Einsteigen - Verstehen - Beherrschen

DM 3,80 öS 30 sfr 3,80

computer kurs

Heft 21

Ein wöchentliches Sammelwerk

Der Apple Macintosh

ASCII-Code und Hexzahlen

Elektronik-Gewehr

Pioniere der Computerindustrie

computer kurs

Heft 21

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

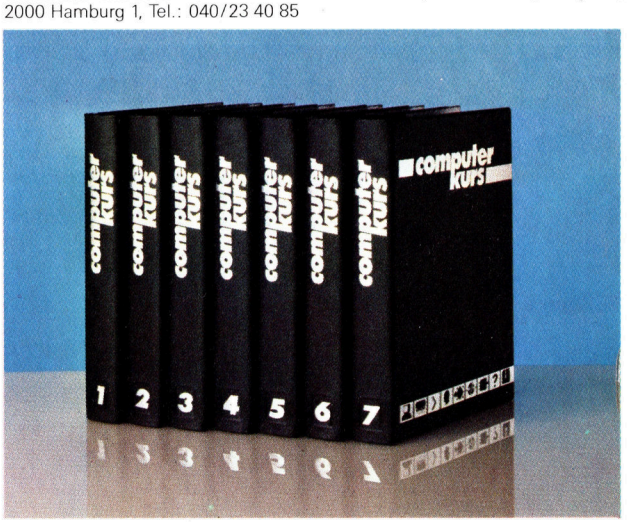
Schweiz: Der Sammelordner kostet sfr 15. Bitte wenden Sie sich an Ihren Kiosk; dort werden Sie jederzeit die gewünschten Exemplare erhalten.

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

Beispielhaft: Der Macintosh

Die gesamte Konstruktion des Apple Macintosh ist darauf ausgerichtet, dem EDV-Anfänger den Einstieg in die Welt des Computers zu erleichtern.

Der Macintosh baut auf der Technik des (teureren) Lisa auf und stellt mit seinem eingebauten Diskettenlaufwerk, integriertem Monitor, der Maus und dem leicht zu bedienenden Betriebssystem einen grundlegenden Fortschritt in der Konstruktion von Computern dar. Er läßt sich mit keinem der Microcomputer vergleichen, die wir bis jetzt vorgestellt haben. Statt sich in die Masse der Hersteller einzureihen, die ihre Maschinen dem IBM-Standard angeglichen haben, nahm Apple das Risiko auf sich, eine völlig neuartige Maschine für den kommerziellen Betrieb zu erschaffen. Mit diesem mutigen Schritt festigte Apple seinen Ruf als „Schöpfer neuer Geräte" inmitten einer Branche, die hauptsächlich Nachbauten erfolgreicher Maschinen produziert.

Schon das Gehäuse des Macintosh fällt aus dem Rahmen: Für eine Maschine mit derart vielen Möglichkeiten ist es ungewöhnlich schmal und kompakt. Der 9-Zoll-Monitor ist für hochauflösende Grafik ausgelegt. Die Diskettenstation wurde von Sony gebaut und arbeitet mit Disketten im 3 1/2-Zoll-Format. Die Maschine läßt sich noch in die Kategorie der tragbaren Computer einreihen, da sie inklusive Tastatur, Maus und dem (als Zusatzgerät erhältlichen) Tragekoffer nur zwölf Kilo wiegt.

Die „Maus"

Die Schreibmaschinentastatur des Macintosh ist leichtgängig und für Schnellschreiben geeignet. Sie hat einen eigenen Prozessor für Spezialfunktionen und die internationalen Zeichensätze. Ein besonders anwenderfreundliches Bedienungselement ist die „Maus". Dieses handliche Gerät ist kaum größer als eine Zigarettenschachtel. Es wird über eine flache Oberfläche bewegt und steuert damit die Bewegungen des Cursors auf dem Bildschirm. Mit Hilfe der Maus können Computerfunktionen auf einfachste Weise abgerufen werden. Da bei dieser Eingabemethode genaue Kenntnisse der Systemfunktionen nicht erforderlich sind, hat auch der Computer-Neuling kaum Probleme im Umgang mit dieser Maschine. Wenn Sie z. B. Text speichern wollen, brauchen Sie nur den Cursor mit der Maus auf das Bildschirmsymbol zu bewegen, das ein Blatt Papier darstellt. Mit dem Druck auf die in die Maus eingelassene Taste erscheint auf dem Bildschirm die entsprechende Arbeitsmöglichkeit. Nachdem Sie die gewünschten Daten über die Tastatur eingegeben haben, gehen Sie mit der Maus in das Hauptmenü zurück

Die Konstruktion des Macintosh ist darauf ausgerichtet, auf einem Schreibtisch den geringstmöglichen Platz einzunehmen. Die hohe Auflösung des Bildschirms läßt grafische Tricks zu.

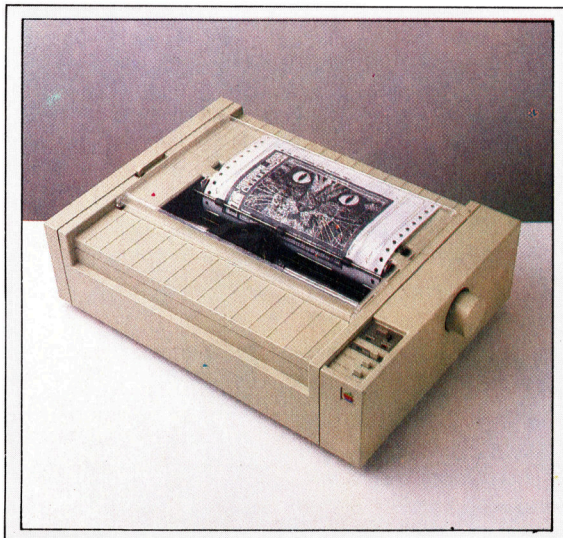
und können die Daten jetzt auf der Diskette speichern, indem Sie den Cursor auf das Diskettensymbol setzen.

