausschließen. Dies ist nur möglich, wenn sich der Roboter auf Rädern bewegt. Das hat den Vorteil einfacher Konstruktion, denn Räder ermöglichen einen gleichmäßigen Bewegungsablauf.

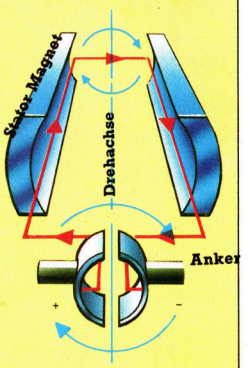
Geht man davon aus, daß die Roboterfortbewegung am besten auf Rädern stattfindet, bleibt das Problem der Kontrolle der Bewegungsrichtung. Als Beispiel sei der Aufziehmotor eines Spielzeugautos angeführt, das auf Rädern läuft, aber kein echter Roboter ist und nie „weiß", wo es sich befindet. Erforderlich wäre ein Koordinatensystem, mit dessen Hilfe die Bestimmung der Position eines Objektes auf einer beliebigen Oberfläche möglich ist. Das allgemein verbreitete System zur Ortsbestimmung ist das Cartesische System. Damit lassen sich Position wie Bewegung genau spezifizieren. Auf dieser Grundlage bleibt nur die Entwicklung eines Gerätes, mit dessen Hilfe sich der Roboter innerhalb des vorgegebenen Referenzrahmens bewegen kann.

Man hat sich nach Versuchen mit hydraulischem und pneumatischem Antrieb für Elektromotoren entschieden, um Roboter fortzubewegen. Ein einfacher Elektromotor macht Bewegung und in bescheidenem Umfang Richtungssteuerung möglich. Eine genaue Kontrolle kann aber nicht stattfinden, da ein Elektromotor um 180 Grad weiterdreht, bevor er zum Stillstand kommt. Der Drehwinkel ist häufig noch größer.

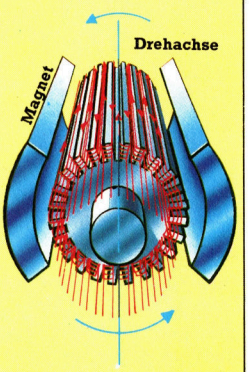
Deshalb werden für Roboter bevorzugt Schrittmotoren verwendet. Schrittmotoren enthalten mehrere Induktionsspulen, die sehr kleine, dabei äußerst exakte Drehbewegungen ermöglichen – unabhängig von der Vielzahl der angebotenen Typen. Anders gesagt: Es gibt kaum Über- bzw. Unterdrehungen.

Roboterbewegungen lassen sich also mit Hilfe von Schrittmotoren und cartesischem Koordinatensystem relativ präzise steuern. Doch um einen Roboter an Hindernissen vorbeizuführen, ihn kurzfristig reagieren zu lassen und eine Anpassung an die jeweilige Umgebung zu ermöglichen, ist mehr erforderlich. Diese Fähigkeiten werden später erläutert.

Schreiten nach Maß

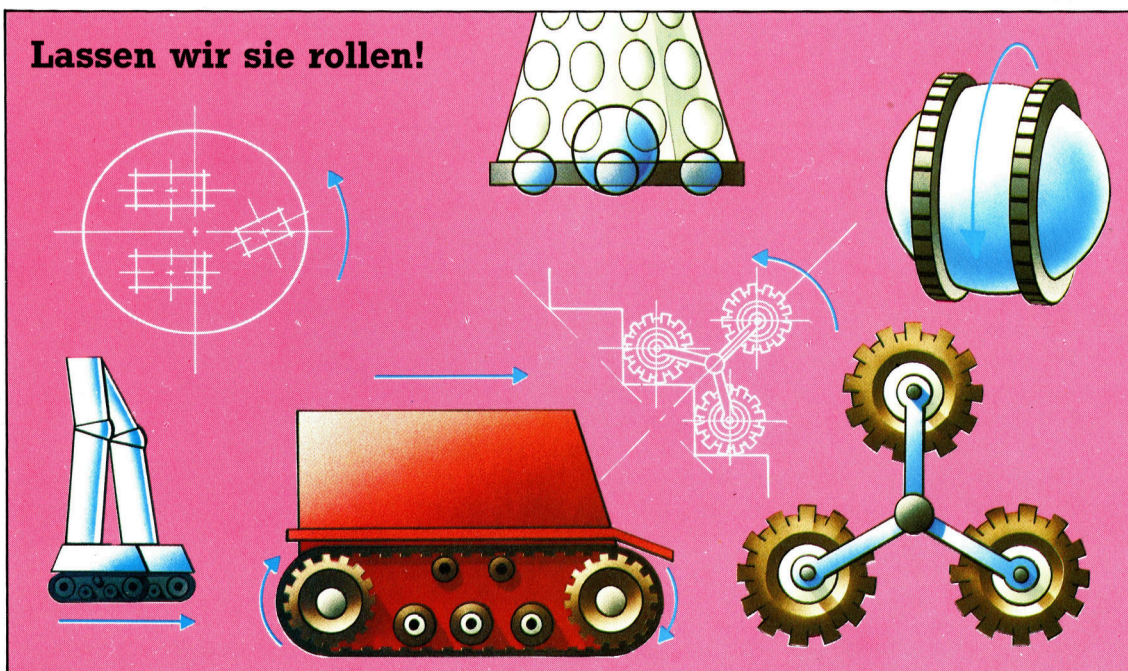


Schrittmotoren enthalten viele Spulen. Der Stromfluß ermöglicht genau kontrollierbare Drehungen.



Bei einem einfachen Elektromotor wird ein dem des Stators entgegengesetztes Magnetfeld erzeugt. Dies bewirkt die Drehung.

Lassen wir sie rollen!



Roboter müssen sich bewegen – nicht nur auf ebenem Boden. Hier sind einige der Fortbewegungsmöglichkeiten dargestellt. Laufketten haben den Nachteil, nicht exakt steuerbar zu sein, erlauben aber eine ständige Bewegung, ohne daß ein Heben der Beine erforderlich ist. Damit entfallen die Gleichgewichtsprobleme. Mit Laufketten ausgestattete Roboter werden zum Beispiel bei der Bombenentschärfung und in der Raumforschung eingesetzt. Dreiaxiale Räder sind die derzeit einzige Möglichkeit, einen Roboter Treppen steigen zu lassen. Eine mit Stabilisatoren ausgestattete Kugel ist leicht steuerbar, reagiert aber besonders empfindlich bei unebener Oberfläche.

Das Design und die Auslegung des IBM PC zeigen die große Erfahrung der Firma auf dem Gebiet der Computer und Büromaschinen: Der PC ist solide und nach ergonomischen Gesichtspunkten konstruiert. Die drei Hauptgeräte – Tastatur, Prozessorgehäuse und Monitor – sind frei beweglich und können beliebig angeordnet werden. Die Standardausführung der Maschine enthält einen monochromen Bildschirm; ein Farbmonitor wird ebenfalls angeboten.

Der IBM PC

Branchenexperten behaupten, daß die Microcomputer-Revolution erst durch die Entwicklung des IBM PC einsetzte. Obwohl IBM als der weltgrößte Produzent von Büromaschinen und Computern seinen Microcomputer erst im Jahre 1981 herausbrachte, veränderte sich dadurch der Markt vollständig.

IBM hat in der Groß-EDV und im Bereich der Minicomputer nicht den Ruf, sehr fortschrittlich zu sein. Die solide Bauweise der Geräte ist jedoch bekannt, und auch der IBM PC entspricht diesem Standard. Wie fast alle IBM-Geräte ist er teurer als seine Konkurrenten. Der IBM PC ist mindestens ebensooft nachgebaut und kopiert worden wie der Apple, jedoch in einem wesentlich kürzeren Zeitraum.

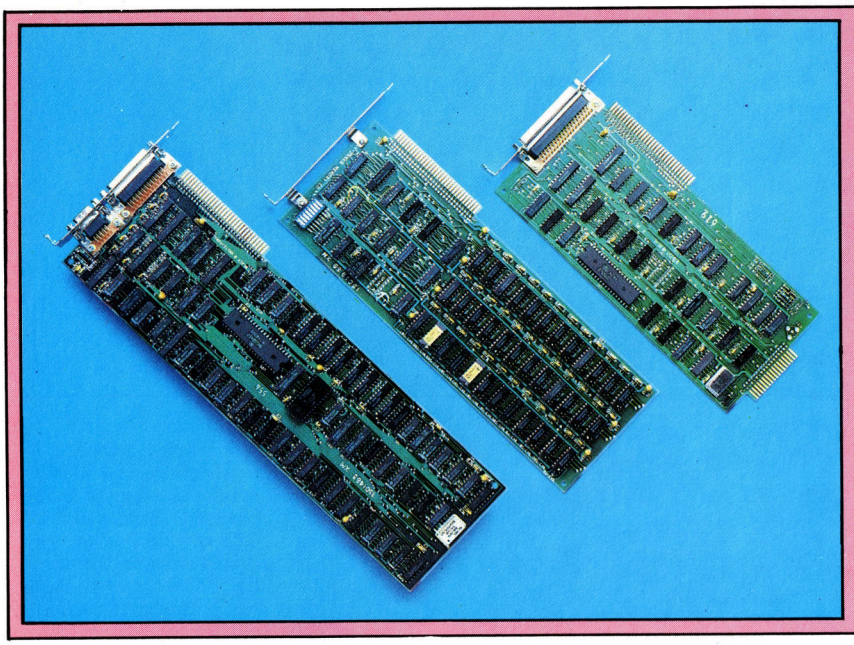
IBM erklärt den hohen Preis des PC mit der Breite der Wartungsmöglichkeiten. Diese Unterstützung besteht in der Tat – wenn Sie bereit sind, circa 11,2 Prozent des Kaufpreises pro Jahr für einen Wartungsvertrag zu zahlen. Andererseits sind Wartungsverträge der meisten anderen Firmen mindestens zwei Prozent teurer, wobei nur wenige während der Reparaturzeit Ersatzgeräte zur Verfügung stellen. All diese Vorteile lassen die Maschine einer so großen Firma wie IBM attraktiv erscheinen.

Doch die in den PC eingebaute Technik ist nicht weltbewegend. Er verfügt über einen 8088-Prozessor, der zwar als 16-Bit-CPU beschrieben wird, aus Platzgründen aber nur einen 8-Bit-Datenbus besitzt. Die Verarbeitungsgeschwindigkeit ist nicht besonders hoch, und die internen Rechenzeiten sind etwa 25 Prozent schneller als bei 8-Bit-Maschinen.

Erweiterungen sind sinnvoll

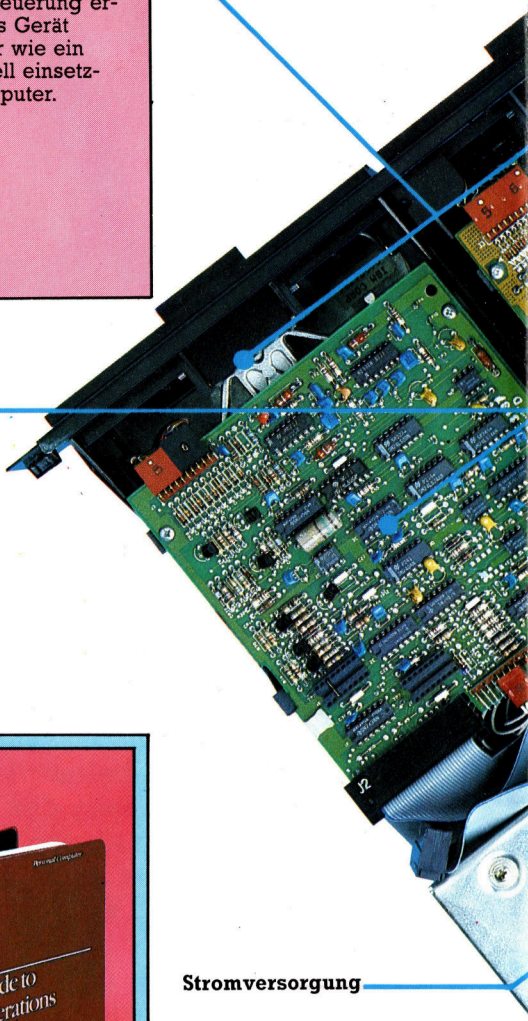
Die Standardausführung des Gerätes besitzt nur einen kleinen Arbeitsspeicher, der für komplexe Programme nicht ausreicht und erweitert werden muß, bevor der PC vernünftig eingesetzt werden kann. Es gibt nicht genug Ein-/Ausgabemöglichkeiten. Eine Funktionskarte muß meist dazugekauft werden.

Die Grafikfähigkeiten des PC sind beeindruckend, werden aber nicht standardmäßig



Erweiterungsplatinen
Die Standardversion des IBM PC ist kaum einen Vergleich mit langjährig im Markt stehenden Maschinen wert. Mit Zusatzplatinen für (von links nach rechts) Farbgrafik, Speichererweiterung und einer hochentwickelten Ein- und Ausgabesteuerung erscheint das Gerät schon eher wie ein kommerziell einsetzbarer Computer.

Diskettenstationen
Standardmäßig ist die Maschine nur mit einem Laufwerk für die einseitige Beschriftung von Disketten mit einfacher Schreibdichte ausgerüstet. Sie kann aber nachträglich auf doppelseitiges Format mit doppelter Schreibdichte aufgerüstet werden.



Diskettensteuerung

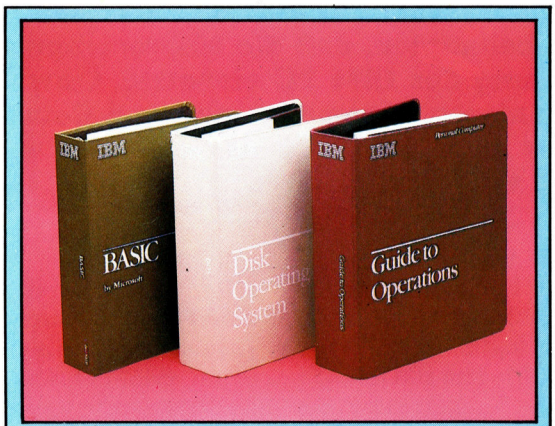
Stromversorgung

mitgeliefert und stehen nur mit einer zusätzlichen Grafikkarte zur Verfügung. Es gibt sie in zwei Ausführungen – schwarz/weiß und Farbe. Sie sind mit zusätzlicher Speicherkapazität bestückt und enthalten elektronische Bauteile zur Erzeugung des Videosignals. Normalerweise werden die Karten vom Hauptprozessor gesteuert. Es gibt aber schon Versionen, die über einen eigenen Videoprozessor verfügen, der CPU damit die Aufgabe abnehmen, den Bildschirm ständig aktualisieren zu müssen, und dadurch die Geschwindigkeit des Gerätes erhöhen. Leider sind diese Karten nicht gerade preiswert.

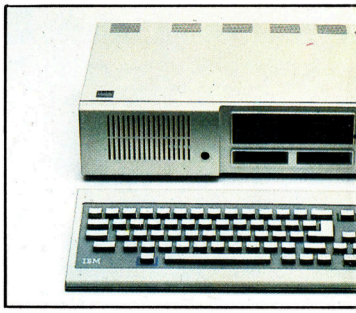
Der PC hat fünf Steckleisten für Erweiterungen, deren Verwendung wegen der geringen Anzahl gut geplant werden muß. Fast alle Karten sind daher umfangreich, komplex, fähig, viele Arbeiten (oft auch gleichzeitig) auszuführen und – teuer. Da der IBM PC ohne Erweiterungen nur wenig brauchbar ist, sollten die Kosten für Karten beim Kauf mit in Betracht gezogen werden.

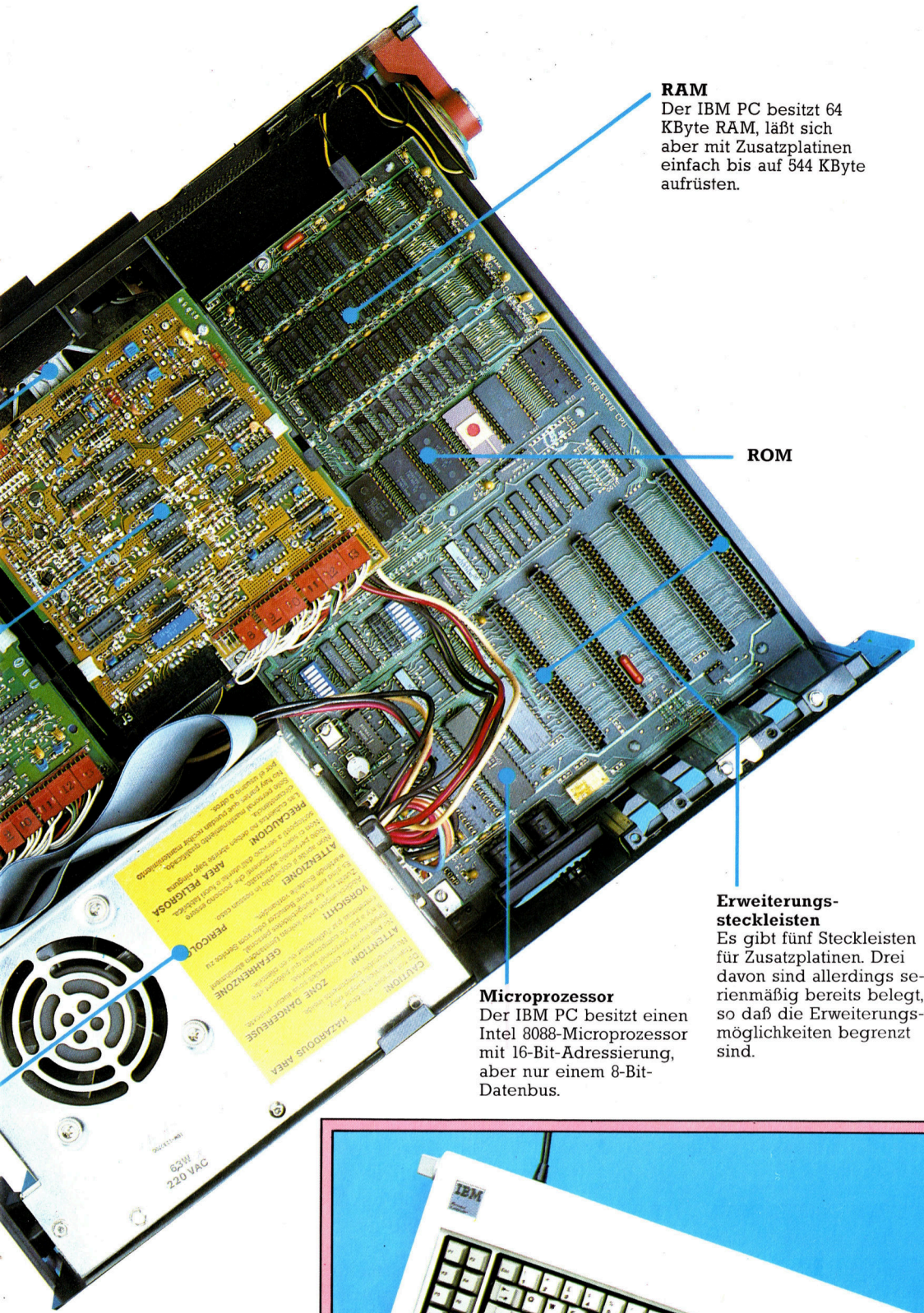
Natürlich werden Modelle angeboten, bei denen viele Erweiterungen bereits standardmäßig eingebaut sind, wie z. B. die XT-Version mit Festplatte. Die Preise dieser Geräte reichen aber schon bis an den Lisa von Apple heran.

Dem Trend der Zeit folgend hat IBM den PC auch als tragbares Modell herausgebracht, wobei ein Gewicht von 14 kg das Wort „tragbar" allerdings etwas strapaziert. In diesem Gerät wurden die beiden Standard-Diskettenlaufwerke durch ein doppelseitiges Laufwerk ersetzt, wobei in der freien Öffnung ein 9-Inch-Monitor mit bernsteinfarbenem Bildschirm seinen Platz fand. Eine leichtere und kleinere Tastatur läßt sich in die Vorderseite der Maschine einrasten, die außerdem noch mit einem neuen Gehäuse versehen wurde.



Software für den IBM PC
Einer der Hauptgründe für den Kauf eines IBM PC – und auch die Hauptursache für die vielen Kopien und Nachbauten anderer Hersteller auf der ganzen Welt – ist die breite Palette kommerziell einsetzbarer Software. Der PC hat das Betriebssystem PC-DOS (Disk Operating System), das von der Firma Microsoft auf der Grundlage von CP/M entwickelt wurde. Die Liste der kommerziellen Software schließt alle wichtigen für Microcomputer verfügbaren Programme ein: Textverarbeitungssysteme, Kalkulationsprogramme, Datenbanken und Managementpakete. Da der IBM PC in den Vereinigten Staaten eher als Heimcomputer angesehen wird, gibt es auch eine große Anzahl von Spielen amerikanischer Softwarehäuser.





RAM
Der IBM PC besitzt 64 KByte RAM, läßt sich aber mit Zusatzplatinen einfach bis auf 544 KByte aufrüsten.

ROM

Erweiterungs-steckleisten
Es gibt fünf Steckleisten für Zusatzplatinen. Drei davon sind allerdings serienmäßig bereits belegt, so daß die Erweiterungsmöglichkeiten begrenzt sind.

Microprozessor
Der IBM PC besitzt einen Intel 8088-Microprozessor mit 16-Bit-Adressierung, aber nur einem 8-Bit-Datenbus.

IBM PC

PREIS
ca. 7500 Mark

ABMESSUNGEN
142x500x410 mm

ZENTRALEINHEIT
Intel 8088

SPEICHER-KAPAZITÄT
64 K RAM, auf 576 K erweiterbar; 40 K ROM

TAKTFREQUENZ
4,77 MHz

BILDSCHIRM-DARSTELLUNG
25 Zeilen mit je 80 Zeichen

SCHNITTSTELLEN
Centronics parallel und fünf Steckleisten für Erweiterungen

PROGRAMMIER-SPRACHEN
BASIC und alle Sprachen, die unter PC-DOS laufen

TASTATUR
79 Schreibmaschinentasten

HANDBÜCHER
Die Handbücher sind auf dem von IBM und den Softwareherstellern zu erwartenden hohen Niveau. Die Qualität der Handbücher unabhängiger Softwarefirmen ist unterschiedlich.

STÄRKEN
Der Standard-PC ist verläßlich und brauchbar. Er kann schrittweise erweitert und zu großen Kapazitäten ausgebaut werden.

Der PC Junior von IBM erhielt während seiner Entwicklungsphase den Spitznamen „Peanut" (Erdnuß) und ist eine kleinere Version der Standardmaschine. Die vielleicht interessanteste Neuerung daran ist der Einsatz einer Infrarotverbindung zwischen Prozessor und Eingabetastatur.

Tastatur
Die freibewegliche Tastatur des PC läßt – wie von dem größten Hersteller von Schreibmaschinen und anderen Tastengeräten zu erwarten – kaum Wünsche offen. Sie ist außerordentlich flach, im Winkel verstellbar und hat fünf Reihen von Profiltasten, deren äußere Reihen hochgezogen sind.

Testlauf

Bevor Daten-Files im Programm eingesetzt werden, sollten Sie die einzelnen Funktionen anhand einer Rohstruktur überprüfen.

Am Ende des letzten Teiles unseres BASIC-Kurses haben wir Sie mit der Lösung eines Problems zurückgelassen: Wie kann man das Programm dazu veranlassen, eine Datei einzulesen (von Cassette oder Diskette), die beim ersten Programmstart noch nicht existiert? Die erste Aufgabe des Programms ist es, eine Datei einzulesen und die Daten Bereichen oder Variablen zuzuweisen. Wenn man jedoch festlegt, immer zuerst etwas auf die Datei zu schreiben, muß man bei der Programmierung sehr vorsichtig vorgehen, um nicht alle Daten innerhalb der Datei zu verlieren.

Glücklicherweise gibt es eine ganz einfache Lösung. Viele kommerzielle Software-Pakete beinhalten ein „Install"- oder „Set Up"-Programm, das man vor dem ersten Programmlauf starten muß. Das ist genau die Lösung, die wir anzustreben versuchen. Solche Programme gestatten dem Anwender meistens noch in einem eingeschränkten Umfang spezielle Anpassungen des Programms (beispielsweise ob der verwendete Drucker ein Epson oder ein Brother ist, parallel oder seriell usw.).

Um *LSINDT* (die Routine zum Einlesen der Datei und Zuordnen der Daten in die Speicherbereiche) ausführen zu können, schreiben wir ein sehr einfaches „Set Up"-Programm, das nicht mehr macht, als eine Datei zu öffnen und einen Übergangswert hineinzuschreiben. Wir werden einen Wert wählen, der keinen Einfluß auf das Programm hat und kein gültiges Adreßbuch-Verzeichnis ist. Für diesen Zweck ist eine Zeichenkette wie z. B. @ERST ideal, da kein Name und keine Adresse mit einer solchen Zeichenkette beginnen wird. Die Routine *LSINDT* muß etwas modifiziert werden, sodaß sie vor dem Einlesen der Daten zuerst diesen Wert überprüft.

```
10 REM DIESES PROGRAMM KREIERT
   EINE
20 REM DATEN-DATEI, DIE VOM
   ADRESSBUCH-PROGRAMM
   VERWENDET WIRD
30 REM ES SCHREIBT EIN TEST-
   VERZEICHNIS, DAS VON *LSINDT*
40 REM VERWENDET WERDEN KANN
50 REM
60 REM
70 OPEN "0",#1, "ADBK.DAT"
80 PRINT #1, "@ERST"
90 CLOSE #1
100 END
```

Wie bereits in vorangegangenen Teilen des BASIC-Kurses erwähnt, unterscheiden sich zwar die Details des Lesens und Schreibens zwischen den BASIC-Versionen, doch das Grundprinzip ist identisch. Als erstes muß die Datei geöffnet werden, bevor Daten von ihr gelesen oder in ihr abgelegt werden können. Als nächstes muß die Richtung des Datenflusses festgelegt werden – entweder IN (=Einlesen) oder OUT (= Ausschreiben). Danach muß der Datei eine „Kanalnummer" zugeordnet werden. Dadurch ist es möglich, mehr als nur eine Datei zur gleichen Zeit geöffnet und in Benutzung zu haben (in unserem Beispiel verwenden wir nur eine einzelne Datei). Als letzter Schritt ist der Name der Datei, die Sie verwenden wollen, anzugeben.

Datei öffnen und schließen

Zeile 70 des oben gezeigten Programms ist in Microsoft-BASIC geschrieben und prinzipiell den OPEN-Anweisungen der meisten BASIC-Versionen ähnlich. OPEN sagt aus, daß eine Datei geöffnet werden soll, und „0" besagt, daß Daten ausgegeben werden. #1 ist die Zahl, die wir der Datei für diese Operation zugeordnet haben; eine andere Datei-Nummer könnte später vergeben werden. „ADBK.DAT" ist der Name, den wir für die Datei verwenden wollen.

Zeile 80 schreibt einfach ein einzelnes Verzeichnis in die Datei. Zeile 90 schließt diese (CLOSE). Dateien können geöffnet bleiben, solange sie innerhalb des Programms benötigt werden. Aber geöffnete Dateien sind relativ ungeschützt und sollten daher geschlossen werden, sobald dies möglich ist, um die darin befindlichen Daten zu schützen.

Es bestehen einige Unklarheiten darüber, wie die Begriffe Verzeichnis und Datei in einem Computer verwendet werden. Diese Verwirrung wird größer, wenn wir auf der einen Seite von Stammdateien und andererseits von Daten-Dateien sprechen. In einer Stammdatei besteht die Datei aus einer systematisch geordneten Ansammlung aufeinander bezogener Informationen.

Eine sequentielle Daten-Datei auf einer Diskette oder Cassette kümmert sich jedoch nicht darum, ob die darin enthaltene Information von einem Programm organisiert oder verwendet wird. Daten-Dateien enthalten lediglich eine Reihe von Datensätzen, und jeder individuelle Datensatz wird Verzeichnis genannt.

Wenn das Programm gestartet wird, kann es nicht feststellen, ob sich bereits echte Daten in dieser Datei befinden oder nicht. Die erste Tätigkeit von *LSINDT* ist das Öffnen der "ADBK.DAT"-Datei, sowie das Einlesen des ersten Verzeichnisses (oder Datensatzes) in die String-Variable TEST$. Bevor irgendein anderes Verzeichnis eingelesen wird, wird TEST$ überprüft, ob er den String @ERST enthält. Wenn TEST$ diesen @ERST-String enthält, „weiß" das Programm, daß sich innerhalb dieser Datei keine gültigen Daten befinden. Demzufolge kann diese Datei geschlossen werden, und das restliche Programm kann fortfahren.

Wenn andererseits der Wert von TEST$ nicht @ERST ist, muß das Programm davon ausgehen, daß sich innerhalb dieser Datei gültige Daten befinden, und beginnt dann damit, diese Daten den zugehörigen Bereichen zuzuordnen. Nachfolgend sehen Sie die modifizierte *LSINDT*-Unterroutine:

```
1400 REM *LSINDT*-UNTERROUTINE
1410 OPEN "I", #1, "ADBK.DAT"
1420 INPUT #1, TEST$
1430 IF TEST$ = "@ERST" THEN GOTO
     1530: 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 REM PLATZ ZUM AUFRUF DER
     *GROSS*-UNTERROUTINE
1520 NEXT L
1530 CLOSE #1
1540 RETURN
```

Zeile 1420 ordnet ein einzelnes Verzeichnis aus der ADBK.DAT-Datei der Variablen TEST$ zu. Die nächste Zeile überprüft dann diese Zuordnung, um zu sehen, ob der Wert @ERST ist. Trifft dies zu, wird ein GOTO verwendet, um zu der Zeile zu springen, die die Datei schließt (Zeile 1530). Die Unterroutine kehrt dann (mit RETURN) wieder ins Hauptprogramm zurück.

Aufruf von *SETFLG*

In der Annahme, daß sich innerhalb dieser Datei keine gültigen Daten mehr befinden, wird die Programm-Kontrolle wieder an *INITIL* abgegeben, die dann *SETFLG* aufruft. Alles, was diese Routine in diesem Moment macht, ist, den Wert von GROSS auf 1 zu setzen, wenn TEST$ gleich @ERST ist. Der Code für *SETFLG* wird nachfolgend gezeigt.

```
1600 REM *SETFLG*
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
```

SETFLG kehrt dann zu *INITIL* zurück, das wiederum mit RETURN ins Hauptprogramm zurückspringt. *HAUPTPG* ruft anschließend *BGRUES* auf, mit dem die Begrüßungsmeldung angezeigt wird.

Die nächste Routine, die vom Hauptprogramm aufgerufen wird, ist *AUWAHL*. Eine kleine Modifizierung der bereits gezeigten *AUWAHL*-Unterroutine bietet einen Weg, um den Anwender aufzufordern, ein Verzeichnis einzufügen, sobald das Programm zum ersten Mal gestartet wird.

```
3500 REM *AUWAHL*-UNTERROUTINE
3510 REM
3520 IF TEST$="@ERST" THEN GOSUB 3860
3530 IF TEST$="@ERST" THEN RETURN
3540 REM "CHMENU"
3550 PRINT CHR$(12)
3560 PRINT "WOLLEN SIE"
3570 PRINT
3580 PRINT
3590 PRINT
3600 PRINT "1. VERZEICHNIS DURCH
     NAMEN SUCHEN"
3610 PRINT "2. NAMEN DURCH TEIL EINES
     NAMENS SUCHEN"
3620 PRINT "3. VERZEICHNISSE NACH
     STADTANGABEN SUCHEN"
3630 PRINT "4. VERZEICHNISLISTE DURCH
     INITIALEN"
3640 PRINT "5. LISTE ALLER VERZEICH-
     NISSE"
3650 PRINT "6. HINZUFUEGEN EINER
     ADRESSE"
3660 PRINT "7. AENDERN EINER ADRESSE"
3670 PRINT "8. LOESCHEN EINER ADRESSE"
3680 PRINT "9. PROGRAMM BEENDEN UND
     DATEN SPEICHERN"
3690 PRINT
3700 PRINT
3710 REM "INWAHL"
3720 REM
3730 LET L=0
3740 LET I=0
3750 FOR L=0 TO 1
3760 PRINT "WAEHLEN SIE (1—9)"
3770 FOR I=1 TO 1
3780 LET A$=INKEY$
3790 IF A$="" "THEN I=0
```

```
3800 NEXT I
3810 LET WAHL=VAL(A$)
3820 IF WAHL <1 THEN L=0 ELSE L=1
3830 IF WAHL>9 THEN L=0
3840 NEXT L
3850 RETURN
```

Zwei Zeilen wurden eingefügt. Die erste überprüft TEST$. Diese Variable enthält immer noch den Wert, der innerhalb der *LSINDT*-Routine eingelesen wurde. Wenn dieser Wert @ERST ist, wissen wir, daß sich in dieser Datei keine gültigen Daten befinden. Aus diesem Grund ist die einzig mögliche Option ADDVER, die Nummer 6. Sobald der Test abgeschlossen ist, wird die Kontrolle an *ERSTLA* übergeben, eine Routine, die eine entsprechende Meldung anzeigt, und die Variable WAHL auf 6 setzt. Wenn die Unterroutine in Zeile 3530 zurückkehrt, wird TEST$ erneut überprüft, und die Unterroutine kehrt zurück zum Hauptprogramm, wobei sie den Rest der AUWAHL-Unterroutine überspringt.

Die *ERSTLA*-Unterroutine ist einfach und klar: Der Bildschirm wird gelöscht, und eine Meldung erscheint, die den Anwender darüber informiert, daß ein Verzeichnis eingegeben werden soll. Zeile 3870 setzt WAHL auf 6, so daß die *ADDVER*-Routine automatisch ausgeführt wird, sobald die Programm-Kontrolle an *AUSFUH* zurückgegeben wurde. Nachfolgend sehen Sie den Code für *ERSTLA*:

```
3860 REM *ERSTLA*-UNTERROUTINE
     (STELLT MELDUNG DAR)
3870 LET WAHL=6
3880 PRINT CHR$(12): REM LOESCHT
     BILDSCHIRM
3890 PRINT
3900 PRINT TAB(8); "ES SIND KEINE
     VERZEICHNISSE"
3910 PRINT TAB(8); "IN DER DATEI.
     SIE MUESSEN"
3920 PRINT TAB(8); "MIT DER EINGABE
     EINES
     VERZEICHNISSES BEGINNEN"
3930 PRINT
3940 PRINT TAB(5); "(LEERTASTE UM
     FORTZUFAHREN)"
3950 FOR B=1 TO 1
3960 IF INKEY$ <>" " THEN B=0
3970 NEXT B
3980 PRINT CHR$(12): REM LOESCHT
     BILDSCHIRM
3990 RETURN
```

Die bereits zuvor gezeigte *ADDVER*-Unterroutine beinhaltet zwei kleine Veränderungen gegenüber der gerade beschriebenen Form. Nachdem die Felder als Elemente in die einzelnen String-Arrays eingegeben wurden, wird die Variable GROSS erhöht, und TEST$ wird auf einen Null-String gesetzt (siehe Zeilen 10090 und 10100).

Zeile 10090 erhöht GROSS, so daß der Wert jetzt 2 ist. Wenn *ADDVER* nun wieder ausgeführt wird, werden die Daten in das zweite Element jedes Bereiches eingesetzt. Zuletzt setzt *ADDVER* TEST$ in Zeile 10100 auf " ". Das wird gemacht, weil jetzt ein Verzeichnis eingegeben wurde. Wenn *AUWAHL* wieder ausgeführt wird, was gemacht werden muß, um die Daten abzuspeichern und das Programm zu beenden, ist es nicht notwendig, ein neues Verzeichnis einzugeben. Wenn TEST$ nicht gelöscht würde, würde sich das Programm in einer Endlos-Schleife verfangen, und die einzige Möglichkeit, um diese Schleife zu stoppen, wäre das Drücken der RESET-Taste oder das Abschalten des Computers – wobei alle Daten verlorengehen würden.

Was passiert mit GROSS?

Durch das Setzen von TEST$ auf einen Null-String schlagen die Tests in Zeile 3520 und 3530 von *AUWAHL* fehl und ermöglichen eine Anzeige des Auswahlmenüs. Was jetzt mit GROSS passiert, ist abhängig davon, welche Routine ausgeführt wird. Bis jetzt haben wir nur sichergestellt, daß GROSS gleich 1 ist, wenn sich in der Datei keine gültigen Daten befinden, und daß dieser Wert jedesmal um 1 erhöht wird, wenn ein Verzeichnis eingegeben wurde. Aber was würde passieren, wenn sich in dieser Datei eine Anzahl gültiger Daten befunden hätte? Um diese Frage beantworten zu können, müssen wir uns nochmals *LSINDT* ansehen.

Zeile 1420 liest den ersten Datensatz in TEST$ ein. Wenn dieser nicht @ERST entspricht, wird angenommen, daß es sich um gültige Daten handelt. Die Verzeichnisse innerhalb einer Datei bleiben immer in derselben Reihenfolge: NAMFLD, MODFLD, STRFLD, STDFLD, STAFLD, TELFLD, INDFLD, NAMFLD, MODFLD usw. Wenn das erste gelesene Verzeichnis gültige Daten enthält, wird es dem ersten Element des NAMFLD-Bereiches zugeordnet. Daher überträgt Zeile 1440 seine Daten von TEST$ in NAMFLD(1). Die nächsten beiden Zeilen füllen die ersten Elemente der anderen fünf Bereiche. Wir wissen jetzt, daß wir mindestens ein komplettes Verzeichnis erstellt haben, aus diesem Grund wird GROSS auf 2 erhöht. Dieser Wert muß um 1 gößer sein als die Anzahl der gültigen Verzeichnisse, die in die Bereiche eingelesen wurden.

Dann liest eine Schleife von 2 bis 50 die Verzeichnisse in alle sechs Bereiche ein und erhöht dabei jeweils die Zählvariable L. Wir haben bereits entschieden, daß das Programm darauf beschränkt sein soll, nur Dateien mit 50 Namen und Adressen zu bearbeiten. Und die DIM-Befehle in der *CREARR*-Unterroutine

BASIC-Dialekte

SPECTRUM

Da der Spectrum die Möglichkeit bietet, ganze Bereiche mit dem SAVE-DATA-Befehl zu speichern oder einzuladen, wird die *RDINFL*-Unterroutine völlig unterschiedlich eingelesen. Wenn wir in der nächsten Ausgabe damit anfangen, Daten einzuschreiben, werden wir eine vollständige Version der Unterroutinen für diese Maschine veröffentlichen. In der Zwischenzeit können sich Spectrum-Besitzer beispielsweise mit dem Problem beschäftigen, wie eine Test-Datei mit @ERST erstellt werden kann, oder herausfinden, wie viele gültige Eingaben sich in diesem Bereich befinden, wenn die Datei eingelesen wird.

INKEY$

Siehe vorangegangene BASIC-Dialekte.

OPEN CLOSE

Siehe vorangegangene BASIC-Dialekte.

haben hierfür bereits Platz freigehalten. Wenn Sie jedoch das erste Mal dieses Programm starten, haben Sie den Nachteil, eine Datei vor sich zu haben, für die 50 Eingaben benötigt werden. Deshalb brauchen wir eine Routine, die die Variable GROSS schrittweise erhöhen und die Einlese-Schleife beenden kann.

Aus diesem Grund haben wir die Zeile 1510 eingefügt, um eine *GROSS*-Unterroutine aufrufen zu können, die später in diesem Kurs entwickelt wird. Es gibt drei Möglichkeiten, um dieses Problem zu beseitigen. Als erstes könnten wir, wenn wir die Daten auf Band schreiben, arrangieren, daß zuerst die Variable GROSS geschrieben wird. Die Unterroutine *LSINDT* könnte dann so modifiziert werden, daß GROSS eingelesen wird, bevor eine Schleife FOR L=1 TO GROSS gesetzt wird, die die Verzeichnisse einliest. Die zweite und bessere Möglichkeit wäre ein Verfahren, das dann ausgeführt wird, wenn alle Verzeichnisse geschrieben wurden. Dabei sollte ein spezielles Flag an das Ende angesetzt werden.

Die dritte Methode wäre, die EOF (End OF File)-Funktion zu verwenden, die von einigen BASIC-Versionen angeboten wird und in Wirklichkeit eine automatisierte Version der zweiten Methode ist. Diese Dialekte enthalten ein EOF-Flag, das normalerweise auf Null gesetzt ist (für FALSE = FALSCH), aber auch den Wert 1 (für TRUE = WAHR) annehmen kann, sobald das Ende der Datei erreicht wurde.