Der Macintosh wird standardmäßig mit einem Arbeitsspeicher von 128 KByte geliefert, läßt sich aber auf 512 KByte ausbauen. Dabei werden die bestehenden RAM-Chips nur gegen Chips mit 256 KByte Kapazität ausgetauscht. Die 64 KByte ROM des Mac enthalten Systemsoftware, die fast alle Abläufe in der Maschine steuert und auch Spezialfunktionen ausführen kann. Mit den Sony-Laufwerken lassen sich auf den 3 1/2-Zoll-Disketten bis zu 400 KByte (auf einer Seite) abspeichern. Zudem sind diese Datenträger zuverlässiger als 5 1/4-Zoll-Disketten.

Beeindruckende Bildschirmgrafiken

Die Auflösung des Macintosh-Bildschirms beträgt 512x342 Pixel. Er arbeitet mit der Methode des „Bit Mapping" und kann daher alle 175 104 Punkte einzeln ansprechen. Mit dieser Fähigkeit lassen sich erstaunliche grafische Effekte erzielen. Abgesehen von den Spielmöglichkeiten sind die Grafiktricks des Macintosh auch für Grafiker, Architekten, Werbefachleute, Fotografen und viele andere Berufsgruppen interessant. Da der Mac speziell auf den Schnelldrucker „ImageWriter" abgestimmt ist, lassen sich all die beeindruckenden Bildschirmgrafiken problemlos exakt auf Papier übertragen.

Das Ungewöhnliche des Macintosh ist jedoch nicht seine Zuverlässigkeit und die hohe Qualität der Hardware, sondern die ausgefeilte Software, mit der die Hardware, das im ROM enthaltene Betriebssystem und die Programme hervorragend aufeinander abgestimmt werden. Programme, die auf anderen Computern entwickelt wurden, lassen sich leicht auf den Macintosh übertragen. Der Computer ist so „benutzerfreundlich", daß er ohne EDV-Kenntnisse sofort eingesetzt werden kann.



Bildschirmkopie
Der ImageWriter druckt seriell mit einer Geschwindigkeit von 120 Zeichen pro Sekunde. Besonders bei Grafik ist die Schnelligkeit von Vorteil. Mit der „Bit Mapping"-Methode lassen sich exakte Kopien des Bildschirminhaltes herstellen.

Analogplatine
Diese Platine steuert Bildschirm und Stromversorgung. Der Macintosh benötigt keinen Ventilator, da überschüssige Wärme über Metallplatten zu den Lüftungsschlitzen des Gehäuses geleitet wird.

Eingebauter Lautsprecher

Schreib-/Lesekopf für Disketten

Regler für Bildkontrast

3 1/2-Zoll-Diskettenstation von Sony
Dieses speziell für Apple gebaute Laufwerk speichert bis zu 400 K auf einer Diskettenseite.

Bildschirmspeicher
Ein Teil der für den Bildschirm benötigten Speicherkapazität liegt in dieser DMA (Direct Memory Access)-Schaltung für den direkten Speicherzugriff.

Tastatur
Die frei bewegliche Tastatur des Macintosh hat einen eigenen Prozessor, mit dem die internationalen Zeichensätze und Spezialfunktionen gesteuert werden.

Tastaturanschluß

Die „Maus"
Mit der Maus lassen sich die Bewegungen des Cursors steuern und Objekte aus dem Bildschirminhalt „auswählen".

Ausgang für A/D-Wandler
Hier können Analog/Digital-Platinen angeschlossen werden.