Auf einigen Geräten stellt sich das EOF-Flag als ein einzelnes Bit dar, das mit dem PEEK-Befehl zugänglich gemacht wird. Um herauszufinden, ob Ihr Computer über ein EOF-Flag verfügt, sollten Sie im Bedienungshandbuch des Computers nachlesen. Da es hier große Unterschiede zwischen den einzelnen Geräten gibt, wollen wir das EOF-Flag hier nicht verwenden. Aber als kleines Beispiel könnten Sie die *LSINDT*-Unterroutine modifizieren, um durchzutesten, wie Sie eine Datei mit weniger als 50 Eingaben bearbeiten können.

```
   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 END
 1000 REM *INITIL*-UNTERROUTINE
 1010 GOSUB 1100: REM *CREARR* (DEFINIERE BEREICHE)-UNTERROUTINE-
 1020 GOSUB 1400: REM *LSINDT* (LESE DATEI EIN)-UNTERROUTINE
 1030 GOSUB 1600: REM *SETFLG* (SETZE FLAGS)-UNTERROUTINE
 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 STDFLD$(50)
 1140 DIM STAFLD$(50)
 1150 DIM TELFLD$(50)
 1160 DIM INDFLD$(50)
 1170 REM
 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
10000 REM *ADDVER*-UNTERROUTINE
10010 PRINT CHR$(12): REM LOESCHE BILDSCHIRM
10020 INPUT "NAMEN EINGEBEN"; NAMFLD$ (GROSS)
10030 INPUT "STRASSE EINGEBEN"; STRFLD$ (GROSS)
10040 INPUT "STADT EINGEBEN"; STDFLD$ (GROSS)
10050 INPUT "STAAT EINGEBEN"; STAFLD$ (GROSS)
10060 INPUT "TELEFONNUMMER EINGEBEN"; TELFLD$ (GROSS)
10070 LET RMOD = 1: REM "VERZEICHNIS MODIFIZIERT" FLAG SET
10080 LET INDFLD$ (GROSS) = STR$ (GROSS)
10090 LET GROSS = GROSS + 1
10100 LET TEST$ = " "
10110 REM *MODNAM*
10120 REM
10130 REM
10140 REM
10150 RETURN
```

```
4000 REM *EXECUT*-UNTERROUTINE
4010 REM
4019 IF WAHL = 6 THEN GOSUB 10000:
     REM SIEHE FUSSNOTE
4020 REM NORMALERWEISE SIEHE
     FUSSNOTE BEI "ON WAHL GOSUB etc."
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 *MODVER*
4090 REM 6 IST *ADDVER*
4100 REM 7 IST *MODVER*
4110 REM 8 IST *LOEVER*
4120 REM 9 IST *ENPROG*
4130 REM
4140 RETURN
```

Die *EXECUT*-Routine würde normalerweise nicht in Zeile 4019 stehen (daher die ungerade Zeilennummer), und Zeile 4020 würde statt dessen sein:

```
ON WAHL GOSUB Zahl, Zahl, Zahl etc.
```

oder eine Reihe von:

```
IF WAHL = 1 THEN GOSUB Zahl
IF WAHL = 2 THEN GOSUB Zahl etc.
```

Zeile 4019 wurde eingefügt, damit das Programm laufen kann, obwohl die anderen *EXECUT*-Unterroutinen bis jetzt noch nicht codiert wurden.

Das Büro im Computer

Da trotz sinkender Computerpreise die Programmentwicklung teuer blieb, entstand im kommerziellen Bereich ein Markt für Programme „von der Stange". Diese Systeme werden inzwischen von vielen kleinen und mittleren Betrieben eingesetzt.

Eine breite Palette von Anwendungen trägt die Bezeichnung „kommerzielles Programmpaket", wobei sich Einsatzbereiche und Kapazitäten sehr voneinander unterscheiden. Große Unterschiede bestehen dabei zunächst zwischen Cassettenprogrammen, die für Heimcomputer entwickelt werden, und Software, die für Diskettenlaufwerke ausgelegt ist. Obwohl eine Diskettenstation den Preis einer Anlage um 800 Mark bis 2000 Mark erhöhen kann, ist sie für jedes kommerzielle System sehr zu empfehlen, da sich gespeicherte Daten damit schneller abrufen lassen.

Der zweite große Unterschied besteht zwischen Einplatz- und Mehrplatzsystemen (Microcomputer, an die sich mehrere Terminals gleichzeitig anschließen lassen). In dieser Serie werden wir zunächst Einplatz-Systeme untersuchen, später aber auch auf Mehrplatzanwendungen eingehen.

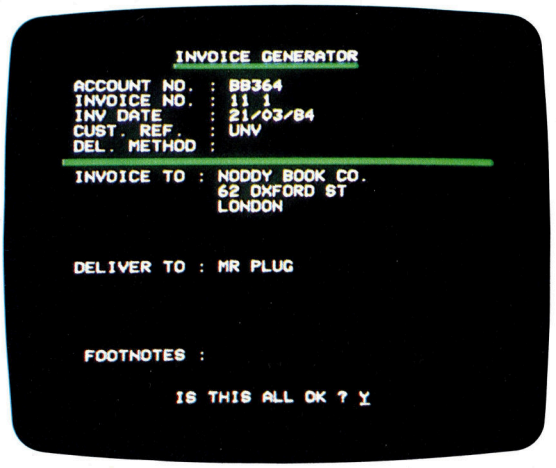
Kommerzielle Programme werden für den Computerlaien und nicht für EDV-Spezialisten geschrieben. Fast alle arbeiten – unabhängig von der Größe des Systems – nach der „interaktiven" Methode, bei der ein Anwender über den Bildschirm durch alle Funktionen des Systems geführt wird.

Ein typisches Programm blendet dabei „Menüs" auf, deren Funktionen sich über die Eingabe einer zugeordneten Ziffer aufrufen lassen. Bei der Anwahl einer Funktion erscheint oft ein zweites und auch drittes „Menü" mit weiteren Bearbeitungsmöglichkeiten. Die Dateneingabe ist zumeist dem manuellen Eintrag beispielsweise in ein Kassenbuch nachempfunden. Diese „Benutzerfreundlichkeit" begründete den Erfolg der kommerziellen Anwendungsprogramme.

Wir wollen die Funktionsweise dieses Programmtyps am Beispiel einer Fakturierung untersuchen. Eine Fakturierung enthält drei Hauptbereiche: Rechnungsschreibung, Bestellwesen und Gesamtüberblick (oder Bilanz). Mit der Rechnungsschreibung werden normalerweise alle ausgehenden Rechnungen erstellt, während die Warenbewegungen über das Bestellwesen geführt werden. Der Gesamtüberblick ermöglicht zu jeder Zeit die Analyse der finanziellen Situation.

Kommerzielle Programme arbeiten hauptsächlich mit Dateien, die sich in Datensätze und diese wiederum in einzelne Felder unterteilen lassen. Die Unterscheidung dieser drei Bestandteile ist einfach. Ein Rechnungsprogramm enthält eine „Hauptdatei", in der ein Datensatz jeweils die Daten eines Kunden speichert. Innerhalb des Datensatzes sind Informationen wie z. B. Name und Adresse in einzelnen Feldern untergebracht.

Das Programm muß die Daten weiterhin bearbeiten können. Ein Beispiel dafür sind Eingaberoutinen (zur Speicherung der Daten) und Berechnungsmodule, mit denen sich die Zahlenwerte einzelner Felder sowohl innerhalb



Die Kapazitätsgrenzen von Cassetten können mit Programmen erweitert werden, die sich bei Bedarf zusammen einsetzen lassen. Das nebenstehende Rechnungsprogramm kann manuell bedient werden, verarbeitet aber auch Daten, die von anderen darauf abgestimmten Programmen erstellt wurden. Diese Lösung bietet den Vorteil, daß sich ein Betrieb damit schrittweise auf Computer umstellen läßt.

Kommerzielle Cassettensoftware ist in der Vielfalt und Kapazität der einzelnen Programmteile eines Paketes stark begrenzt. Programme wie das nebenstehende beschränken sich daher nur auf die Ausführung einer bestimmten Aufgabe. Das Paket enthält zwar die Elemente eines Buchhaltungsprogramms, wie z. B. Debitoren und Kreditoren, Jahres- und Tagesbilanzen etc., ist aber nur zur Unterstützung des Bargeldverkehrs gedacht.

eines Datensatzes als auch in der gesamten Datei bearbeiten lassen.

Alle größeren kommerziellen Systeme arbeiten auf Disketten-Basis, um einen schnellen Zugriff auf die Rechnungs- und Bestelldateien zu gewährleisten. Dabei wird jede Bestellung und jeder Kauf in den Konten der einzelnen Kunden und Lieferanten so genau wie möglich aufgezeichnet.

Je mehr Daten eine Datei enthält, desto länger braucht ein Computer für die Sortierung. Es werden deshalb vielfach nur die ausstehenden Rechnungen und Bestellungen in allen Einzelheiten gespeichert („ausstehend" bedeutet, ein Vorgang ist noch nicht abgeschlossen bzw. eine Rechnung noch nicht bezahlt). Systeme dieser Art werden auch „Offene-Posten-Rechnung" genannt.

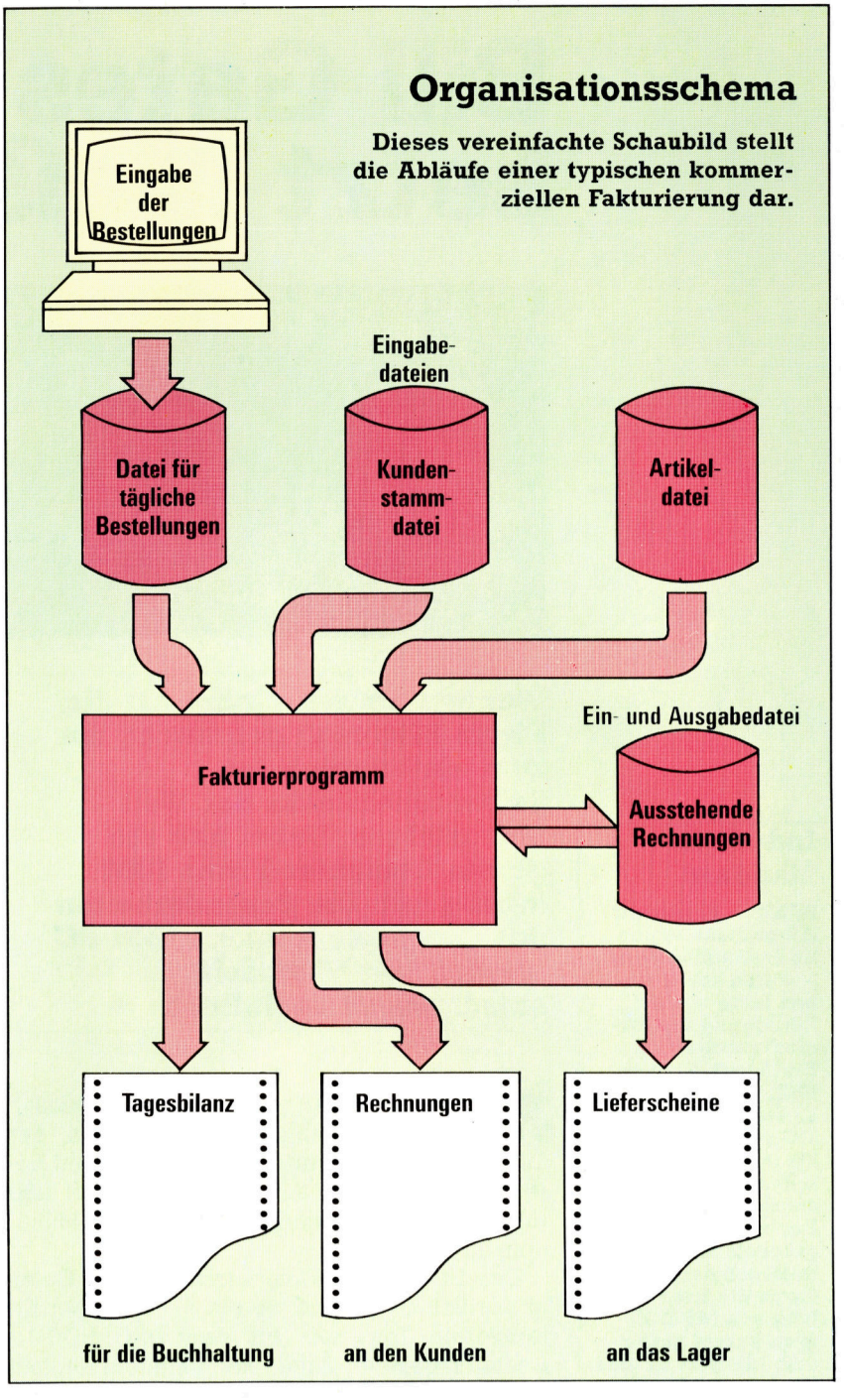
Bei einer anderen Lösung speichert der Computer nur die Gesamtsummen der Vorgänge. Programmpakete dieser Art werden als „Bilanzsysteme" bezeichnet. Sie verarbeiten weniger Informationen, haben dafür aber einen geringeren Speicherbedarf und sind einfacher zu bedienen. Jedes dieser Systeme ist ein Kompromiß zwischen der Menge der gespeicherten Einzelheiten und der Verarbeitungsgeschwindigkeit der Maschine.

Der Weg des Geldes

Kommerzielle Pakete enthalten oft eine ganze Reihe von Funktionen, die sich am leichtesten verstehen lassen, wenn man den „Weg des Geldes" verfolgt. So wird bei einem Rechnungs- bzw. Buchhaltungsprogramm im allgemeinen der Überblick über die Einnahmen und Ausgaben einer Firma als wichtigste Aufgabe angesehen. Die Führung eines Kassenbuches ist dafür die einfachste Lösung. Ein konventionelles (nicht über den Computer geführtes) Kassenbuch enthält auf jeder Seite eine Anzahl Spalten, in denen das eingenommene und ausgegebene Bargeld aufgezeichnet wird. Dabei wird bei jedem Betrag auch die Mehrwertsteuer verzeichnet, damit die entsprechenden Beträge vom Finanzamt abgefordert oder dorthin bezahlt werden können.

Ein Kassenbuch läßt sich täglich oder wöchentlich führen, wobei entweder jeder einzelne Vorgang eingetragen oder die Bareinnahme des gesamten Zeitraums als ein Eintrag verzeichnet wird. Der Unterschied zwischen einem Kassenbuch und einem vollwertigen Buchhaltungssystem mit Rechnungsschreibung, Bestellungen und Bilanz liegt hauptsächlich in der Menge der gespeicherten Daten.

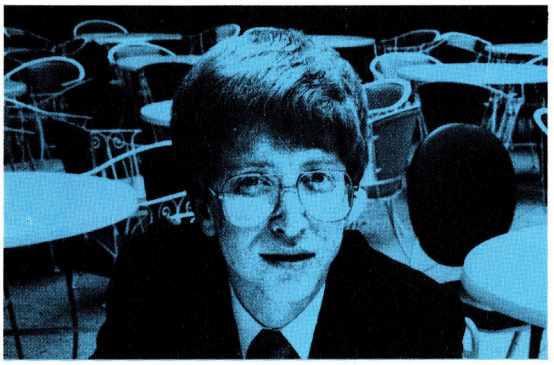
Fakturierungen und Buchhaltungsprogramme, die mit Cassetten arbeiten, haben nur wenig Arbeitsspeicher zur Verfügung, da die Daten nicht auf eine Diskette geschrieben werden können und der Arbeitsspeicher somit nicht für neue Daten freigegeben werden kann.



Statt drei verschiedene Programme für Rechnungen, Bestellungen und Bilanzen einsetzen zu müssen, werden diese drei Funktionen oft zu einem einzigen Programm zusammengefaßt. Ein typisches Beispiel dafür wäre ein System, bei dem ein Einzelhändler seine Einnahmen und Ausgaben auf einer wöchentlichen Basis eingeben kann und die Summierung der Daten die eigentliche Hauptaufgabe ist. Dabei wird nur das während der Woche eingenommene und ausgegebene Bargeld addiert, die Ergebnisse miteinander verglichen und der Gewinn oder Verlust angezeigt.

In unserem nächsten Artikel untersuchen wir den Aufbau dieser Art von Programmen.

Bill Gates setzt Maßstäbe



Weniger als zehn Jahre hat die Firma Microsoft gebraucht, um zum einflußreichsten Softwarehersteller der Welt aufzusteigen. Dabei half die größte Computer-Firma IBM kräftig mit: Die Besonderheiten des Computer-Renners IBM PC wurden von Microsoft entscheidend beeinflußt.

Microsofts Firmengeschichte ist ein klassisches Erfolgs-Märchen: Bill Gates, der 1972 noch als talentierter Amateur galt, saß bereits mit 28 Jahren auf dem Chefsessel und führte die Firma in wenigen Jahren zum Millionen-Umsatz.

Den Umgang mit Computern hatte Bill Gates in der Schule gelernt, wo ein dynamischer Elternverein Terminals für den DEC PDP-11-Rechner gestiftet hatte. Nach seinem Diplom an der Harvard-Universität eröffnete Bill zusammen mit seinem Schulfreund Paul Allen in der Heimatstadt Bellevue die Firma Traff-O-Data. Das Unternehmen befaßte sich in öffentlichem Auftrag mit der Überwachung des Verkehrsflusses in Seattle.

Micro-Durchbruch

Damals schaffte der Microprozessor gerade seinen Durchbruch, und Eingeweihte sahen schon die große Zukunft der Intel 4004- und 8008-Chips voraus. Bill Gates war zu dieser Zeit mit dem DEC PDP-11 gut vertraut und beschäftigte sich intensiv mit der Beseitigung systeminterner Mängel. Dabei kam er auf die Idee, das DEC-BASIC für den 8080-Prozessor umzuschreiben. Er besaß jedoch kein eigenes Entwicklungssystem und mußte mit seinen auf Band gespeicherten Daten zur Firma Altair nach Neu Mexiko fahren. Das Erstaunen war groß, als seine Entwicklung schon beim ersten Laden in den Rechner funktionierte: MBASIC war geboren, eine bis heute unübertroffene Computersprache.

Microsoft erwarb sich schnell einen Ruf durch die Ausrüstung von neuentwickelten Rechnern mit einem Betriebssystem. IBM wandte sich an Gates, als die Spezifikationen für einen „kleinen" Personal-Computer zur Diskussion standen. Gates schlug für diese Aufgabe Gary Kildall vor, der sich durch seine CP/M-Entwicklung einen Namen geschaffen hatte; IBM entschied sich jedoch für Microsoft. Also ging man in Gates' Firma daran, die Sprachen PASCAL, FORTRAN und MBASIC für 16-Bit-Rechner umzuändern. Außerdem wurde GW-BASIC entwickelt, das über erweiterte Musik- und Grafikfunktionen verfügt.

Weltweiter Standard

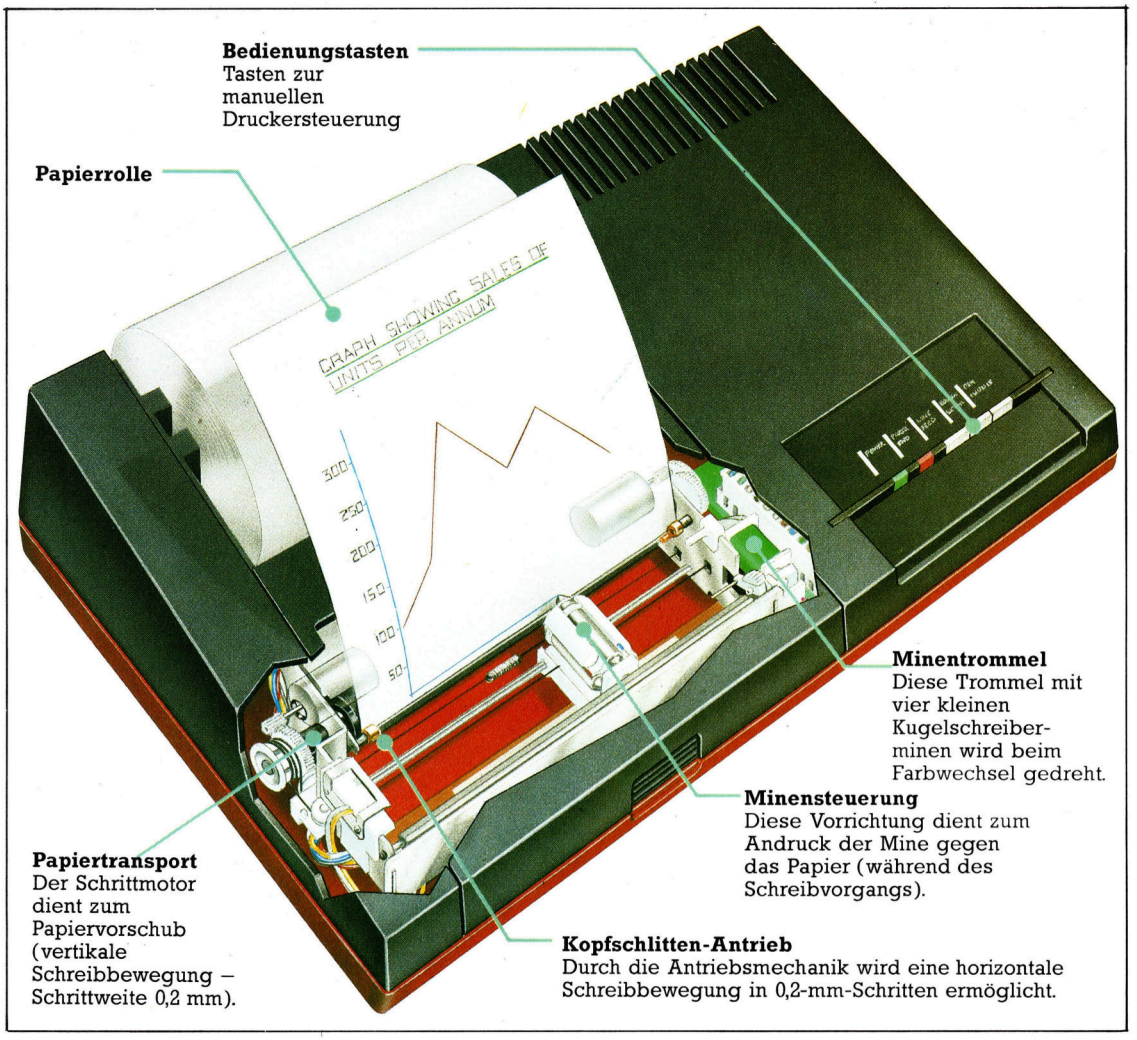
Zu dieser Zeit erkannte Gates, daß sich ein leistungsstarkes, mehrplatzfähiges Betriebssystem der Bell Laboratories für die neue Generation der 16/32 Bit-Prozessoren hervorragend eignete: Aus Unix wurde Xenix, das sowohl von Tandy als auch von Apple für ihre Rechnermodelle verwendet wurde – Microsoft hat damit auch dem Apple Macintosh seinen Stempel aufgedrückt.

Damit nicht genug: Die mit dem Japaner Kay Nishu gegründete Tochtergesellschaft ASCII-Microsoft verkauft im Fernen Osten Betriebssysteme für die neuen Handheld-Computer, etwa den NEC PC 8201 oder den Tandy 100. Die japanischen Hersteller wünschten sich einen weltweiten Standard – nicht nur bei den Programmiersprachen, sondern auch für die gesamte Rechner-Peripherie. Dieser Wunsch führte zum MSX-Standard. Es ist wahrscheinlich, daß Microsoft bald mit einem genormten Disketten-Format herauskommt, das die Datenübertragung zwischen den drei gebräuchlichsten Betriebssystemen – MSX, MSDOS und Xenix – möglich macht.

Die Innovateure von Microsoft denken aber noch weiter in die Zukunft: Derzeit arbeiten sie an Systemen, die noch einfacher zu bedienen sein sollen. Die bisherigen Erfolge lassen hoffnungsvolle Schlüsse zu – die „Maus" war demnach erst der Anfang.

Industrie-Standard

BASIC – Beginners' All-purpose Symbolic Instruction Code – wurde schon sieben Jahre vor der Einführung des Microprozessors am Dartmouth College, USA, entwickelt. Der BASIC-Dialekt MBASIC von Microsoft ist als internationaler Industrie-Standard anerkannt.
Der auf MBASIC begründete Ruf des Unternehmens ist durch die Entwicklung von MS-DOS noch besser geworden: MS-DOS ist eine Konkurrenz für Digital Researchs CP/M-System und läuft ebenfalls auf vielen Microcomputern.
In letzter Zeit hat Microsoft sein Angebot erweitert und bietet neben der reinen Software Peripherie für die komfortable Bedienung an: MS-WINDOWS mit der „Maus". Wie bei den Konkurrenten Apple und Xerox besteht auch die Microsoft-Maus aus einem Trackball mit Extratasten zur Cursor-Steuerung.



Das Gerät hat im Grafikmode eine Auflösung von etwa 500 x 2000 Bildpunkten und kann verschiedene Schriftgrößen darstellen. Die Druckfarbe wird durch Drehen der Schreibstifttrommel gewählt. Beim Zeichnen und Drucken wird das Papier auf- und abbewegt, während sich gleichzeitig der Schreibkopf hin- und herschiebt. Die Farbmine wird gegen das Papier gedrückt oder abgehoben, je nachdem ob gezeichnet oder nur neu positioniert werden soll. Die Zeichengenauigkeit an sich ist sehr hoch; erst häufiger Papiervor- oder -rückschub führt zu unpräzisen Darstellungen.

Doppelter Druck

Aus Japan kommen diese interessanten Geräte, die sowohl drucken als auch zeichnen können.

Komplizierte Plotter, mit denen sich detailreiche Grafik auch in Farbe zu Papier bringen läßt, sind für den Hausgebrauch meist zu teuer. Es gibt aber auch preiswerte Geräte, die sich ausgezeichnet für Heimcomputer eignen: Hinter Bezeichnungen wie Atari 1020, Commodore 1510 und Oric MCP-40 verbirgt sich bei näherem Hinsehen das gleiche Druckermodell. Es wird von einem japanischen Hersteller unter Berücksichtigung spezieller Firmenwünsche hinsichtlich Design und Typenbezeichnung gefertigt.

Dieser Plotter zeichnet nicht nur grafische Darstellungen, sondern kann auch Text „drukken". Daher wird er Printer/Plotter (Grafikdrucker) genannt. Der trommelförmige Schreibkopf mit vier Kugelschreiberpatronen sitzt auf einer Schlittenführung. Zum Zeichnen wird die gewünschte Patrone (Standardfarben sind Rot, Grün, Blau und Schwarz) in Druckposition gedreht und dann auf das Papier abgesenkt. Waagerechte Linien entstehen durch Bewegung des Schlittens allein, senkrechte durch Papiertransport bei ruhendem Kopf, diagonale durch Kombination von beidem. Textzeichen werden unter Verwendung der gleichen Methode produziert. Die entsprechenden Schriftbilder sind im Drucker abgespeichert.

Gute Qualität

Sobald der Computer einen Druckbefehl für ein Zeichen sendet, sucht sich der Drucker das zugehörige Symbol heraus. Das Ergebnis erscheint, im Vergleich zu den meisten preisgünstigen Matrixdruckern, in guter Qualität.

Beim Textausdruck arbeitet der Printer/Plotter ähnlich wie andere Drucker. Er erzeugt auf

dem 115 mm breiten Papier nach Wahl 40 oder 80 Zeichen je Zeile. Wegen der geringen Papierbreite wirkt das 80-Zeichen-Format relativ eng, aber die Zeichen sind sehr gut zu erkennen. Die Textausgabe ist in jeder der vier Farben möglich. Die Druckgeschwindigkeit von zehn Zeichen pro Sekunde ist jedoch, verglichen mit den meisten Matrixdruckern, ziemlich gering.

Nach Umschaltung auf den Grafikmode erzeugt das Gerät gerade Linien, Kurven sowie vergrößerte Schriftsymbole.

Für das Zeichne von Linien wird zum Beispiel folgende BASIC-Anweisung eingegeben:

```
LPRINT "D 0,100,100,100,100,0,0,0"
```

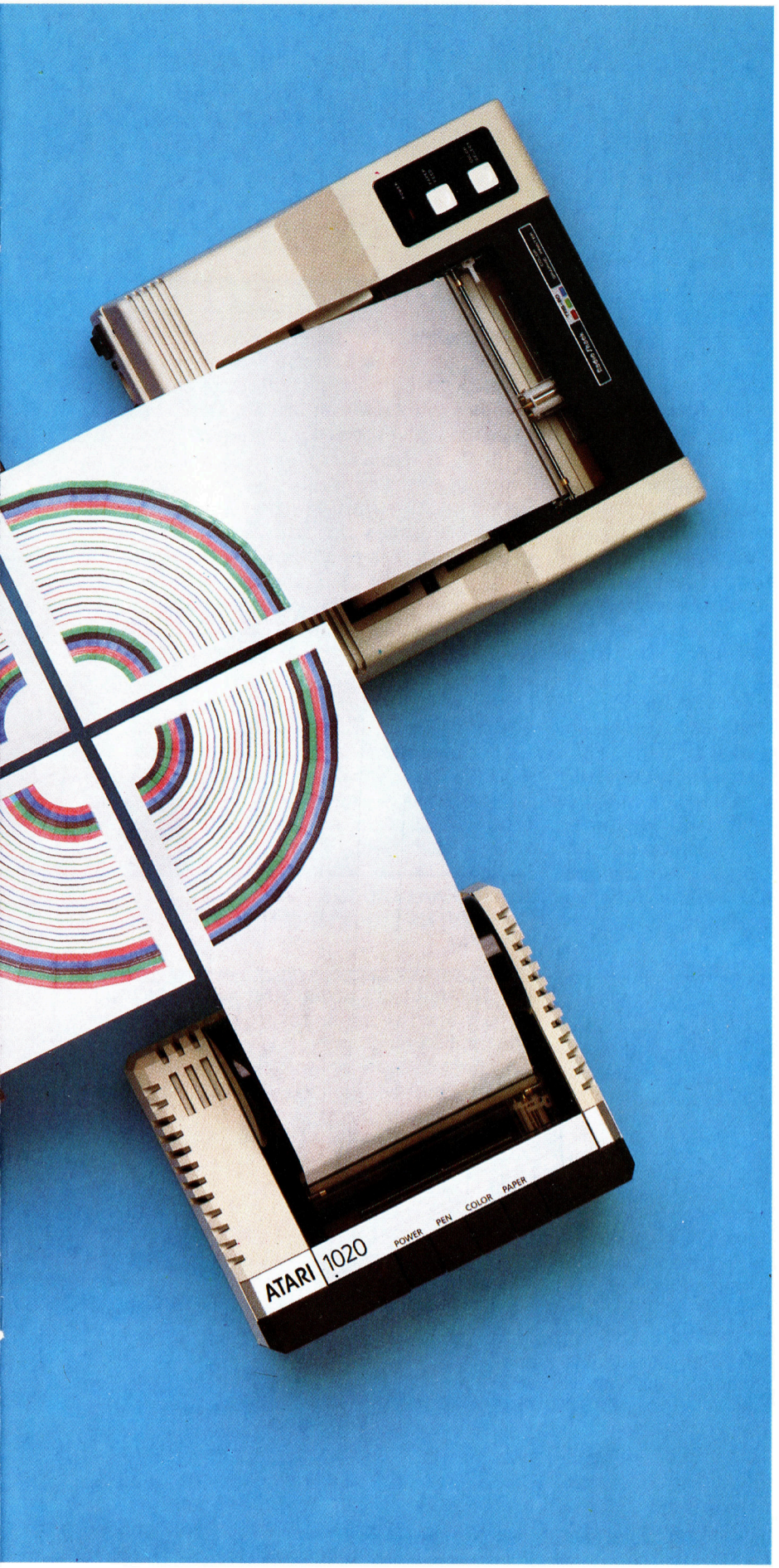
Dabei steht das „D" (Draw) für „Linienziehen", und darauf folgen die Koordinaten der Punkte, die verbunden werden sollen. Im obigen Beispiel würde ein Quadrat gezeichnet (vorausgesetzt, daß sich der Schreibkopf in der Ausgangsposition 0,0 befindet). Andere Zeichenbefehle haben ein ähnliches Format.

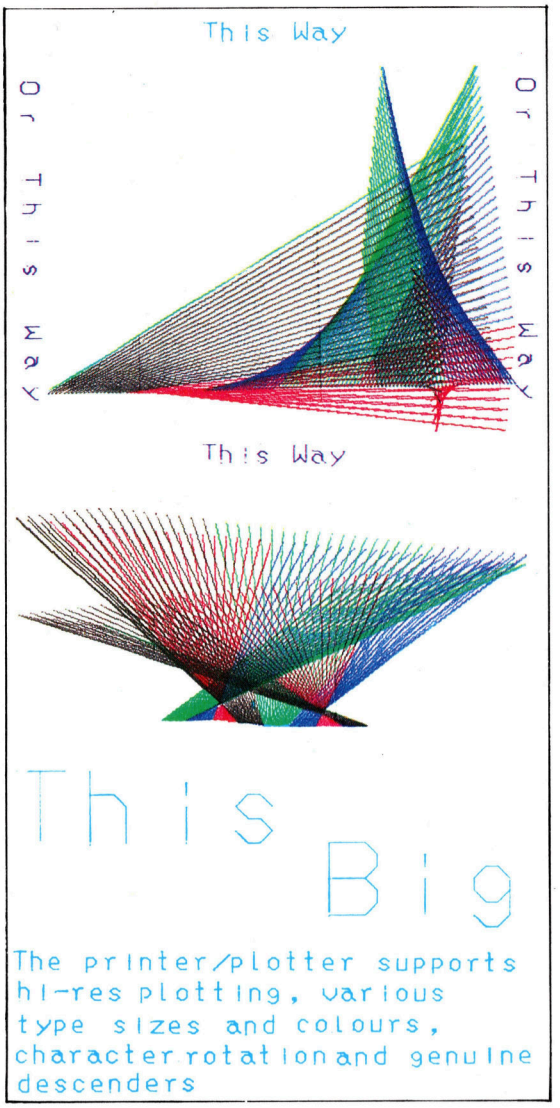
Aufwendige Grafiken

Die Papierbreite wird rechnerintern in 480 Positionen unterteilt, die der Schreibkopf einzeln ansteuern kann. Je nach Umfang der verwendeten Papierrolle lassen sich fast endlose Zeichnungen fabrizieren. Meterlange präzise Grafiken können auf diesem Drucker jedoch nicht erstellt werden, da sich das Papier durch die zahlreichen Papiervorschübe verschiebt und somit die Überschneidungspunkte der Linien nicht mehr genau aufeinandertreffen.

Sie können mit dem Printer/Plotter auch sehr aufwendige Grafiken anfertigen, aber dazu gehört dann eine langwierige Programmierung, und es gibt keine Möglichkeit, die Bildschirmdarstellung direkt auf das Papier zu übertragen. Strichzeichnungen sind zwar einfach anzufertigen, aber flächige Darstellungen müssen wegen des Fehlens von Anweisungen wie PAINT oder FILL durch eine Folge zahlloser feiner Linien erzeugt werden. Dieses Verfahren nimmt viel Zeit und Farbe in Anspruch. Kurvenverläufe lassen sich dagegen sehr elegant wiedergeben, z. B. gibt es eine spezielle Anweisung zum Zeichnen des Achsenkreuzes einschließlich Skalierung.

Im Textmode sind zwei Typengrößen wählbar, beim Plotten auch noch größere Schriftformate. Außerdem können Sie das Papier längs beschreiben, wenn Sie umfangreiche Texte unterbringen wollen. Die Farbpatronen trocknen übrigens schnell ein, wenn sie nicht aus dem Gerät entfernt und mit Kappen versehen werden. Beim Einschalten des Gerätes wird ein Systemtest aufgerufen, bei dem automatisch vier Kästchen auf das Papier gebracht werden, anhand derer Sie die Schreibqualität der Minen überprüfen können.





Die Tabelle zeigt, welches Plottermodell an den jeweiligen Computer angeschlossen werden kann. Bei Atari, Commodore, Oric und Sharp werden passende Kabel für die Rechner mitgeliefert. Der „Sinclair Spectrum" fehlt in der Aufstellung, da er über keine geeignete Schnittstelle verfügt.

	Atari 1020 400 Mark	Commodore 1520 550 Mark	MCP-40 500 Mark	Oric MCP-40 500 Mark	Sharp MZ-1P01 610 Mark	Tandy CGP-115 615 Mark
Atari 400/600XL/800+XL	●					
Acorn B			●	●		●
CGL/Sord M5			●	●		●
Colour Genie						●
Commodore VC 20		●				
Commodore 64		●				
Dragon 32/64			●	●		●
Memotech MTX			●	●		●
Oric-1/Atmos			●	●		●
Sharp MZ-700					●	
Tandy TRS-80 Colour						●

Mehr-stimmig

Die Soundbefehle der Heimcomputer von Atari können vier Stimmen unabhängig voneinander steuern.

Anhand zahlreicher Computerspiele läßt sich aufzeigen, daß die Atari-Computer über ausgezeichnete Soundmöglichkeiten verfügen. Die Steuerung der entsprechenden Funktionen ist jedoch teilweise etwas eigenwillig. Es gibt vier unabhängige Oszillatoren für Rechteckschwingungen mit einem Tonumfang von je drei Oktaven. Um dem Klang „Farbe" zu geben, kann das Signal nach der Erzeugung auf sieben verschiedene Arten verzerrt werden. Diese Möglichkeiten lassen sich zwar leicht über den SOUND-Befehl des Atari-BASIC steuern, doch sind damit die faszinierenden Fähigkeiten des Soundchips POKEY mit seinen Spezialfiltern und besonderen Ausführungsarten bei weitem noch nicht erschöpft. Der gesamte Bereich der Klangsteuerung kann über POKE-Befehle und durch Verwendung der Maschinensprache voll eingesetzt werden. Leider würde eine ausführliche Beschreibung dieser Möglichkeiten den Rahmen unseres Kurses sprengen. Die Tonausgabe ist nur über den Lautsprecher des Fernsehers möglich.

Dieser Befehl ist einfach aufgebaut und hat folgendes Format:

```
SOUND O, P, D, V

O = Oszillator (0—3)
P = Tonhöhe (0—255)
D = Verzerrung (gerade Zahlen zwischen
    0—14)
V = Lautstärke (1—15)
```

Der SOUND-Befehl kann jeweils nur einen Oszillator ansprechen. Bei vierstimmigen Harmonien, die alle Oszillatoren einsetzen, sind dadurch leichte Tonverzögerungen hörbar.

Die Tonhöhe wird etwas seltsam berechnet, wodurch einige Frequenzen ungenau sind. Die Frequenz nimmt bei steigenden Tonhöhenzahlen ab und bietet einen Tonbereich von „C" bei 29 (1046,5 Hz) bis „C" bei 243 (130,81 Hz).

Die folgende Tabelle enthält einige Tonhöhenzahlen und die entsprechenden Notensymbole. Das Atari-Handbuch beinhaltet die vollständige Liste aller drei Oktaven mit sämtlichen Halbtönen.

	Oktave – 1	Oktave – 3
(Mittleres)	C – 121	C – 29
	B – 128	B – 31
	A – 144	A – 35
	G – 162	G – 40
	F – 182	F – 45
	E – 193	E – 47
	D – 217	D – 53
	C – 243	C – 60

Die Variable P für die Verzerrung entspricht dem Rauschkanal anderer Computermodelle, ist jedoch weitaus vielseitiger. Jede gerade Zahl mischt eine unterschiedliche Zusammenstellung von Zufallsimpulsen in das Signal der Oszillatoren. Eigenartigerweise erzeugt die Zahl 10 ein verzerrungsfreies Signal und nicht, wie man erwarten würde, die Null. Durch Probieren und den „gefühlvollen" Einsatz der Verzerrungen lassen sich interessante Klänge erzeugen, die sich gut für Spezialeffekte eignen.

Die Lautstärke „V" kann auf eine Zahl zwischen 1 und 15 gesetzt werden, wobei der mittlere Bereich bei 7 oder 8 liegt. Es gibt keine einfache Methode, die Länge der Töne oder der dazwischenliegenden Pausen festzulegen. Normalerweise werden dafür sorgfältig abgestimmte FOR . . .NEXT-Schleifen verwandt.

Das folgende Beispiel zeigt den Einsatz von SOUND. Dabei wird ein unverzerrtes „G" in der Oktave 3 auf dem Oszillator 1 mit einer Lautstärke von 8 während der Ausführzeit von 50 FOR . . .NEXT-Schleifen gespielt:

```
10 SOUND 1,40,10,8
20 FOR N=1TO50:NEXT N
30 END
```

Der Befehl END in Zeile 30 schaltet alle Oszillatoren ab. Ein neuer SOUND-Befehl für den gleichen Oszillator würde den alten Ton ebenfalls beenden und sofort den neuen erklingen lassen. Ein Programm für eine einfache Melodie sieht folgendermaßen aus:

```
10 REM*DIXIE*
20 FOR I=1TO7
30 READ N:REM*NOTE*
40 SOUND 3,N,10,7:REM*SPIEL DIE NOTE*
50 FOR P=1TO400:NEXT P:REM*PAUSE*
60 NEXT I
70 DATA 219, 162, 128, 144: REM*D G B A*
80 DATA 162, 193, 162:REM*G E G*
90 END
```

Die Soundmöglichkeiten des Atari sind auch über den Einsatz des POKE-Befehls an den Speicherstellen 53760 bis 53763 zugänglich. Bei dieser Methode laufen die Tonroutinen schneller ab, und alle Oszillatoren lassen sich gleichzeitig betreiben. Weitere Informationen finden Sie in speziellen Büchern.