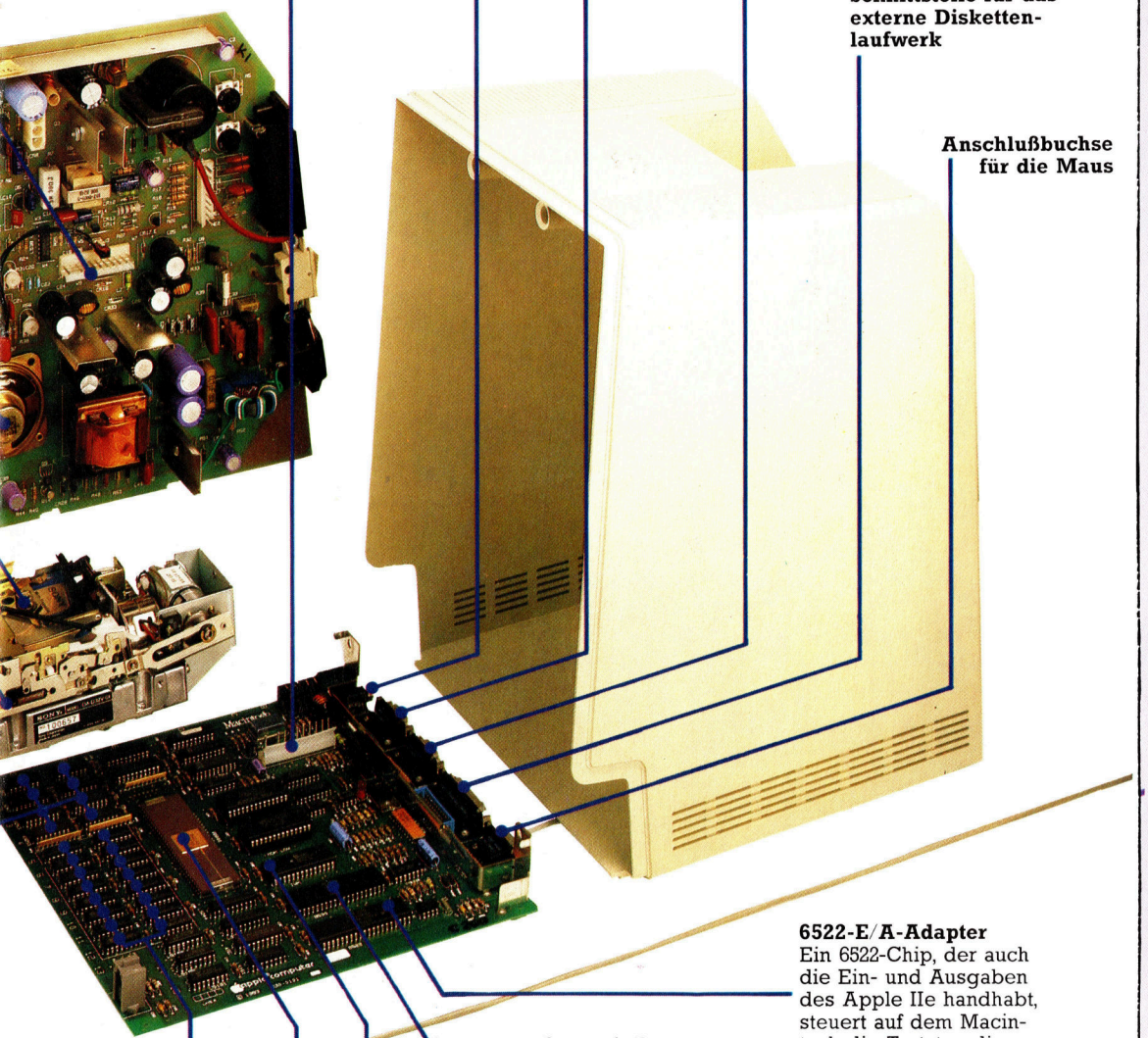
Ausgang für akustische Signale

Serieller Datenbus
Auch „virtuelle Steckleiste" genannt. Über den seriellen Datenbus kann eine ganze Reihe von Peripheriegeräten angeschlossen werden.

Druckerausgang

Schnittstelle für das externe Diskettenlaufwerk

Anschlußbuchse für die Maus



6522-E/A-Adapter
Ein 6522-Chip, der auch die Ein- und Ausgaben des Apple IIe handhabt, steuert auf dem Macintosh die Tastatur, die Maus und die Echtzeituhr.

Steuerung für serielle Datenübertragung

Diskettensteuerung
Dieser Chip steuert das eingebaute Sony-Laufwerk und (falls vorhanden) die externe Diskettenstation.

Zentraleinheit
Motorola 68000. Dieser Chip kann intern 32 Bits gleichzeitig verarbeiten, sendet und empfängt aber im 16-Bit-Format.

128-K-Arbeitsspeicher
Diese 16 Chips können gegen 256-K-RAM-Chips ausgetauscht werden. Damit wird der Macintosh auf insgesamt 512-K RAM aufgerüstet.

Apple Macintosh

PREIS
ca. 8500 Mark

ABMESSUNGEN
343 × 254 × 254 mm

ZENTRALEINHEIT
Motorola 68000

TAKTFREQUENZ
7,83 MHz

SPEICHERKAPAZITÄT
128 K RAM, 64 K ROM

BILDSCHIRMDARSTELLUNG
Eingebauter Monochrom-Monitor, 512 × 342 Bildpunkte, Fenstertechnik, Untermenüs, Symbole

SCHNITTSTELLEN
Maus, Drucker, externes Diskettenlaufwerk, HiFi-Signalverstärker, serieller Bus

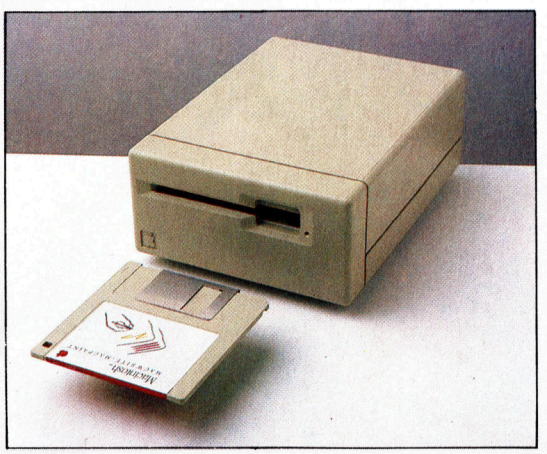
PROGRAMMIERSPRACHEN
BASIC, COBOL, PASCAL

TASTATUR
Schreibmaschinentastatur mit 59 Tasten, numerische Tastatur zusätzlich anschließbar.

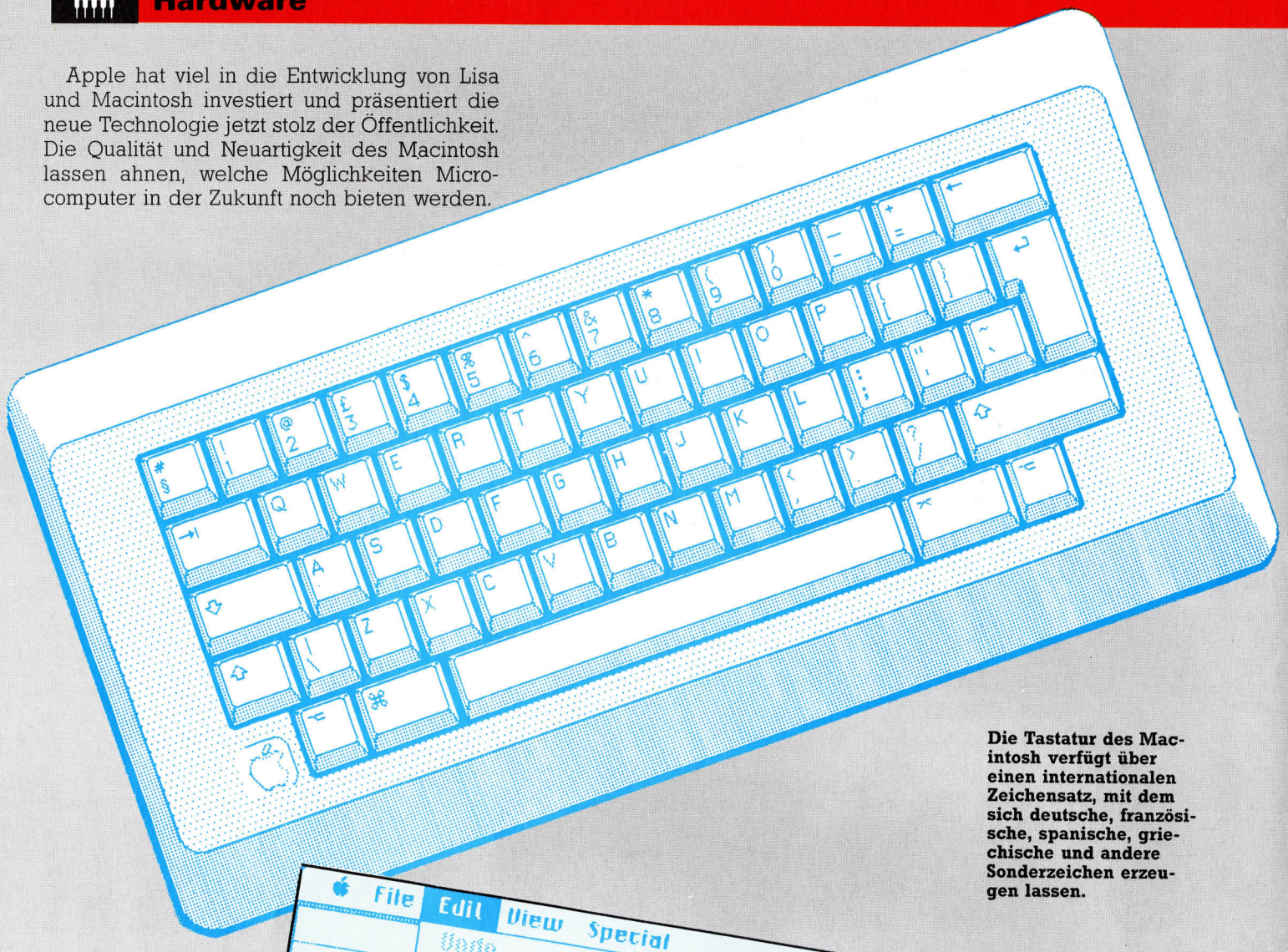
DOKUMENTATION
Mitgeliefert wird ein Betriebshandbuch mit Audiocassette und Demonstrationsdiskette. Für die Programme MacPaint und MacWrite gibt es ebenfalls Handbücher mit Audiocassetten.

STÄRKEN
Der Mac ist eine leistungsstarke Maschine, die aufgrund sorgfältig konzipierter Software leicht zu handhaben ist. Mit der Maus ist die Bedienung sehr einfach.

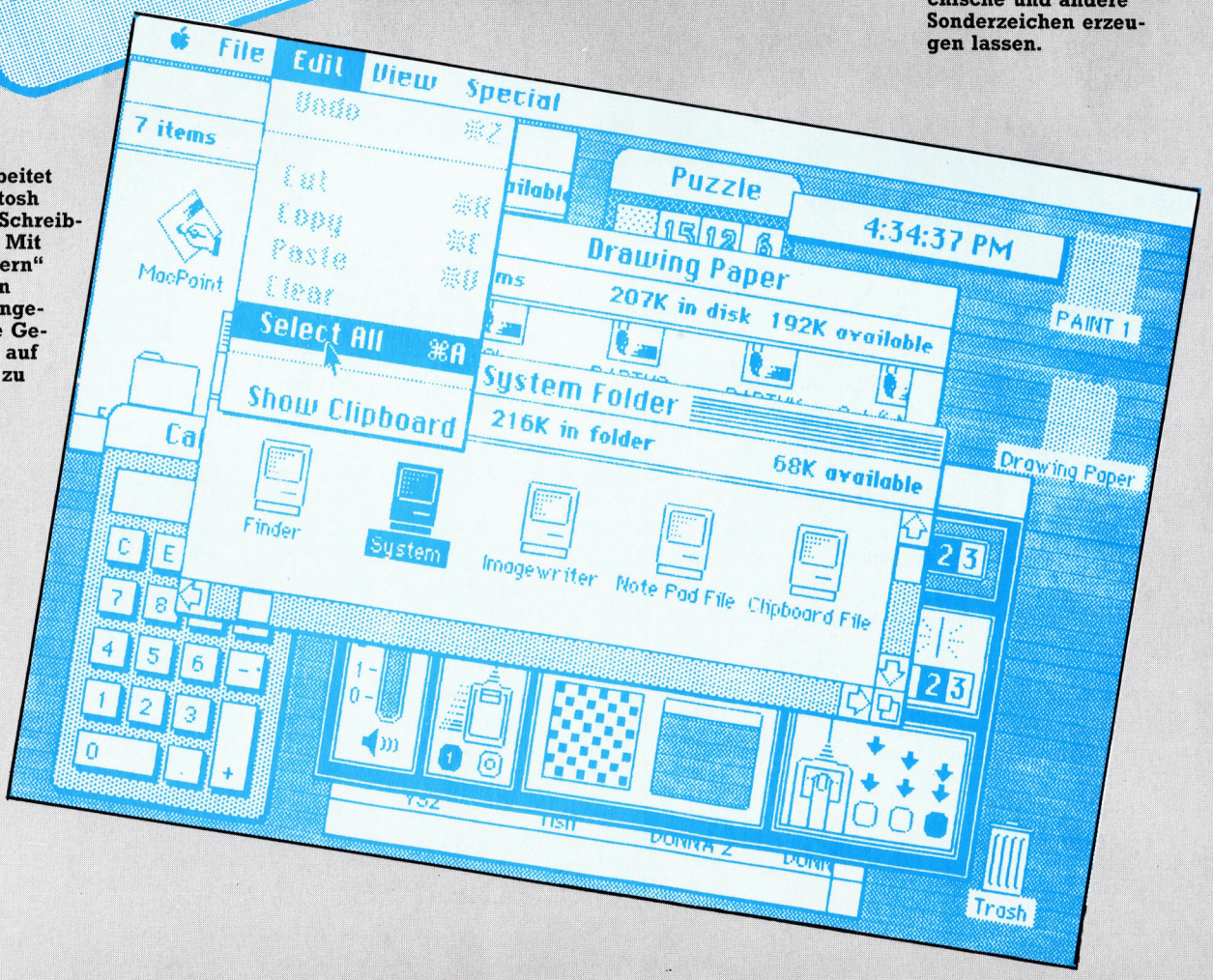
Externe Diskettenstation
Die externe Diskettenstation wurde von Sony als Auftragsarbeit für Apple gebaut. Dieses Zusatzgerät ist wichtig, da viele Funktionen bei Verwendung von nur einem Laufwerk durch den Diskettenwechsel erheblich verlangsamt werden. Das Gerät kostet ca. 1700 Mark und benötigt kein zusätzliches Interface.