Tanzende Lichter

Das Grafiksystem des Oric läßt viele Ähnlichkeiten mit dem des Spectrum erkennen.

Der Oric-Heimcomputer kam 1983 als Alternative zu dem ZX Spectrum auf den Markt. Er bietet vier Darstellungsmodi, von denen sich allerdings nur einer für hochauflösende Grafik verwenden läßt. Der Anwender hat die Möglichkeit, zwischen acht Farben zu wählen. Die Farbgebung des Vorder- und Hintergrundes wird mit den Befehlen INK und PAPER festgelegt. Das BASIC des Oric enthält einige Spezialbefehle als Programmierhilfe für hochauflösende Grafik.

Drei Grafikarten

Auf dem Bildschirm werden 28 Zeilen mit je 40 Zeichen dargestellt. Der Standard-ASCII-Characterset läßt sich nach Laden in den RAM-Speicher verändern. Zusätzlich kann der Anwender 80 Zeichen selbst definieren. Die einzelnen Zeichen sind jedoch nicht aus der üblichen acht-mal-acht-Pixelmatrix aufgebaut, sondern aus acht mal sechs Bildpunkten. In der hohen Auflösung ist der Bildschirm in 240×200 Pixel aufgeteilt, wobei die unteren drei Bildschirmzeilen für Informationen und Fehlermeldungen reserviert sind. Ein PAINT-Befehl ist beim Oric nicht vorgesehen, aber mit ein wenig Geschick läßt sich mit dem Befehl FILL der gleiche Effekt erzielen. Wie bei dem Spectrum lassen sich hochauflösende Grafik und Text im gleichen Bildschirmbereich miteinander mischen, bei dem Oric kann jedoch jede Linie innerhalb eines Zeichenfeldes eine eigene Farbe haben, während auf dem Spectrum für jedes Zeichenfeld nur eine Farbe möglich ist.

Der Oric bietet drei Grafikarten mit niedriger Auflösung: TEXT, LORES0 und LORES1. LORES0 und LORES1 arbeiten auf die gleiche Weise, bringen aber unterschiedliche Zeichensätze auf den Schirm. Im TEXT-Mode lassen sich die Zeichen mit dem TAB-Befehl nur horizontal positionieren. In den beiden LORES-Grafikarten ist aber über PLOTx,y,A$ auch eine vertikale Positionierung möglich. Dabei geben x und y die Koordinaten einer bestimmten Position an, während A$ das enthält, was auf dem Bildschirm dargestellt werden soll. Das folgende Programm zeigt, wie ein Name vertikal geschrieben werden kann:

```
10 REM VERTIKALE SCHRIFT
20 CLS
30 LORES0
40 A$="STEVE"
50 FOR X=1TO5
60 B$=MID$(A$, X, 1)
70 PLOT 16, 11 + X, B$
80 NEXT X
90 END
```

Der Befehl HIRES aktiviert die hochauflösende Grafik. Der Nullpunkt der Bildschirmkoordinaten liegt dabei in der oberen linken Ecke des Schirms.

Das BASIC des Oric enthält einige Befehle, die grafische Darstellungen vereinfachen. CURSET x,y,k setzt den Cursor auf Koordinatenposition x, y. Über die Variable „k" können dabei folgende Funktionen (unten) aufgerufen werden:

k	Funktion
0	stellt das Pixel in der Hintergrundfarbe dar
1	stellt das Pixel in der Vordergrundfarbe dar
2	kehrt die Farben um (Invertierung)
3	hat keine Funktion

CURMOVx,y,k funktioniert ähnlich wie CURSET, bezieht die Cursorbewegung jedoch auf die vorige Position des Cursors. DRAWx,y,k zeichnet eine gerade Linie von der augenblicklichen Cursorposition zu dem mit x,y festgelegten Bildpunkt, während CIRCLEr,k einen Kreis mit dem Radius r auf den Bildschirm bringt. Interessant ist der Befehl PATTERNn, der Linien oder Kreise in eine Folge von Punkten oder Strichen aufteilt. Das Muster wird dabei von der Variable „n" bestimmt, die Werte zwischen 0 und 255 annehmen kann. Der Oric setzt diese Zahl in ein Bitmuster um und produziert ein Punkt/Strichmuster, das dem binären Zahlenwert entspricht. Hier zwei Beispiele:

Zahl n	Binärzahl	Punkt/Strichmuster
170	10101010	- - - - - - - - - -
15	00001111	— —

Schließlich gibt es noch den Befehl FILLa,b,n. Allen Pixelzeilen der Zeichenfelder ist eine Zahl zugeordnet, über die die Vorder- und Hintergrundfarben, das Zeichen und die Darstellungsart (blinkend oder nicht blinkend) festgelegt sind.

Diese Zahl wird als die „Eigenschaft" (das Attribut) der Zeile bezeichnet. FILLa,b,n füllt „b" Zeichenfelder mit „a" Zeilen und den Eigenschaften, die die Zahl „n" festlegt.

Dieses Programm zeigt einige Fähigkeiten der hochauflösenden Grafik des Oric. Mit einer Reihe von Kreisen mit wachsendem Radius wird ein Tannenzapfen gezeichnet. Der Befehl PATTERN versieht die Kreislinien mit einem Muster.

```
10 REM TANNENZAPFEN
20 HIRES
30 CURSET 120,0,3
40 PAPER3:INK4
50 FOR R=1TO65
60 PATTERN 200-R
70 CURMOV0,2,3
80 CIRCLE R,1
90 NEXT R
100 END
```

Darf man Programme kopieren und verkaufen?

Die Programmiersprache LOGO wird auf Datenträgern angeboten.

? Wie kann mein Computer mit LOGO arbeiten?

Wie bereits im ersten Teil des LOGO-Kurses erwähnt wurde, ist LOGO eine Programmiersprache, die wie andere Sprachen, die nicht im Computer integriert sind, etwa Pascal oder Cobol, zusätzlich erworben werden muß. LOGO wird auf Datenträgern in Form von Cassette, Diskette oder Cartridge angeboten. Für den C 64 zum Beispiel ist LOGO auf Diskette und für den Atari auf Cartridge erhältlich. Nachfolgend finden Sie eine Aufstellung der Computer, für die LOGO derzeitig angeboten wird: Apple, Atari, Commodore, DEC, Epson, IBM, NDR-Computer, Philips, Schneider, Siemens, Sinclair, Sony und TI.

Für diejenigen, die mit einem deutschen Befehlssatz arbeiten möchten, wird vom IWT-Verlag eine deutsche LOGO-Version speziell für den Apple angeboten.

In den letzten Monaten erschienen zahlreiche LOGO-Lehrbücher, die den Umgang mit dieser Sprache erleichtern. Wer sich noch intensiver mit LOGO beschäftigen möchte, findet außerdem Literatur über die Entstehung dieser Programmiersprache, auch in deutscher Übersetzung.

? Welchen Computer würden Sie mir empfehlen?

Diese Frage hören wir nur allzuoft. Eine allgemein gültige Antwort darauf gibt es nicht. Sie sollten sich vor dem Kauf eines Computers genau überlegen, wie und wo Sie das Gerät einsetzen und wieviel Geld Sie investieren wollen. Generell kann man sagen, daß für fast jeden Anwendungsbereich und jeden Geldbeutel ein entsprechender Computer angeboten wird.

Möchten Sie den Computer hauptsächlich zu Hause benutzen und sich zuerst langsam an diese Materie „herantasten", sollten Sie sich für ein Gerät entscheiden, für das eine umfangreiche Software- und Literaturbibliothek angeboten wird. Ferner sollte für den Computer Ihrer Wahl eine große Palette von Peripheriegeräten zur Verfügung stehen. Es nützt Ihnen nichts, wenn Sie ein preisgünstiges Gerät erwerben, für das weder Drucker noch Diskettenstation vorhanden sind, auch wenn Sie diese zum gegenwärtigen Zeitpunkt noch nicht brauchen. Je länger Sie mit dem Computer arbeiten, desto größeren Wert werden Sie auf ein ausbaubares System legen.

Informieren Sie sich anhand von Fachzeitschriften, oder lassen Sie sich durch Bekannte, die bereits einen Computer besitzen, über Vor- und Nachteile des Systems beraten. In den meisten der sogenannten Computerabteilungen in Kaufhäusern oder Elektronikläden ist der Computer an sich und im Besonderen noch weitestgehend ein Buch mit sieben Siegeln.

? Darf man Programme kopieren und weiterverkaufen?

Die Antwort ist ein schlichtes NEIN! Das Kopieren und Weitergeben von Software ist Diebstahl geistiger Eigenleistung und wird, genau wie das Duplizieren von Literatur, Filmen oder Tonaufnahmen, strafrechtlich verfolgt.

Wenn Sie bereits Erfahrung mit dem Programmieren gesammelt haben, können Sie sicherlich nachvollziehen, welcher Arbeitsaufwand mit der Planung, Programmierung und Prüfung eines Programms verbunden ist. Wären Sie begeistert, wenn Sie Ihr mühsam erstelltes Programm unter den Kleinanzeigen in Zeitschriften entdeckten?

Kommerziell angebotene Software wird deshalb mit speziellen Schutzroutinen versehen, die das Auflisten und Kopieren verhindern sollen. Zum gegenwärtigen Zeitpunkt gibt es jedoch den optimalen Kopierschutz noch nicht. Sogenannte „Knacker", nicht zu verwechseln mit den „Hakkern", die ihre Hauptaufgabe darin sehen, sich der Datenbanken „zu bedienen", benutzen ihre Kenntnisse, um diesen Schutz zu knacken, die Programme zu kopieren und dann auf dem grauen Markt anzubieten. Das geschieht in den meisten Fällen per Kleinanzeige, wo die „Frisch geknackte Supersoftware" preisgünstig offeriert wird. Das Nachsehen haben dabei nicht nur die legalen Softwareanbieter und Programmierer, sondern in zahlreichen Fällen auch die Käufer der Raubkopien, da diese Programme oftmals nicht lauffähig sind.

Programme dürfen nicht einfach kopiert werden – Copyright beachten!

Zufallszahlen

In diesem Teil des LOGO-Kurses werden Sie die Möglichkeiten kennenlernen, die LOGO für die Bearbeitung von numerischen Werten bietet.

Fast alle LOGO-Versionen arbeiten mit ganzzahligen (Integer-) sowie mit Dezimal-Werten in Verbindung mit den Operatoren +, –, * und /, die wie üblich zwischen die Werte gesetzt werden und mathematische Operationen definieren (etwa 3 + 4). Einige LOGO-Versionen beinhalten Befehle, durch die der Operator überflüssig wird, wie zum Beispiel bei SUM 3 4, wodurch die beiden Werte automatisch addiert werden.

Dazu ein Beispiel mit den Prozeduren SUM und PRODUCT:

```
TO SUM :A :B
  OUTPUT :A + :B
END

TO PRODUCT :A :B
  OUTPUT :A * :B
END
```

Die Reihenfolge, in der die Operationen ausgeführt werden, ergibt sich aus den üblichen mathematischen Grundregeln. Zuerst sind die in Klammern stehenden Berechnungen, dann Multiplikation und Division und zuletzt Addition und Subtraktion zu bearbeiten.

```
PRINT (3 + 4) * 5
PRINT 3 + 4 * 5
```

Und nun in der anderen Schreibweise:

```
PRINT PRODUCT 5 SUM 3 4
PRINT SUM 3 PRODUCT 4 5
```

Wie Sie sehen, sind bei diesen Befehlen die Operatoren überflüssig, da bereits PRODUCT bzw. SUM bestimmen, wie die Zahlen zu berechnen sind.

Um eine Division anzuzeigen, wird das / eingegeben. Zusätzlich gibt es in LOGO die Anweisungen QUOTIENT und REMAINDER:

```
QUOTIENT 47 5 ist 9
REMAINDER 47 5 ist 2
```

Die einfachste Methode, um eine Zahl mit der Basis 10 in eine Binärzahl umzuwandeln, ist die, sie durch zwei zu teilen, bis das Ergebnis Null lautet. Die Restwerte (Remainder) dieser Divisionen (rückwärts gelesen) ergeben dann den binären Wert, wie die Umwandlung der Zahl 12 zeigt:

```
12/2 = 6; Restwert = 0
 6/2 = 3; Restwert = 0
 3/2 = 1; Restwert = 1
 1/2 = 0; Restwert = 1
```

In diesem Beispiel ergibt der binäre Code der Zahl 12: 1100.

Mit Hilfe von QUOTIENT und REMAINDER läßt sich diese Berechnungs-Technik auch in LOGO praktizieren. Mit der folgenden Prozedur werden die Ergebnisse der Divisionen rückwärts ausgegeben:

```
TO BIN :X
  IF :X = 0 THEN STOP
  BIN QUOTIENT :X 2
  PRINT1 REMAINDER :X 2
END
```

Für die Ausgabe von ganzzahligen Werten stehen in LOGO zwei Anweisungen zur Verfügung: INTEGER und ROUND. INTEGER zeigt den ganzzahligen Teil eines Wertes an und ignoriert alle Stellen hinter dem Dezimalpunkt. ROUND dagegen rundet den Wert zur nächstliegenden ganzen Zahl auf bzw. ab.

Das folgende Programm berechnet den Zinseszins einer Einlage auf der Basis des angegebenen Zinssatzes. INTEGER gibt dabei den Betrag in Mark aus, während ROUND die Pfennige auf- oder abrundet.

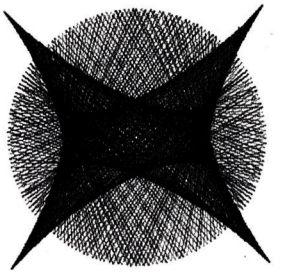
```
TO ZINSESZ :KAPITAL :ZINSEN :JAHRE
  IF :JAHRE = 0 THEN VERDIENST.PRINT
    :KAPITAL STOP
  ZINSESZ :KAPITAL * (1 + :ZINSEN / 100)
    :ZINSEN :JAHRE — 1
END

TO VERDIENST.PRINT :GELD
  MAKE "MARK INTEGER :GELD
  MAKE "PFENNIGE ROUND (:GELD —
    :MARK) * 100
  (PRINT :MARK "MARK :PFENNIGE
    "PFENNIGE)
END
```

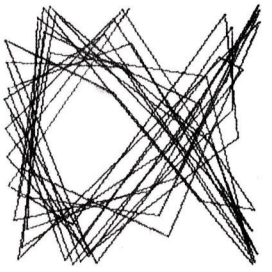
In vorangegangenen Programmbeispielen wurden bereits die Operatoren =, ‹ und › für logische Vergleiche eingesetzt. Zusätzlich sind in einigen LOGO-Versionen ALLOF, ANYOF und NOT enthalten, die sich mit den anderen Operatoren kombinieren lassen. ALLOF gibt TRUE aus, wenn beide Eingaben „wahr" sind, ANYOF dagegen, wenn eine der Eingaben „wahr" ist. NOT errechnet TRUE, wenn sämtliche Eingaben „falsch" sind.

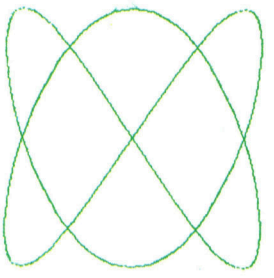
```
IF ANYOF :X > 0 :X = 0 THEN PRINT
  "POSITIV
IF NOT :X < 0 THEN PRINT "POSITIV
IF ALLOF :X > 0 :X < 100 THEN PRINT
  [ZWISCHEN 0 UND 100]
```

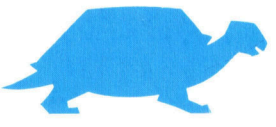
Die Operation NUMBER? fragt ab, ob es sich bei der Eingabe um einen numerischen Wert handelt, und gibt das Ergebnis ebenfalls in TRUE oder FALSE aus. In der nächsten Prozedur wird die Abfrage PRIME? benutzt, die

Grafische Figuren









LOGO-Dialekte

Das Atari-LOGO enthält die Anweisungen SUM und PRODUCT; das Spectrum-LOGO beinhaltet zusätzlich den DIV-Befehl. Bei der Apple-Version läßt sich QUOTIENT einsetzen.
Bei einigen LOGO-Versionen muß
INTEGER durch INT,
NUMBER? durch NUMBERP,
PRINT1 durch TYPE,
SETXY durch SETPOS und
DRAW durch CS ersetzt werden.
Ferner variieren die Eingabeformen bei IF z. B.: IF :X = 0 THEN "ZERO

Übungen

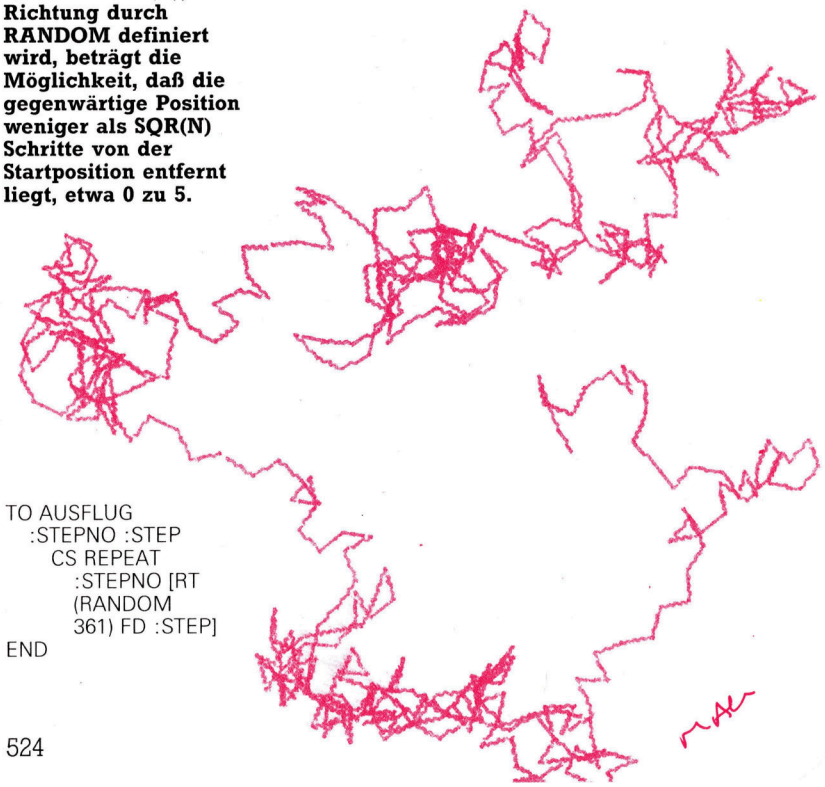
1. Schreiben Sie eine Prozedur, die die Potenz der Zahl n berechnet, etwa POWER 4 2 = 16.
2. Entwickeln Sie ein Programm, das Dezimalzahlen in hexadezimale Werte umwandelt.
3. Schreiben Sie eine Prozedur, die überprüft, ob die eingegebene Zahl gerade oder ungerade ist, und als Ergebnis TRUE oder FALSE ausgibt.
4. Ermitteln Sie mit Hilfe der Monte-Carlo-Methode den Bereich unter der Kurve $y=x^2$ zwischen x=0 und x=10.

prüft, ob die Eingabe eine Primzahl darstellt oder nicht. Zuerst überprüft das Programm jedoch, ob der eingegebene Wert numerisch und größer als zwei ist. PRIME.TEST berechnet anschließend, ob zwischen der Wurzel der Eingabezahl und 2 ein ganzzahliger Wert liegt, der sich dividieren läßt, ohne einen Restwert zu hinterlassen.

```
TO PRIME? :NO
  IF NOT NUMBER? :NO THEN PRINT
    [NOT A NUMBER DUMMY] STOP
  IF :NO < 2 THEN OUTPUT "FALSE
  OUTPUT PRIME.TEST :NO INTEGER
    SQRT :NO
END

TO PRIME.TEST :NO :FACT
  IF :FACT = 1 THEN OUTPUT "TRUE
  IF (REMAINDER :NO :FACT ) = 0 THEN
  OUTPUT "FALSE
  OUTPUT PRIME.TEST :NO :FACT — 1
END
```

Turtle-Ausflug

In dieser Prozedur unternimmt die Turtle einen längeren Ausflug. Nach einer bestimmten Schrittanzahl (N), deren Richtung durch RANDOM definiert wird, beträgt die Möglichkeit, daß die gegenwärtige Position weniger als SQR(N) Schritte von der Startposition entfernt liegt, etwa 0 zu 5.



```
TO AUSFLUG
  :STEPNO :STEP
    CS REPEAT
      :STEPNO [RT
      (RANDOM
      361) FD :STEP]
END
```

Die Anweisung RANDOM n erzeugt einen ganzzahligen Zufallswert zwischen 0 und n-1. Die folgende Prozedur steuert die Turtle über den Bildschirm, wobei sie nach jedem Schritt eine Drehung um einen Zufallswinkel ausführt. Mit der Variablen A wird der maximale Drehungswinkel bestimmt.

```
TO DREHEN :A
  FORWARD 1
  RIGHT (— :A / 2 + RANDOM :A)
  DREHEN :A
END
```

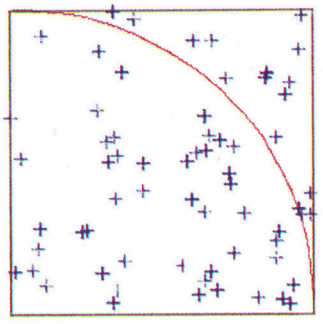
Die sogenannte „Monte-Carlo-Methode" stellt eine Technik zur Lösung von mathematischen Aufgaben in Verbindung mit Zufallszahlen dar.