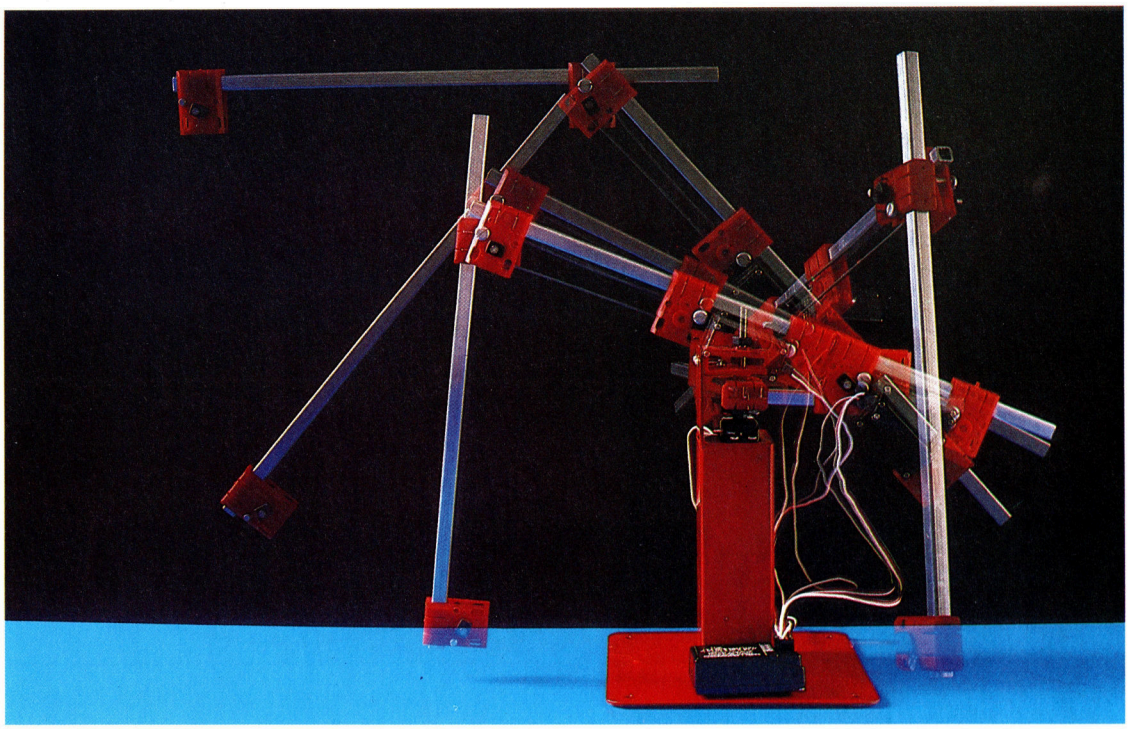
Apple hat viel in die Entwicklung von Lisa und Macintosh investiert und präsentiert die neue Technologie jetzt stolz der Öffentlichkeit. Die Qualität und Neuartigkeit des Macintosh lassen ahnen, welche Möglichkeiten Microcomputer in der Zukunft noch bieten werden.



Die Tastatur des Macintosh verfügt über einen internationalen Zeichensatz, mit dem sich deutsche, französische, spanische, griechische und andere Sonderzeichen erzeugen lassen.



Wie der Lisa arbeitet auch der Macintosh mit vertrauten „Schreibtischsymbolen". Mit Hilfe von „Fenstern" werden Teile von Schriftstücken angezeigt, die für die Gesamtdarstellung auf dem Bildschirm zu groß sind. Die Fenster lassen sich mehrfach überlagern.



Im nächsten Jahrzehnt wird der Robotergrundtyp ein einfacher Arm sein, der mit einer Vielzahl von „Händen" für den Einsatz in verschiedenen Bereichen wie Industrie, Haushalt und Freizeit ausgestattet ist. Nur in wenigen Anwendungsbereichen ist die selbständige, unabhängige, denkende Maschine erforderlich, wie wir sie aus der Science Fiction kennen. Doch sicherlich wird ein programmierbarer, halbintelligenter Greifer in der Zukunft ebensolche Bedeutung erlangen wie heute schon ein Pflug oder ein Teleskop.

Bewegtes Metall

Da die verschiedenen Möglichkeiten der Steuerung von Robotern bereits dargelegt wurden, soll nun ein anderer, sehr wichtiger Aspekt erläutert werden: Kontrolle und Bewegung von „Arm" und „Hand" eines Roboters.

Die Nutzbarkeit eines Roboters hängt wesentlich davon ab, mit welcher Genauigkeit er Objekte handhaben kann. Viele Roboter werden lediglich dazu verwendet, einfache Operationen auszuführen, so z. B. Gegenstände vom Fließband aufzunehmen und an anderer Stelle wieder abzusetzen. Entscheidend ist dabei die Gestaltung des Roboterarms.

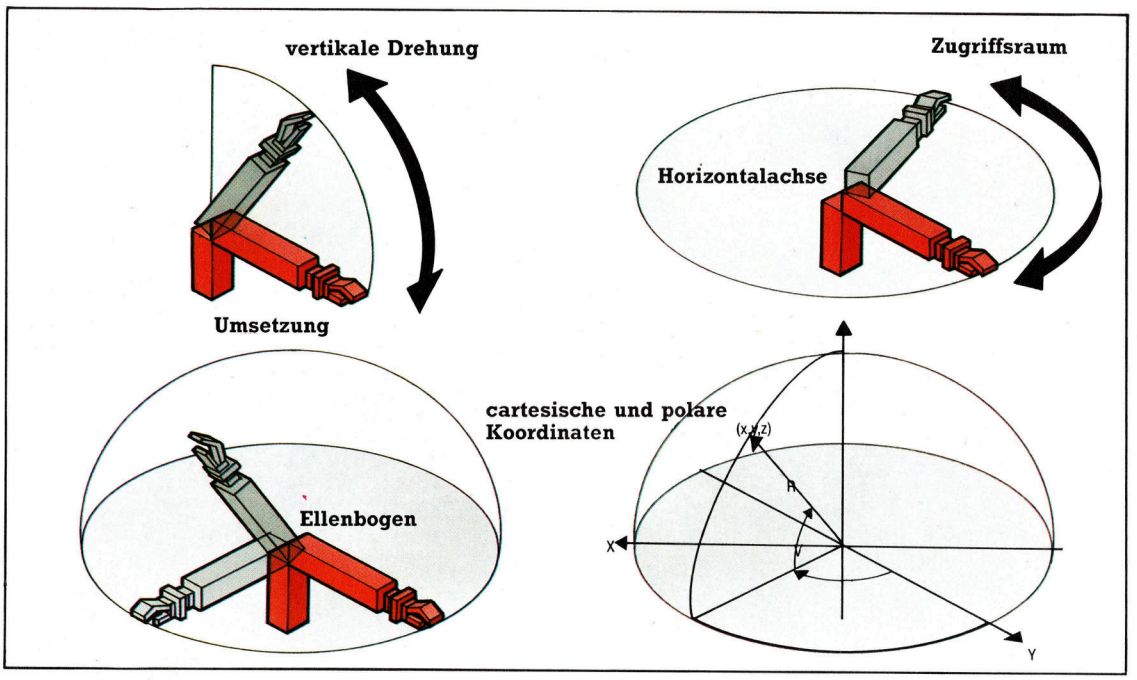
Grundsätzlich sind drei Forderungen bei der Konstruktion zu erfüllen. Zunächst muß ein System vorhanden sein, das die Position des Arms jederzeit exakt bestimmen kann. Außerdem sollte der Arm mit einem „Skelett" ausgestattet sein. Und schließlich ist eine „Muskulatur" erforderlich, die den Arm in die entsprechende Position bringt. Wenngleich diese wichtigen Elemente in ihrem Zusammenwirken unterschiedlichen Bedingungen unterliegen – je nach Aufgabe des betreffenden Roboters –, gibt es doch Grundregeln für die Konstruktion von mechanischen Armen. Eine Klassifizierung der verschiedenen Roboterarme orientiert sich an den Methoden, die zur Positionsbestimmung eingesetzt werden.

Bei der Betrachtung der Roboterbewegung wurde bereits das cartesische Koordinatensystem erläutert. Bei Anwendung dieser Methode kann die jeweilige Position eines Roboters durch zwei rechtwinklig einander zugeordnete Achsen mit den Bezeichnungen X und Y genau definiert werden. Dasselbe Prinzip ist auf einen Roboterarm übertragbar. Allerdings bewegt sich der Arm frei in den bekannten drei Dimensionen. Daher ist eine weitere Variable (Z) erforderlich, um die räumliche Position des Arms zu beschreiben.

Es ist unproblematisch, einen Roboterarm zu konstruieren, der sich exakt längs dieser drei Koordinaten bewegt. Das Ergebnis wäre ein frei steuerbarer „Kran", dessen Ausleger auf und ab, seitlich sowie vor- und rückwärts bewegt werden kann. Solche Arme eignen sich für den Einsatz an einem festen Arbeitsplatz mit begrenztem Greifraum. Der Roboter könnte sich z. B. an einer Werkbank befinden, an der sämtliche Aufgaben ausgeführt werden müßten. Ein Nachteil: Zum Betrieb wäre ein fest installierter Unterbau erforderlich. Dadurch wird der Einsatz an einem anderen Ort erschwert.

Die Beschreibung der Position eines Armes ist allerdings auch durch die Verwendung zy-

Roboter-Drehung
Die einfachste Armkonstruktion, bestehend aus einem Greifer und einem zweischenkligen Ellenbogengelenk, kann Objekte präzise plazieren. Der „Unterarm" befindet sich an einem Gelenk, das halbkreisförmige Vertikalbewegungen erlaubt und sich zugleich dreht. Damit ist Horizontalbewegung möglich. Durch gleichzeitige Drehung vertikal wie horizontal ist jeder Punkt der Umgebung erreichbar. Das läßt sich trigonometrisch aus den cartesischen Koordinaten (X, Y und Z) ableiten. H, die Horizontaldrehung, ist gleich ARCTAN (X, Y), wogegen die Vertikaldrehung (V) gleich ARCSIN (Z/R) ist.



lindrischer Koordinaten möglich. Um das zu verstehen, müssen Sie sich eine leere Konservendose vorstellen. Jede Position, jeder Punkt im Innern dieser Dose kann wie folgt definiert werden: Die Variable r gibt die Entfernung von der Mittelachse der Dose an, die Winkelvariable Θ nennt den Abstand zu einem bestimmten Fixpunkt innerhalb der Dose, und z gibt an, wie weit die Position von der Wand der Dose entfernt ist. Unter Verwendung dieser zylindrischen Koordinaten läßt sich nun leicht ein System entwickeln, mit dem jedes Objekt, das sich in einer definierbaren Position im Innern der Dose befindet, ergreifen ließe.

Arme, die durch ein sphärisches Koordinatensystem gesteuert werden, sind in ihrer Position durch zwei Winkel und einen Abstand bestimmbar. „Abstand" bedeutet in diesem Fall: Länge des Arms. Und die beiden Winkel sind die Größen der Basis-Rotation bzw. der Höhenwinkel (der Abstand des Armes vom Boden). Diese Art von Armen ist einem Geschützturm vergleichbar, in dem die Länge des Geschützrohres veränderbar ist. Sphärische Koordinaten werden gewöhnlich mit r, Θ und Φ beschrieben.