Die folgende Berechnung soll unter Verwendung dieser Methode den Näherungswert von Pi ermitteln. Die Abbildung zeigt einen Viertelkreis in einem 100 x 100 Einheiten umfassenden Quadrat. Dieser Bereich ergibt sich aus folgender Rechnung: (1/4) x Pi x 100 x 100. Das Verhältnis des Kreisausschnittes zum Quadrat berechnet sich aus Pi/4. Jetzt werden 1000 Punkte wahllos (per Zufallsgenerator) auf diesem Quadrat verteilt. Mit Hilfe der Variablen IN soll ermittelt werden, wieviele Punkte sich anschließend in diesem Kreisausschnitt befinden. Der Wert aus IN/1000 müßte ein annähernd Ergebnis wie Pi/4 enthalten.

```
TO MC
  DRAW
  PU
  MAKE "IN 0
  MC1 1000 100 100
  (PRINT [PI BETRAEGT] 0.004 * ( :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
```

Die Prozedur MC legt die Bedingungen fest, ruft MC1 auf und stellt die Ergebnisse dar. MC1 ruft RANDOM.POINT zur Positionierung der Turtle auf und aktualisiert den Wert von IN.

Monte-Carlo-Methode



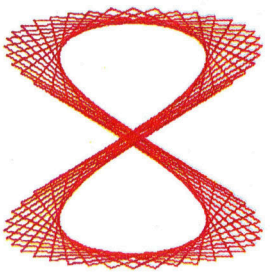
RANDOM.POINT setzt die Turtle auf eine Zufallsposition im Quadrat, und INSIDE? prüft, ob sie sich innerhalb des Abschnittes befindet. Die nächsten Prozeduren ermitteln die x-Koordinaten der Punkte mit der Sinus- und die y-Koordinaten durch die Kosinus-Funktion.

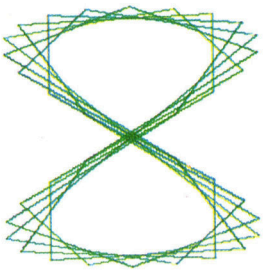
```
TO LJ :COEFF1 :COEFF2 :STEP
  DRAW PU HT
  POS :COEFF1 :COEFF2 0 PD
  LJ1 :COEFF1 :COEFF2 0 :STEP
END
```

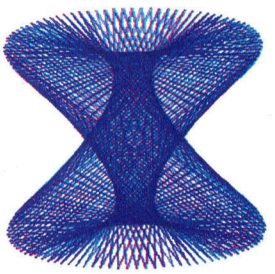
```
TO RANDOM.POINT :XNO :YNO
  SETXY RANDOM :XNO RANDOM :YNO
END

TO INSIDE?
  IF ( XCOR * XCOR + YCOR * YCOR )
     < 10000 THEN OUTPUT "TRUE
  OUTPUT "FALSE
END
```

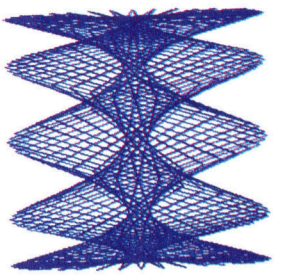
```
TO POS :COEFF1 :COEFF2 :ANGLE
  MAKE "X 100 * SIN ( :COEFF1 * :ANGLE)
  MAKE "YT100 * COS ( :COEFF2 * :ANGLE)
  SETXY :X :Y
END
TO LJ1 :COEFF1 :COEFF2 :ANGLE :STEP
  POS :COEFF1 :COEFF2 :ANGLE
  LJ1 :COEFF1 :COEFF2 ( :ANGLE
     + :STEP ) :STEP
END
```

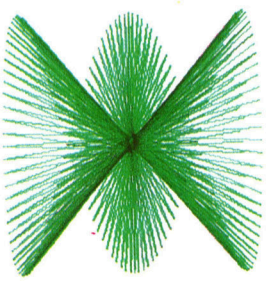
Grafische Figuren

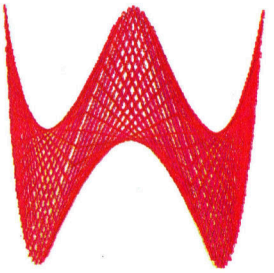












Antworten

1. Spielsteuerung über Tastatur: Ändern: SET.DEMONS WATCH, CHECK. Löschen: JOYH. MOVE und READKEY einsetzen.

```
TO SET.DEMONS
  WHEN OVER :SHEEP1 :FENCE [SETSP 0]
  WHEN OVER :SHEEP2 :FENCE [SETSP 0]
  WHEN TOUCHING :SHEEP1:SHEEP2
     [SETSP 0]
  WHEN TOUCHING :DOG :SHEEP1 [SETSP 0]
  WHEN TOUCHING :DOG :SHEEP2 [SETSP 0]
END
TO WATCH
  MOVE READKEY
  IF :SPEED = 0 [CHECK]
  WATCH
END
TO CHECK
  IF COND OVER :SHEEP1 :FENCE [ASK
     :SHEEP1 [BK 10 RT 90]]
  IF COND OVER :SHEEP2 :FENCE [ASK
     :SHEEP2 [BK 10 RT 90]]
  IF COND TOUCHING :SHEEP1 :SHEEP2 [BUMP]
  IF COND TOUCHING :DOG :SHEEP1 [ASK
     :SHEEP1 [RT 90]]
  IF COND TOUCHING :DOG :SHEEP2 [ASK
     :SHEEP 2 [RT 90]]
  SET.SPEEDS
END
TO MOVE :COM
  IF :COM = "W [ASK :DOG [SETH 0]]
  IF :COM = "S [ASK :DOG [SETH 90]]
  IF :COM = "Z [ASK :DOG [SETH 180]]
  IF :COM = "A [ASK :DOG [SETH 270]]
  IF :COM = "Q [ASK :TURTLE [DRAW.CAGE]]
END
  TO READKEY
  IF KEYP [OUTPUT RC]
  OUTPUT "
END
```

2. Raumschiff gegen Meteoriten

```
TO PLAY
  CS FS
  SET 0 1 [—100 80] 180 199
  SET 1 1 [0 80] 180 199
  SET 2 1 [100 90] 180 199
  SET 3 2 [0 —80] 90 50
  SET.DEMONS
  RANDOM.MOVE 0
END
TO SET :NO :SHAPE :POS :HEAD :SP
  TELL :NO SETSH :SHAPE
  PU SETPOS :POS
  SETH :HEAD ST SETSP :SP
END
TO SET.DEMONS
  WHEN TOUCHING 0 3 [BANG]
  WHEN TOUCHING 1 3 [BANG]
  WHEN TOUCHING 2 3 [BANG]
  WHEN 15 [JOYH]
END
TO BANG
  TELL [0 1 2 3]
  SETSP 0 SS
  PRINT " PRINT "
  PRINT "SPLATTERED
END
TO JOYH
  IF (JOY 1) < 0 [STOP]
  ASK 3 [SETH 45 * JOY 1]
END
TO RANDOM.MOVE :NO
  IF SPEED = 0 [( PRINT "SCORE :NO ) STOP]
  ASK RANDOM 3 [SETH 145 + RANDOM 70]
  RANDOM.MOVE :NO + 1
END
```

Acorn Electron

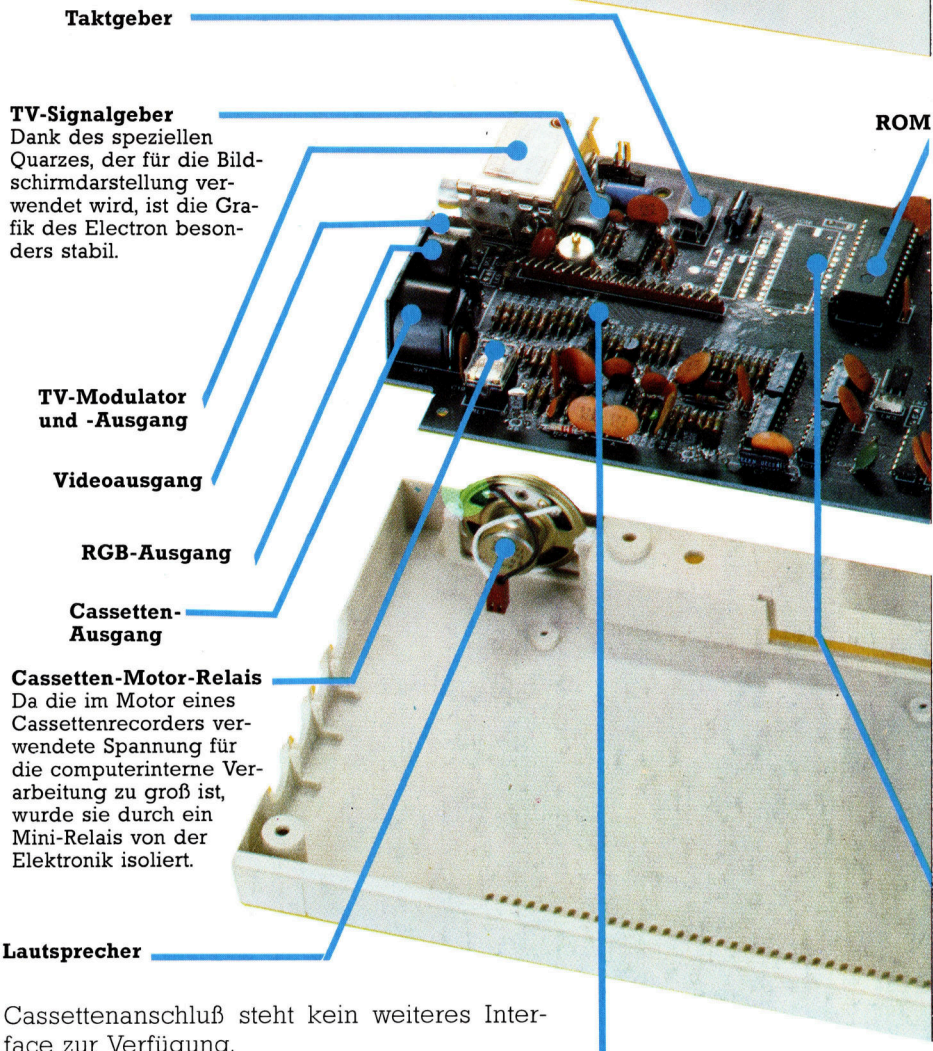
Zwei Jahre liegen zwischen der Einführung des Acorn B und des Electron. Die Microcomputer-Technologie hat sich in diesem Zeitraum erheblich weiterentwickelt.

Der Acorn Electron ist ein eleganter Computer, der den Eindruck einer robusten und gut gestalteten Maschine vermittelt. Wenngleich er als heruntergefahrene Version des Acorn B leistungsmäßig nicht so überzeugend ist, macht er doch einen komfortablen Eindruck. Die meisten Fähigkeiten des Acorn B finden sich auch beim Electron wieder. So etwa wird auf beiden Rechnern der SOUND-Befehl in Verbindung mit dem Befehl ENVELOPE verwendet, um unterschiedliche Klang-Arten von Musikinstrumenten künstlich darzustellen.

Mit Ausnahme des MODE 7 (Teletext), der beim Modell B von einem speziellen Chip generiert wird, verfügt der Electron über alle Grafik-Modi. Besagter Chip ist nicht in die Electron-Platine integriert. Teletextähnliche Darstellungen können nur durch Neudefinition der meisten Charaktere und Imitierung von Teletext in MODE 6 erzeugt werden.

Die Ein- wie Ausgabe-Möglichkeiten sind ebenfalls reduziert. Die optische Ausgabe erfolgt über den Fernseher oder über monochrome und Farbmonitore. Doch außer einem Cassettenanschluß steht kein weiteres Interface zur Verfügung.

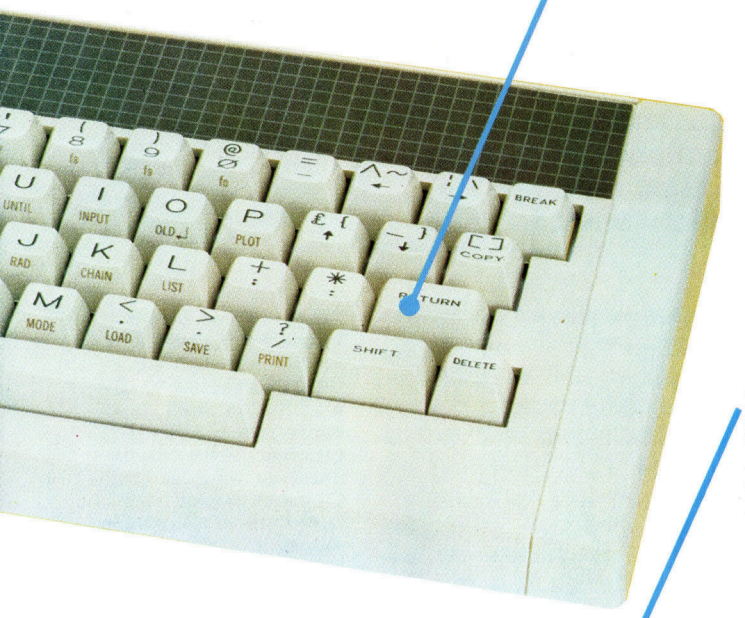
Das dynamische Duo
Chris Curry (links) und Herman Hauser (rechts) waren für die Konstruktion des Acorn B sowie des Electron verantwortlich. Curry arbeitete als Entwicklungsingenieur für Clive Sinclair und stellte Hauser ein. Gemeinsam gründeten sie Acorn.



Erweiterungen sind über eine lange, auf der Rückseite des Rechners befindliche Steckleiste möglich, die – sofern ungenutzt – lediglich durch eine Plastikabdeckung geschützt ist. An diese Leiste läßt sich die Interface-Box „Plus 1" anschließen, die mit Centronics-Schnittstelle, Cartridge-Slot und Analog/Digital-Wandler ausgestattet ist. Eine zweite Interface-Box enthält die speziell für den Electron entwickelte Diskettenstation.

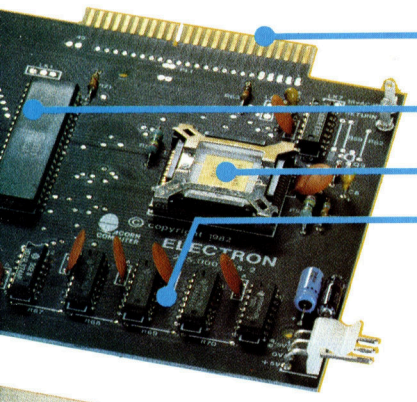
Integriert wurde das vom Acorn B bekannte

Tastatur-Verbindung
Die Zahl der Pins (22) zeigt, daß die Keyboard-Ausgabe nicht in ASCII decodiert wird. In diesem Fall stünden nur 10 Pins zur Verfügung (acht für Daten, dazu je eines für 5 V und Erdung).



Tastatur
Sie gehört zu den besten, die für Heimcomputer verwendet werden, und entspricht einer hochwertigen Schreibmaschinentastatur. Sie ist fast identisch mit der des Acorn B. Zwar verfügt sie nicht über separate Funktionstasten, aber durch gleichzeitiges Drücken der „Caps Low"-Taste und einer Zifferntaste können spezielle Funktionen erfüllt werden. Diese Möglichkeit bieten auch die Buchstabentasten und drei der Tasten für Interpunktion.

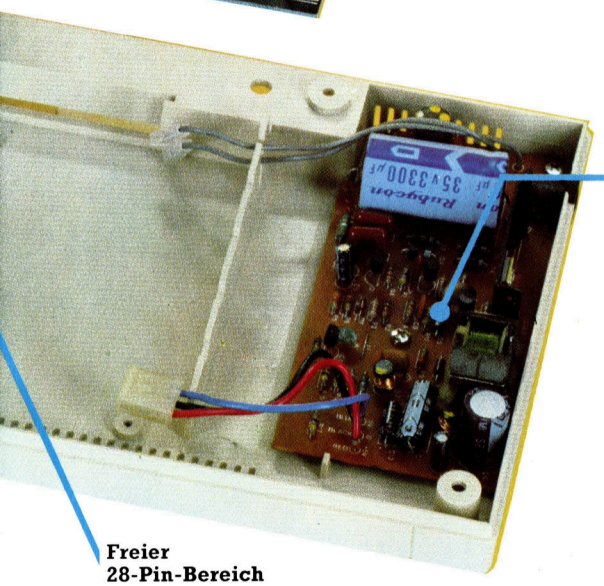
Erweiterungs-Steckleiste
Über diese Steckleiste wird der Electron mit der Interface-Box „Plus 1" verbunden.



CPU
Herz des Rechners ist ein herkömmlicher 6502 A-Chip, auf 1,79 MHz getaktet. Die „Entscheidungen" finden hier statt.

Uncommitted Logic Array
Dies ist die größte je hergestellte ULA. Neben der ULA, der 6502 CPU, dem RAM und dem ROM stehen nur neun weitere Chips auf der Platine zur Verfügung. Diese nutzen durchweg den TTL-Logik-Standard.

RAM
Die Bytes werden aus dem RAM halbweise in die CPU geladen. Dabei gelangen zuerst die niederwertigen vier Bits hinein (jeweils ein Bit von jedem der vier Chips), danach die höherwertigen vier. In den meisten Rechnern werden alle acht Bits eines Byte in einem Chip gespeichert.



Stromversorgung
Der Acorn Electron benötigt die ungewöhnliche Betriebsspannung von 19 Volt.

Freier 28-Pin-Bereich
Hier kann ein zusätzliches ROM oder ein besonderer Chip eingebaut werden.

und bewährte BASIC, das jedoch beachtlich erweitert wurde und somit Vorteile bietet, die den Umgang mit dem Rechner zum Vergnügen werden lassen. Besonders nützlich ist die OSCLI-Routine, mit der die Überleitung von BASIC-Programmbefehlen direkt ins Betriebssystem möglich ist. Erfahrene User können sich damit unschwer aus den Zwängen des BASIC befreien. Dazu wurde das Assembler-Paket des Acorn-B-BASIC um zusätzliche Befehle zur Variablen-Definierung und -Speicherung sowie zur String-Behandlung erweitert.

In der Praxis liegt der Electron weit über dem Durchschnitt: Das Bild ist scharf, die Farben sind klar und sauber.

Acorn Electron

PREIS
ca. 649 DM

ABMESSUNGEN
340x160x55 mm

ZENTRALEINHEIT
6502

TAKTFREQUENZ
1,79 MHz

MASCHINEN-SPEICHER
64 KByte ROM; 32 KByte RAM (keine interne Erweiterungsmöglichkeit)

BILDSCHIRM-DARSTELLUNG
32 Zeilen mit 80 Zeichen. Acht Farben, Vordergrund und Hintergrund unabhängig voneinander wählbar. 127 definierte Zeichen, weitere 255 vom Anwender definierbar.

SCHNITTSTELLEN
Fernsehanschluß, Video- und TTL-RGB-Anschluß, Cassette, Systembus für Erweiterungen.

PROGRAMMIER-SPRACHE
BBC BASIC mit integriertem Assembler

WEITERE PROGRAMMIER-SPRACHEN
Zur Verfügung stehen FORTH und LISP von Acorn Soft, die allerdings als RAM geliefert werden. Auf ROM basierende Sprachen wie BCPL und PASCAL sind mit dem Rechner in der Grundversion nicht kompatibel.

ZUBEHÖR
BASIC-Handbuch, Antennenkabel, Netzgerät, Einführungscassette.

TASTATUR
56 Schreibmaschinentasten. BASIC-Befehlseingabe über Tastatur. 10 frei programmierbare Funktionstasten.

DOKUMENTATION
Hervorragend. Ausführliche Informationen für den Anfänger wie den fortgeschrittenen Programmierer. Alle BASIC-Befehle sind einzeln erläutert. Außerdem ein umfangreicher Assembler-Teil, in dem besonders auf das interne Assembler eingegangen wird. Dank dieser Aufbereitung sind mit dem Rechner zahlreiche Aufgaben relativ leicht lösbar.

In die Hit-Parade

Viele Computerbesitzer sind von der Vorstellung fasziniert, zigtausend Mark mit dem Schreiben von Computerspielen verdienen zu können, wie es eine Reihe Jugendlicher bereits getan hat. Hier erfahren Sie, wie mit einer neuen Programmidee viel Geld verdient werden kann.

Im Sommer 1984 veröffentlichte Gremlin Graphics ein neues Computer-Spiel mit dem Titel „Wanted: Monty Mole". Auf Anhieb versprach das Spiel, ein Bestseller zu werden. Und damit waren dem Programmierer Peter Harrap die bewußten zigtausend Mark sicher. Harrap war gerade erst 19, und „Wanted: Monty Mole" war sein erstes kommerzielles Programm.

Harrap ist wie die meisten Programmierer Autodidakt. Er fing mit einem Sinclair ZX 81 an und schrieb sein erstes Programm, nachdem er merkte, wie gering der Umfang der angebotenen Software für das System war. Voraussetzung dafür: Er mußte wegen der relativen Trägheit des BASIC die Maschinensprache des Z 80 erlernen. Später wandte sich Harrap dem ZX Spectrum zu, angeregt durch Quicksilvas Programm „Ant Attack". Harrap entschlüsselte den Code und modifizierte die dargestellte „Landschaft". Seine Variante wurde allerdings von Quicksilva abgelehnt. Ian Stewart, Inhaber eines kleinen Software-Unternehmens, suchte zu diesem Zeitpunkt nach neuen Talenten und war von dem Programm begeistert, das Harrap ihm zugesandt hatte. Da jedoch die Copyrights von Ant Attack verletzt worden wären, sandte Stewarts Firma, Gremlin Graphics, das Programm zurück. Das Unternehmen engagierte Harrap jedoch sofort.

Jeder Software-Hersteller sucht nach interessanten, ungewöhnlichen Programm-Themen, in der Hoffnung, damit einen Hit zu landen. So war angesichts des Bergarbeiterstreiks das Programm: „Wanted: Moty Mole" sehr erfolgversprechend. Harrap setzte die Idee für den ZX Spectrum um.

Er arbeitete vier Monate an dem Programm, für das er zunächst einen Sprite-Generator schrieb, um die verschiedenen Grafiken schneller realisieren zu können. Dann entstand das Bergwerk, und anschließend fügte er die Regeln ein, nach denen die Bewegungen des Akteurs zu steuern waren.

Gremlin Graphics bezahlt an Harrap eine sogenannte Royalty, einen bestimmten Prozentsatz vom Verkaufspreis jeder Cassette. Diese Royalty ist unterschiedlich, abhängig von der jeweiligen Software-Firma. Vertraglich wird meist eine gleitende Regelung vereinbart: Fünf Prozent für die ersten 3000 verkauften Exemplare und sieben Prozent für die nächsten 10 000, um ein Beispiel zu nennen. Der Prozentsatz basiert entweder auf dem Nettopreis, also dem, den der Hersteller von der Vertriebsfirma erhält, oder dem Endverkaufspreis.

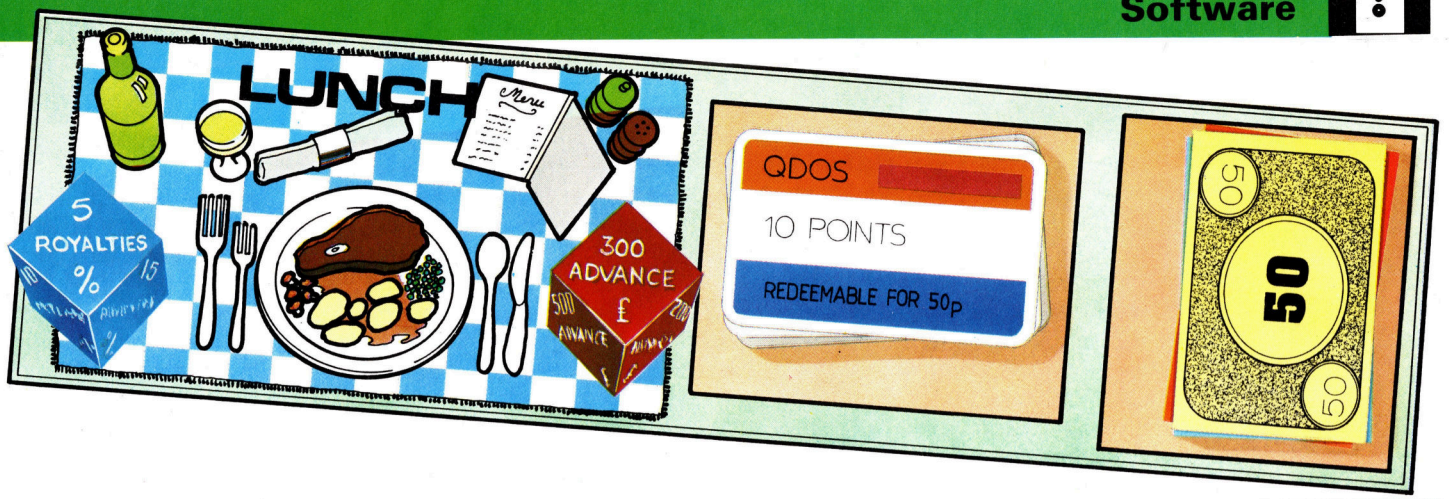
Die meisten Anbieter leisten an ihre Programmierer zudem eine a-conto-Zahlung, die gegen die Royalty verrechnet wird. Andere Gesellschaften zahlen lediglich eine Pauschale für das Programm. Programmierer sollten sich über diese Unterschiede in der Vertragsgestaltung im klaren sein, da das – hat ein Programm Erfolg – für sie mit finanziellem Verlust verbunden sein kann.

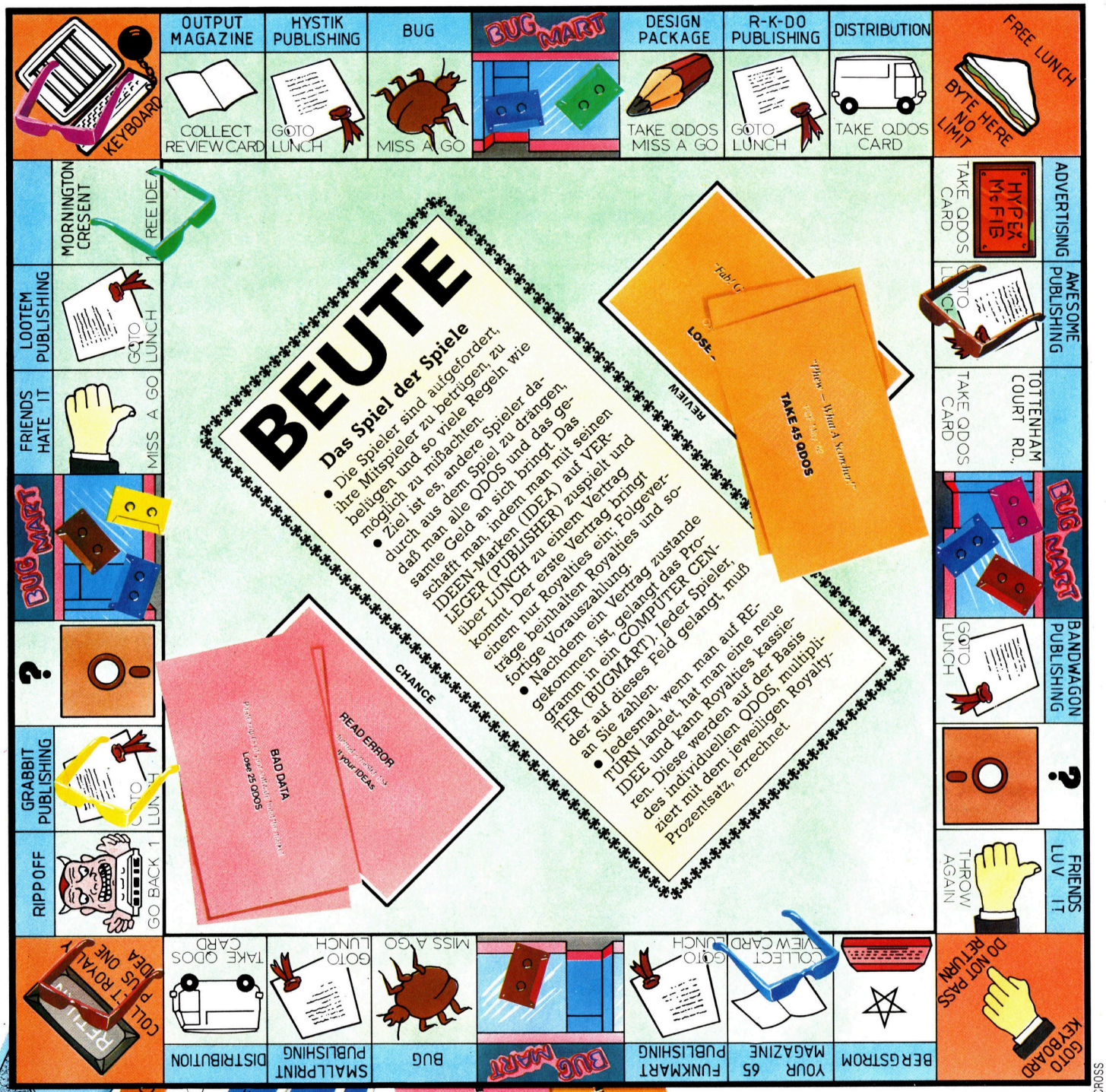
Bei einem Vertrag auf Royalty-Basis behält der Programmierer die Urheberrechte an seiner Software. Das bedeutet, daß der Programmierer bei Copyrightverletzungen beklagt werden bzw. klagen kann. Mit dem Erwerb eines Programms durch einen Softwareanbieter gegen einmalige Zahlung gehen die Copyrights an das Unternehmen über.

Über die Royalty hinaus erhält Harrap eine Art Fixum. Er hat allerdings einen einjährigen Exklusivvertrag mit Gremlin Graphics und darf für andere Software-Häuser nicht arbeiten. Viele Verträge enthalten überdies eine Optionsklausel, die dem Partnerunternehmen das Vorkaufsrecht einräumt und sogar – bei Ablehnung einer Idee – ein Vetorecht.

Tausende von Computerbesitzern schreiben eigene Software, um Geld zu verdienen. Einige finden auch Hersteller, die ihre Programme verwerten. In unserem Spiel ist die Gesamtsituation vielleicht etwas überzogen dargestellt, aber letztlich haben wir es doch mit einer Art von Lotterie zu tun . . .

Am wichtigsten für den Verkauf ist die Präsenz im Geschäft. Hier zeigt sich, ob Werbung, Rezensionen, Verpackung und Umfeld den Käufer positiv eingestimmt haben und ihn in seiner Kaufentscheidung beeinflussen. Lieferfähigkeit ist ein weiterer wichtiger Gesichtspunkt. Deshalb bedienen sich fast alle Software-Hersteller bewährter großer Vertriebsfirmen.


LUNCH
Menu
5
ROYALTIES
%
300
ADVANCE
£
QDOS
10 POINTS
REDEEMABLE FOR 50p
50
50


KEYBOARD
OUTPUT MAGAZINE
COLLECT REVIEW CARD
HYSTIK PUBLISHING
GOTO LUNCH
BUG
MISS A GO
BUG MART
DESIGN PACKAGE
TAKE QDOS MISS A GO
R-K-DO PUBLISHING
GOTO LUNCH
DISTRIBUTION
TAKE QDOS CARD
FREE LUNCH
BYTE HERE
NO LIMIT
ADVERTISING
TAKE QDOS CARD
HYPEX McFIG
AWESOME PUBLISHING
TOTTENHAM COURT RD.
TAKE QDOS CARD
BUG MART
BANDWAGON PUBLISHING
GOTO LUNCH
?
FRIENDS LUV IT
THROW AGAIN
GOTO KEYBOARD
DO NOT PASS RETURN
BERGSTROM
YOUR 65 MAGAZINE
COLLECT REVIEW CARD
FUNKMART PUBLISHING
GOTO LUNCH
BUG MART
BUG
MISS A GO
SMALLPRINT PUBLISHING
GOTO LUNCH
DISTRIBUTION
TAKE QDOS CARD
COLLECT ROYALTY
PAY FOR ONE IDEA
MORNINGTON CRESENT
FREE IDEA
LOOTEM PUBLISHING
GOTO LUNCH
FRIENDS HATE IT
MISS A GO
BUG MART
?
GRABBIT PUBLISHING
RIPPOFF
GO BACK 1
BEUTE
Das Spiel der Spiele
• Die Spieler sind aufgefordert, ihre Mitspieler zu betrügen, zu belügen und so viele Regeln wie möglich zu mißachten.
• Ziel ist es, andere Spieler dadurch aus dem Spiel zu drängen, daß man alle QDOS und das gesamte Geld an sich bringt. Das schafft man, indem man mit seinen IDEEN-Marken (IDEA) auf VERLEGER (PUBLISHER) zuspielt und über LUNCH zu einem Vertrag kommt. Der erste Vertrag bringt einem nur Royalties ein; Folgeverträge beinhalten Royalties und sofortige Vorauszahlung.
• Nachdem ein Vertrag zustande gekommen ist, gelangt das Programm in ein COMPUTER CENTER (BUGMART). Jeder Spieler, der auf dieses Feld gelangt, muß an Sie zahlen.
• Jedesmal, wenn man auf RETURN landet, hat man eine neue IDEE und kann Royalties kassieren. Diese werden auf der Basis des individuellen QDOS, multipliziert mit dem jeweiligen Royalty-Prozentsatz, errechnet
REVIEW
TAKE 45 QDOS
CHANCE
READ ERROR
BAD DATA
Lose 25 QDOS

STEVE CROSS

Als erfolgreich gilt ein Spiel, das mindestens 15 000mal während der Angebotszeit (das sind durchschnittlich vier bis fünf Monate) verkauft wird. Im Falle „Wanted: Monty Mole" ließ der Hersteller für den ZX Spectrum und den C 64 jeweils 10 000 Kopien anfertigen und sicherte sich die Option, weitere ziehen zu können. Bei Veröffentlichung des Programms prognostizierte Ian Stewart 20 000 abgesetzte Exemplare pro System und äußerte die Überzeugung: „Das wird die Nummer eins!"

Weit früher hatte Christopher Kerry aus Sheffield Erfolg. Mit 17 verließ er kurz vor dem Abschluß die Schule, um als freier Mitarbeiter für das Software-Unternehmen House of Thor tätig zu sein. Kerry schrieb das Spiel „Jack and the Beanstalk" auf dem ZX Spectrum. Ohne Werbung und nach nur zweimonatigem Angebotszeitraum wurden nach Auskunft des Herstellers, der noch keine offiziellen Zahlen hat, „Zigtausende" verkauft. Das House of Thor übte sich, was Kerry anbetraf, in Zurückhaltung, da man fürchtete, er würde durch einen anderen Anbieter abgeworben werden. Neben Kerry sind weitere 15, zumeist jugendliche Programmierer für das House of Thor tätig.

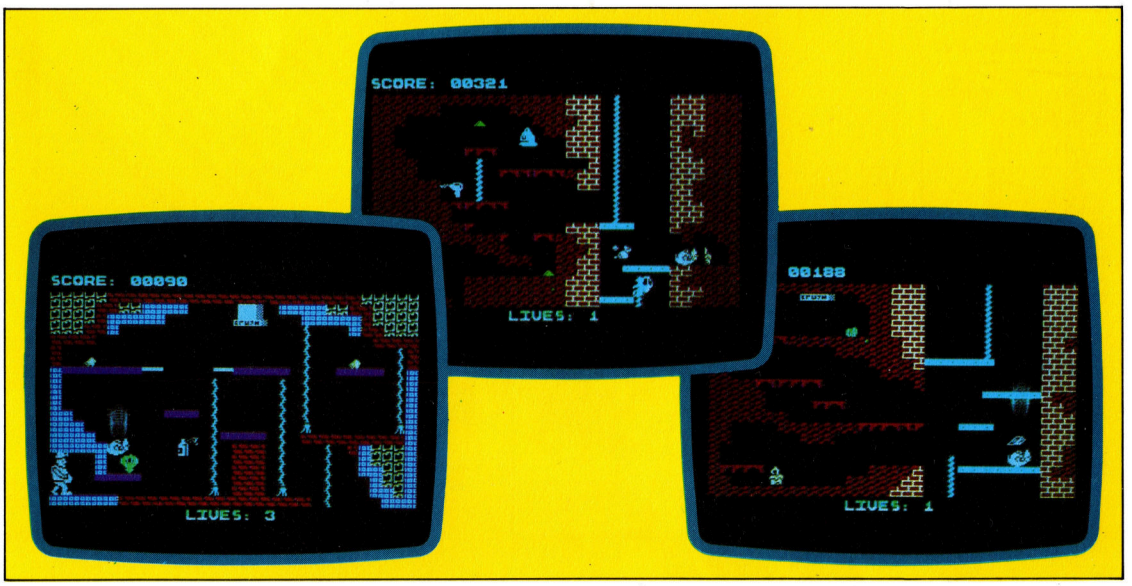
Originalprogramme gefragt

Wie alle Software-Unternehmen ist auch diese Firma an Programm-Einsendungen interessiert. Besteht ein generelles Interesse an einem Programm, erfolgt die Kontaktaufnahme noch am Tag des Posteingangs. Das ist für die Unternehmen lebenswichtig, da man davon ausgeht, daß Konkurrenten die Programme ebenfalls zur Begutachtung zugesandt bekommen haben. Beim House of Thor besteht nur Interesse an Actionspielen, die in Maschinensprache geschrieben sind. Philosophie des Hauses ist es, ausschließlich Originalprogramme zu veröffentlichen. Plagiate älterer Spiele haben also keine Chance. Neben möglichen urheberrechtlichen Schwierigkeiten sieht man für Nachahmungen keine oder nur geringe Verkaufschancen.

Salamander mit fast 40 Programmierern auf der Gehaltsliste gehört zu den größeren Unternehmen dieser Art. Chris Holland, der Boss des Hauses, rät potentiellen Spieleautoren: „Eine Riesenchance hat, wer Programme für ganz neue Systeme schreibt. Wir sind besonders an Software für den Amstrad (= Schneider CPC 464), den Atmos und die MSX-Rechner interessiert. Schwieriger ist die Sache beim Spectrum. Interesse haben wir dann, wenn eine völlig neue Idee zugrundeliegt." Salamander führt über 50 Titel für alle populären Heimcomputersysteme, und das sind nicht nur der Spectrum und der C 64.

Statt Programme an Software-Unternehmen zu verkaufen, versuchen andere Programmierer, eigene Software-Häuser zu gründen. Beispiele dafür sind Llamasoft und Bug-Byte. Dazu ist jedoch viel Kapital erforderlich. Ian Stewarts „Gremlin Graphics" z. B. wurde mit einem Programm, „Potty Pigeon", gestartet. Ende 1984 waren dann fünf Spiele im Angebot. Doch die vier Inhaber mußten einiges Kapital in das Unternehmen stecken, um Werbung und Vertrieb finanzieren zu können. Eine Vierfarbanzeige in einem Computermagazin kostet einschließlich der technischen Produktion rund 4500 Mark. Und selbst eine Reihe erfolgreicher Spiele sind noch keine Garantie für finanzielle Sicherheit. Mit Schulden in Höhe von über vier Millionen Mark meldete „Imagine" im Juli 1984 Konkurs an, und das, obwohl das Unternehmen monatliche Umsätze in Höhe von etwa 1,2 Millionen Mark verbuchen konnte.

Die Gründung eines eigenen Software-Hauses ist risikoreich: Neben exzellenten Titeln ist gute finanzielle Beratung wichtig. Ein Programm „zur Begutachtung" einsenden indes kann jedermann. Vielleicht haben Sie ja den nächsten Hit schon längst in der Schublade.