Die am häufigsten verwendete Methode zur Darstellung einer Position ist die der „zurückgerollten" Koordinaten. Dieses System wurde speziell entwickelt, um Roboterarme steuern zu können, indem die Bewegungen der menschlichen Arme simuliert werden. Wie bei den anderen Beispielen sind auch hier drei Variablen zur Bestimmung der Armposition erforderlich. Nur handelt es sich jetzt um drei Winkel, die Θ-, Φ- und γ-Koordinaten genannt werden. Θ (Theta) steht für den Winkel, um den sich die Basis dreht, Θ (Phi) für den Höhenwinkel des Arms und γ (gamma) beschreibt den Winkel zum anderen (zweiten) Gelenk.

Durch die Wahl des Koordinatensystems wird gleichzeitig der Skelettbau des Roboterarms festgelegt. Hat man das bestimmt, sind „Muskeln" erforderlich, um dem Arm Kraft zu geben. Die Roboter-„Muskulatur" läßt sich in drei Gruppen unterscheiden: elektrische, hydraulische und pneumatische.

Die „Muskulatur"

Im Zusammenhang mit der Bewegung von Robotern wurde die Verwendung elektrischer Kraft bereits behandelt. Auch zum Betrieb eines Roboterarms lassen sich diese Schrittmotoren verwenden. Sie sind an jedem Glied bzw. Gelenk des Roboterarms einsetzbar und bewegen das entsprechende Element in kleinen Schritten direkt oder indirekt.

Effektiver aber wäre die Roboter-„Muskulatur", ließe man sie so funktionieren wie die menschliche. Sie müßte sich also dehnen und zusammenziehen können, um so den Arm zu bewegen. Durch Verwendung von Kolben an den einzelnen Armteilen wäre das möglich. Diese Kolben können hydraulisch (also mit Hilfe von Flüssigkeit) oder pneumatisch (mittels Luft) bewegt werden. Bei den komplexeren Industrierobotern wird Hydraulik bevorzugt, da so ein höherer Druck erzeugt werden kann (womit der Arm stärker ist). Da Flüssigkeiten sich nicht in dem Ausmaß wie Luft ausdehnen und zusammenziehen, werden Kräfte ohne Verluste übertragen.

Ein durch hydraulische Kraft bewegter Kolben stoppt genau am vorgegebenen Punkt. Eine solche exakte Positionierung ist mit Luft oder Gas nicht möglich. Doch unabhängig davon, welches Verfahren eingesetzt wird: Einzelne oder doppelt angebrachte Kolben kön-

nen die gewünschte Bewegung erzeugen. Diese Kraftübertragung bezeichnet man als Linear-Aktivator.

Doch auch dieses Verfahren ist verbesserungsfähig bzw. verfeinerbar. Statt sich vor- und rückwärts bewegende Kolben zu verwenden und diese Bewegung in eine Gelenk-Drehbewegung zu übertragen, ließe sich auch ein Rotations-Aktivator einsetzen. Er erzeugt durch Druck auf ein internes Schraubenblatt direkt eine Drehbewegung. Das Prinzip ist dem des Schrittmotors vergleichbar, doch hydraulische Druckerzeugung übt eine wesentlich größere Kraft aus. Pneumatischer Druck wäre für dieses Verfahren ungeeignet.

Die Roboterhand

Hat man sich für die Mechanik des Roboterarms entschieden, bleibt nur die Konstruktion der „Hand" – des „end effectors", wie es in der Robotik-Sprache heißt. Vorlage ist wiederum die menschliche Hand. Stellen Sie sich einmal vor, das Handgelenk wäre eingegipst: Wie schwierig wäre dann doch die Ausführung von Bewegungen aller Art. Bedient man z. B. eine Tastatur, ermöglicht erst das Handgelenk die Auf- und Abwärtsbewegung der Finger. Diesen Vorgang bezeichnet man als „Pitch" (oder Anschlag). Ohne diesen müßte man den ganzen Unterarm bewegen, um eine Taste betätigen zu können.

Beim Anschlagen der verschiedenen Tasten bewegt sich das Handgelenk überdies seitlich. Dieses „Schwenken", das sogenannte „yaw", erlaubt den Verzicht auf eine Bewegung des ganzen Unterarms. Ist das Schreiben beendet, kann ein Mensch die Handgelenke so drehen, daß die Daumen nach oben zeigen und eine Ruhestellung der Hände links und rechts neben der Tastatur möglich ist. Hätte man kein Handgelenk, wäre eine komplizierte Schulterbewegung erforderlich, um diese Position einnehmen zu können.