Der Programmierer, der „Wanted: Monty Mole" schrieb, ist der typische Programmautor: jung, clever und gut bezahlt. Das Spiel selbst ist völlig atypisch. Es basiert auf dem Bergarbeiterstreik von 1984 und richtet sich gegen die Gewerkschaft. Doch die Bergarbeitergewerkschaft erhält fünf Pence von jeder verkauften Programmcassette.

Abläufe in der CPU

Diese Folge über den Maschinencode beschäftigt sich mit den Funktionen der CPU. Ein Listing für den Acorn B und eins für den Commodore 64 werden Ihnen dabei helfen, die Vorgänge in der Zentraleinheit leichter zu verstehen.

Da die Zentraleinheit eines Computers eine wichtige Funktion erfüllt, wollen wir untersuchen, wie sie arbeitet und welche Funktionen sie ausführt.

Stellen Sie sich eine einfache elektrische Schaltung vor, die aus einer Batterie, einem Schalter und einer Glühbirne besteht. Wird der Schalter geschlossen, geht das Licht an und leuchtet so lange, bis die Batterie leer ist oder der Schalter wieder geöffnet wird. Der Zustand der Glühbirne – AN oder AUS – ist dabei eine Information, während die Schaltung diese Information speichert. Nehmen wir an, der Schalter ist neben dem Eingang einer Fabrik installiert und die Glühbirne befindet sich im Büro des Managers. Wenn der erste Angestellte an das Fabriktor kommt, betätigt er den Schalter. Der Chef in seinem Büro sieht, daß die Glühbirne leuchtet, und weiß, daß jemand zur Arbeit erschienen ist. Der Manager braucht nicht einmal in seinem Büro zu sein, wenn das Licht angeht. Er kann anhand der Glühbirnenschaltung jederzeit erkennen, ob jemand angekommen ist, sofern diese Informationen gespeichert wurden.

Auf fast die gleiche Weise speichert ein Computer die Daten: Alle Informationen werden auf den Zustand elektrischer Schaltungen zurückgeführt. Der Vorgang ist natürlich etwas komplizierter, und daher werden wir unser Informationssystem noch ein wenig verfeinern.

Informationssprache

Bei der Verwendung von vier Stromkreisen (vier Schalter in einer Reihe neben der Eingangstür und die Glühbirnen in der entsprechenden Reihenfolge im Büro des Managers) leuchtet beim Betätigen des linken Schalters auch die linke Glühbirne auf etc. Jedem Angestellten wird mitgeteilt, die Schalter nur auf eine individuell bestimmte Weise zu betätigen, bei der z. B. Katrin den ersten und zweiten Schalter schließt, den dritten und vierten aber öffnet, Richard den vierten schließt und alle anderen Schalter öffnet und Klaus den ersten und dritten schließt, den zweiten und vierten öffnet und so weiter für alle Angestellten. Die optischen Signale im Büro des Managers zeigen jetzt genau an, welcher Angestellte angekommen ist.

Wenn wir die Position AUS als 0 bezeichnen und die Position AN als 1, dann hat Katrin die Schalter auf 1100 (die ersten beiden Schalter AN und den dritten und vierten AUS) gestellt, Richard auf 0001 (den vierten Schalter auf AN und alle anderen auf AUS) und Klaus auf 1010 (den ersten und dritten AN und den zweiten und vierten AUS). Interpretiert der Manager eine eingeschaltete Glühbirne ebenfalls als 1 und eine ausgeschaltete als 0, dann „sprechen" sowohl er als auch die Angestellten die gleiche Informationssprache. „0001" bedeutet für alle dabei eindeutig „Richard".

Wie viele eindeutige Kombinationen lassen sich mit den Schaltern herstellen? Es gibt vier Schalter, die in je zwei Stellungen stehen können, also 2×2×2×2=16 unterschiedliche Kombinationsmöglichkeiten:

0000,0001,0010,0011,0100,0101,0110,0111,
1000,1001,1010,1011,1100,1101,1110,1111

So läßt sich das konkrete Modell der Glühbirnen auf abstrakte Folgen von Einsen und Nullen übertragen. Wenn man in der Abstraktion nur ein wenig weiter geht, lassen sich diese Folgen in Zahlen umwandeln.

Stellen Sie sich vor, Sie zählen und schreiben dabei die Zahlen auf. Von Null bis Neun geht es schnell, da jede dieser Zahlen einen eigenen Namen hat und von einem eigenen Symbol dargestellt wird. Was passiert aber, wenn Sie Zahlen über Neun aufschreiben wollen? Die Zahl hat zwar einen eigenen Namen – Zehn –, läßt sich aber nicht mit einem einzigen Symbol darstellen. Aus diesem Grunde setzen Sie sie aus den bereits geschriebenen Symbolen zusammen: 10,11,12 etc. bis 99. Jetzt sind auch alle zweistelligen Kombinationen erschöpft und die nächste – dritte – Stelle wird eröffnet (100). Dieser Vorgang erscheint trivial, doch denken Sie daran, wie viele Schwierigkeiten gerade die Kästchen mit den Hundertern, Zehnern und Einern einem Schulanfänger bereiten können.

Unser Zahlensystem beinhaltet 10 Ziffern (0,1,2,3 . . .9), aus denen wir alle Zahlen zusam-

mensetzen. Wie aber funktioniert das Zählen, wenn man nur über zwei Symbole verfügen kann: 0 und 1? Bis eins können wir leicht zählen, wie aber läßt sich die nächste Zahl darstellen? Da keine weiteren Symbole zur Verfügung stehen, müssen die vorhandenen wie bei dem Zehnersystem zu Kombinationen zusammengesetzt werden. Die auf eins folgende Zahl ist also 10. Die nächste Zahl heißt Drei und wird in Form von 11 geschrieben. Da jetzt alle zweistelligen Kombinationsmöglichkeiten erschöpft sind, muß die Zahl Vier dreistellig (= 100) sein, die Fünf (101), die Sechs (110) und die Sieben (111) ebenfalls. Wir zählen im Dezimalsystem

Diese zwei kurzen Programme für den Acorn B und den Commodore 64 bringen den vorhandenen Zeichensatz auf den Bildschirm und zeigen die Unterschiede zwischen BASIC und Maschinencode.

Acorn B

```
100 REM*********************************
**********************BBC****************
149 REM*****************************
150 REM*      BBC M/C CODE DEMO     *
151 REM*****************************
200 MODE 4:*TV 254
300 GOSUB 30000
700 FOR P=1920 TO 6079
800 K=K+1:IF K>2679 THEN K=1920
900 ?(HIMEM+P)= (K+47232)
1000 NEXT P
1100 PRINT TAB(13);"THAT WAS BASIC"
1200 INPUT" HIT RETURN FOR MACHINE CODE
VERSION ",A$:CLS
1300 FOR L=0 TO 15:FOR B=0 TO 255 STEP L
1400 ?(SA)=LS:?(SA+1)=HS
1500 ?(FA)=LF:?(FA+1)=HF
1600 DUMMY=USR(PSTRT)
1700 VDU 30
1800 NEXT B,LP
1900 STOP
30000 REM******MC LOADER S/R****
30010 K=1919:PSTRT=PAGE+8:VSTR=HIMEM+192
0
30020 HS=INT(VSTR/256):LS=VSTR-256*HS:LF
=LS+56:HF=HS+2:SA=114:FA=116
30100 DATA 50,169,32,197,112,48,4,240,2,
133,112,165,112,32,227,255
30110 DATA 230,114,208,2,230,115,165,116
,197,114,208,7,165,117
30120 DATA 197,115,208,1,96,230,112,169,
128,197,112,208,224
30130 DATA 169,32,133,112,208,218,96,96,
96
30150 READ ZZ
30160 FOR BY=PSTRT TO PSTRT+ZZ
30170 READ MC:?(BY)=MC
30180 NEXT BY
30200 RETURN
30299 REM*****************************
30300 REM*   SAVE BEFORE RUNNING !! *
30301 REM*****************************
30399 REM*****************************
30400 REM*DO NOT LIST LINE 100 AFTER*
30401 REM*      RUNNING PROGRAM !!   *
30402 REM*****************************
```

Commodore 64

```
99 REM ************************
100 REM*COMMODORE M/C CODE DEMO *
101 REM************************
200 PRINT CHR$(147)  :REM CLEAR SCREEN
300 PRINT "    THIS WON'T TAKE LONG"
400 GOSUB 60000
500 PRINT CHR$(147);CHR$(5) :REM CLS AND
 WHITE
600 CC=0
700 FOR P=SM TO FM
800 POKE P,CC:POKE P+OF,CL
900 CC=CC+1:IF CC>255 THEN CC=0
1000 NEXT P
1100 PRINT TAB(13);"THAT WAS BASIC"
1200 INPUT" HIT RETURN FOR MACHINE CODE
VERSION ";A$
1300 FOR LP=1 TO 9:FOR B=0 TO 255 STEP L
P
1400 POKE SA,LS:POKE SA+1,HS
1500 POKE FA,LF:POKE FA+1,HF
1600 POKE BA,B:POKE CH,0
1700 SYS AA
1800 NEXT B,LP
1900 STOP
60000 REM******MC LOADER S/R****
60010 SM=256*PEEK(648):OF=55296-SM:FM=SM
+999:BD=53280:SC=BD+1:CS=8:CB=6:CL=0
60020 POKE BD,CB:POKE SC,CS
60030 LS=0:HS=PEEK(648):LF=232:HF=HS+3:S
A=251:FA=253:BA=250:CH=2
60100 DATA 850,885,169,0,170,165,250,133
,2,165,2,129,251
60110 DATA 230,251,208,2,230,252,165,253
,197,251,208,7,165,254
60120 DATA 197,252,208,1,96,230,2,208,22
9,240,223
60150 READ AA,ZZ
60160 FOR BY=AA TO ZZ
60170 READ MC:POKE BY,MC
60180 NEXT BY
60200 RETURN
60299 REM*****************************
60300 REM* SAVE THIS BEFORE RUNNING IT*
60301 REM*****************************
```

(Null, Eins, Zwei etc.), schreiben die Zahlen im binären System aber so: 0, 1, 10, 11, 100, 101, ...

Nach genau der gleichen Methode, in der eine Dezimalzahl wie 152 eigentlich (1×100) + (5×10) + (2×1) darstellt, bedeutet die Binärzahl 101: (1×4) + (0×2) + (1×1). Statt Stellen mit Hundertern, Zehnern und Einern für unsere Zahlen zu verwenden, bedeuten die Stellen im Binärsystem Vierer, Zweier und Einer. Bewegt man sich in einer Dezimalzahl von rechts nach links, erhöht sich der Wert jeder Stelle fortlaufend um zehn. Bei einer Binärzahl erhöhen die Stellen sich jeweils um zwei.

Erinnern wir uns noch einmal kurz an unsere Analogie der Fabrikangestellten, die Schalter nach einem individuellen Muster zu betätigen. Die Schalterstellungen lassen sich dabei als vierstellige Binärzahlen ansehen. Das Signal von Katrin ist dann binär 1100 oder dezimal 12. Richards Signal ist binär 0001 (dezimal 1) und das Signal von Klaus binär 1010 (dezimal 10).

Schaltmuster

Wenn der Manager jetzt die Schaltmuster der Glühbirnen ansieht, kann er sie als binäre Zahlen verstehen, in ihre dezimale Entsprechung umwandeln und nachsehen, welchem Namen die Zahl entspricht.

Unser Modell zeichnet ein einfaches Bild, wie in einem Computer Informationen dargestellt werden. Für den Computer existieren nur Folgen von elektrischen Impulsen (Lichter, die den Zustand AN oder AUS darstellen); für uns Menschen ist es aber einfacher, diese Impulsfolgen als Binärzahlen anzusehen. Wenn Sie jetzt daran denken, daß der Code 1010 dem Namen „Klaus" entspricht, dann verstehen Sie, wie Zahlen in der Maschinensprache eingesetzt werden können. In unserer nächsten Folge werden wir untersuchen, wie ein Heimcomputer Informationen durch Binärzahlen darstellt.

Fachwörter von A bis Z

Background = Hintergrund
Ein professionelles Textverarbeitungssystem belegt – selbst bei kontinuierlichem Schreiben – nur etwa zehn Prozent des Leistungsvermögens Ihres Computers. Während Sie kaum mehr als zehn Anschläge pro Sekunde schaffen, könnte der Rechner das Zehnfache bewältigen (Tastaturabfrage, Zeicheninterpretation, RAM-Speicherung und Bildschirmwiedergabe). Deshalb hängt der Rechner rund 90 Prozent der Zeit nur in einer Warteschleife, bis Sie die nächste Taste drücken.

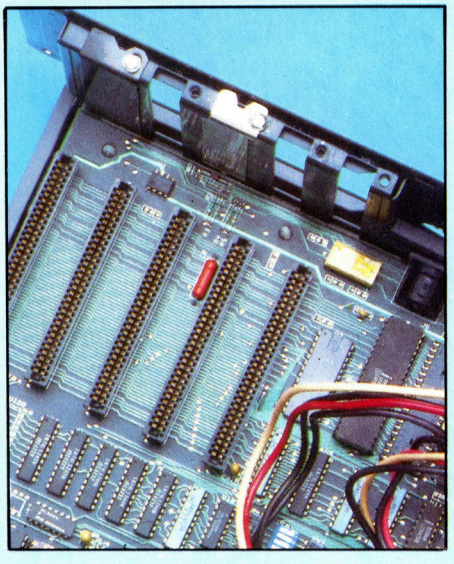
Die Auslastung des Computers wäre erheblich besser, wenn die Zentraleinheit in den Wartepausen andere Aufgaben erledigen könnte. – Darauf basiert die „Hintergrund"-Programmierung: Der Rechner arbeitet im „Vordergrund" (foreground) im Dialog mit dem Benutzer, während im Hintergrund ein anderes Programm abläuft.

Dies bedeutet aber auch, daß nur bestimmte Aufgaben für den Hintergrund geeignet sind. Beispielsweise erfordern Sortierprogramme viel Rechenzeit, obwohl die ständige Mitwirkung des Benutzers nicht notwendig ist. So auch beim Druckvorgang: Im Hintergrund wird ein Dokument ausgedruckt, während Sie schon am nächsten Text arbeiten.

Zeitintensive Aufgaben wie das Sortieren laufen im Hintergrund nicht ganz so schnell ab wie bei völlig freiem Rechner, denn die meisten Vorder/Hintergrund-Systeme arbeiten nach dem Zeitanteilverfahren (Timesharing): Der Prozessor verwendet beispielsweise 1/100 Sekunde auf den Vordergrund, dann 1/100 Sekunde auf den Hintergrund usw. Bei Großrechnern werden auf diese Weise im „Multiprogramming" Hunderte von Benutzern zugleich bedient.

Nur bei wenigen Kleinrechnern ist das Standard-Betriebssystem für diese Arbeitsweise eingerichtet. Meist müssen Sie dazu spezielle Software (unter Namen wie „Multitasking" – oder „Concurrent"-System) kaufen, beispielsweise die Concurrent-Version des bekannten CP/M-Systems.

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.



Backplane = Rückwandplatine
Bei den meisten Heimcomputern sind heute alle Bausteine in einem geschlossenen Gehäuse untergebracht. Die ersten Kleinrechner sahen aber ganz anders aus. – Der Rechner hatte hinten eine frei zugängliche „Rückwandplatine" mit einer Anzahl Steckerleisten für den Anschluß von Leiterplatten und Peripheriegeräten.

Der Vorteil dieses Konzepts bestand nicht nur in der leichten Erweiterbarkeit (beispielsweise durch Speicher- oder Schnittstellenkarten), sondern auch in der Möglichkeit, den Computer mehr oder weniger selbst ausbauen zu können.

Einiges von diesem System ist bei modernen Ausbildungs- und Bürorechnern übernommen worden. Beim „Tycom Microframe" ist man noch einen Schritt weitergegangen und überläßt dem Benutzer sogar die Wahl des Prozessors, ebenso die des RAM und der Schnittstellen.

Alle Anschlußstecker für die Rückwandplatine müssen die gleiche Stiftbelegung für die Bus-Kopplung aufweisen. Bei den ersten Maschinen war die „S-100"-Norm (mit 100poligem Stecker) sehr verbreitet.

Backup = (Sicherungs-) Duplikat
Die Bedeutung der Datensicherung bei Rechnern kann man nicht genug betonen. Unter einem „Backup" versteht man eine Reservekopie von einem Datenträger, die für den Fall der Beschädigung des Originals angefertigt wird. Solche Kopien sind bei Magnetträgern (Disketten und Cassetten) unerläßlich, weil trotz der verbesserten Zuverlässigkeit damit immer Pannen passieren können. Die Diskette kann verlorengehen, versehentlich mit Kaffee begossen oder auf einen starken Magnet gelegt werden, oder schlicht „dropouts" (Zeichenausfälle) bekommen, wenn sich mikroskopisch kleine Teilchen von der Magnetschicht lösen.

Bekanntlich können winzige Fehler oder Störungen eine ganze Datei wertlos machen. Wie oft Sie bei der Programmentwicklung regelmäßig Duplikate anlegen (wofür im Betriebssystem eine spezielle Routine vorgesehen ist), sollte davon abhängen, wie tragisch für Sie ein Verlust der betreffenden Datei wäre. Die meisten lernen das spätestens durch schmerzliche Erfahrungen, wenn der Rechner eine in vielen Stunden erstellte Datei plötzlich nicht wieder verarbeiten „will".

Bildnachweise
505, 517, 529: Steve Cross
506: Kevin Jones
507, 508, 509, 518, 519, 527, U3: Chris Stevens
514, 530: Liz Heaney
515: Liz Dixon
516: Tony Sleep, Microscope
522: Rolf Seiffe
524: Steve Malone
526: July Goldhill
528: Ian McKinnell

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

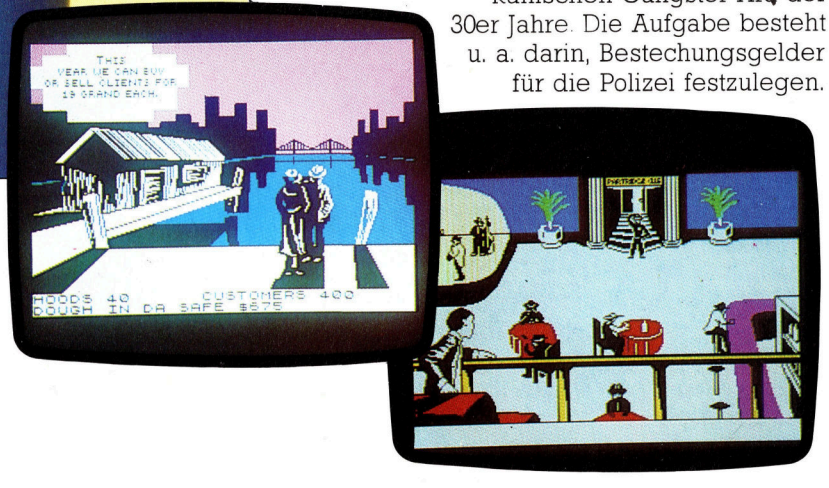
computer kurs Heft 20

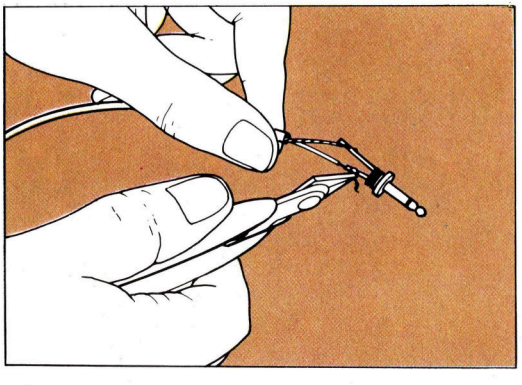
Der Commodore Plus/4

weist zum bekannten 64 einige entscheidende Verbesserungen auf. Besonders die Leistungsfähigkeit der Grafik- und Soundbefehle ist sehr hoch.

Mugsy

ist ein Strategie-Spiel. Der Spieler schlüpft in die Rolle eines Bandenchefs der amerikanischen Gangster-Ära der 30er Jahre. Die Aufgabe besteht u. a. darin, Bestechungsgelder für die Polizei festzulegen.





Kleine Reparaturen

sind oft mit großem Aufwand verbunden. Wir zeigen, wie es einfacher geht.

+++ Portable Sharp 5000 +++ Tips für die Praxis +++ „Daten-Karussell" +++ BASIC und LOGO +++ Robotersteuerung +++ Buchführungs-Programme +++ Der Speicheraufbau +++ Fachwörter +++