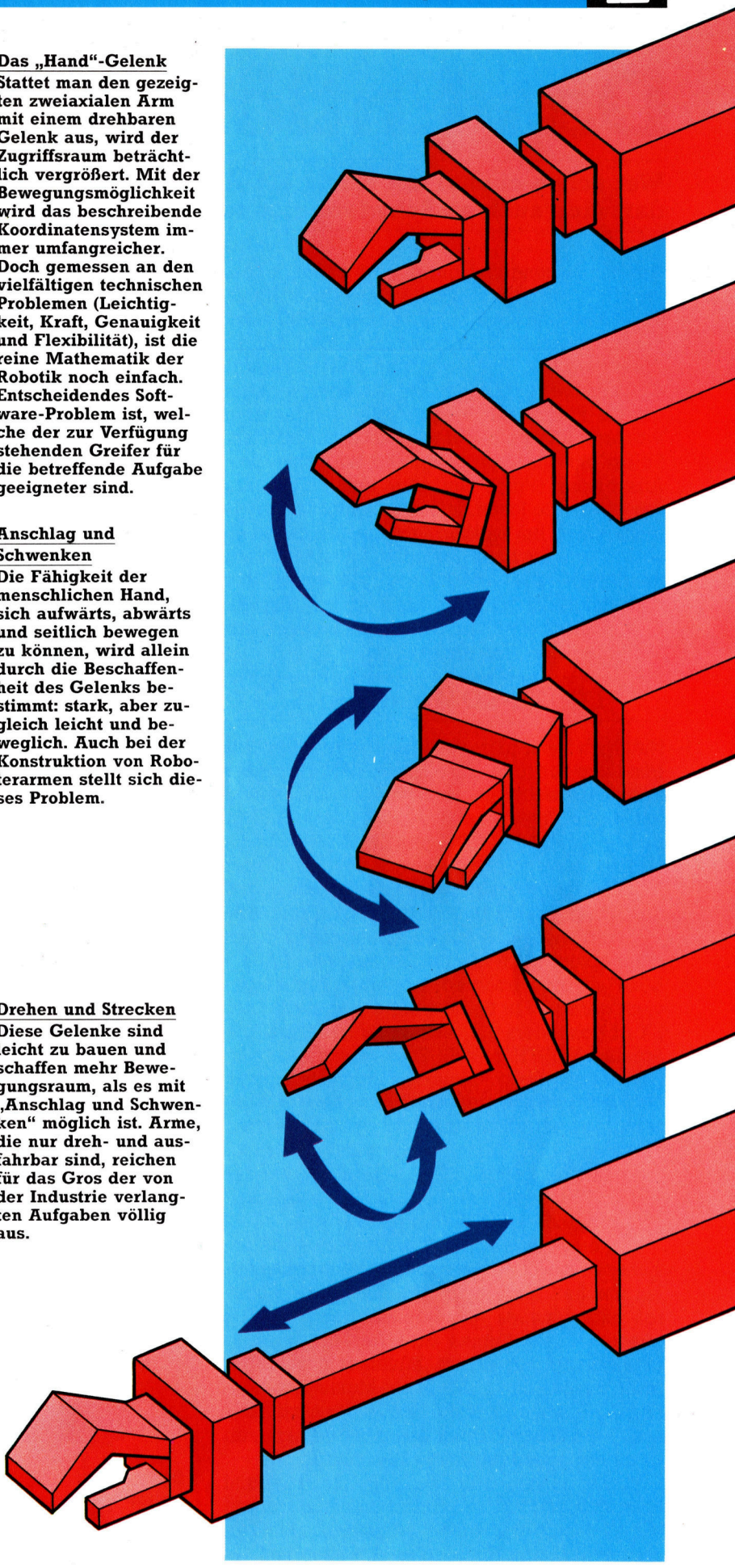
Wie wird nun die Hand selbst konstruiert? Im Idealfall müßte man ans Ende des menschenähnlichen Arms eine menschenähnliche Hand setzen. Und tatsächlich gibt es Roboter, die über eine solche verfügen. Am verbreitetsten ist die dreifingrige Roboterhand, die aus zwei „Fingern" und einem gegenüber angebrachten „Daumen" besteht.

In bestimmten Fällen ist die Ausstattung eines Roboterarms mit einer Hand sogar überflüssig. Der Begriff „end effector", mit dem die „Roboterhand" in der Fachsprache bezeichnet wird, bedeutet, daß „irgend etwas" am Ende des Arms passiert. Ein Schweißroboter beispielsweise muß nicht mit einer Hand ausgestattet sein. Statt dessen findet sich am Ende seines Arms ein Schweißkolben. Es gibt aber auch Roboter, die in der Lage sind, den jeweils für ihre Aufgabe notwendigen „end effector" selbst zu wählen.

Das „Hand"-Gelenk
Stattet man den gezeigten zweiaxialen Arm mit einem drehbaren Gelenk aus, wird der Zugriffsraum beträchtlich vergrößert. Mit der Bewegungsmöglichkeit wird das beschreibende Koordinatensystem immer umfangreicher. Doch gemessen an den vielfältigen technischen Problemen (Leichtigkeit, Kraft, Genauigkeit und Flexibilität), ist die reine Mathematik der Robotik noch einfach. Entscheidendes Software-Problem ist, welche der zur Verfügung stehenden Greifer für die betreffende Aufgabe geeigneter sind.

Anschlag und Schwenken
Die Fähigkeit der menschlichen Hand, sich aufwärts, abwärts und seitlich bewegen zu können, wird allein durch die Beschaffenheit des Gelenks bestimmt: stark, aber zugleich leicht und beweglich. Auch bei der Konstruktion von Roboterarmen stellt sich dieses Problem.

Drehen und Strecken
Diese Gelenke sind leicht zu bauen und schaffen mehr Bewegungsraum, als es mit „Anschlag und Schwenken" möglich ist. Arme, die nur dreh- und ausfahrbar sind, reichen für das Gros der von der Industrie verlangten Aufgaben völlig aus.



Das Entlöten

Wer schon einmal mit Erfolg ein Anschlußkabel selbst gemacht hat, darf sich auch an Schwierigeres wagen.

In der folgenden Übung wird beschrieben, wie Sie mit etwas Geschick aus Ihrem ZX 81 eine Maschine machen, die neben BASIC auch FORTH versteht. Wie man richtig lötet, wurde bereits erklärt. Wie aber eine solche Verbindung sauber wieder lösen? Solange es dabei nur um Kabel oder Stecker geht, ist das kein Problem. Anders bei elektronischen Bauteilen: Selbst ein einfacher Transistor hat schon drei Anschlüsse (Pins). Um aber einen Chip von der Platine zu lösen, müssen sämtliche Pins – das können bei einigen Prozessoren bis zu 40 sein – vom Lot befreit werden. Es ist praktisch unmöglich, alle Anschlüsse gleichzeitig bis zum Schmelzpunkt des Lötzinns zu erhitzen. Statt dessen wird das Lot nach und nach von den einzelnen Pins entfernt.

FORTH-ROM statt BASIC

Natürlich kann man zur Übung einen beliebigen Chip aus- oder zusätzliche RAM-Chips zur Speichererweiterung einlöten. Unser Projekt hat jedoch gleichzeitig einen echten Nutzen für Sie: Das BASIC-ROM des ZX 81 soll entfernt und durch einen Sockel ersetzt werden. Dieser ist durch ein Flachkabel mit einem Veroboard außerhalb des Rechners verbunden. Auf dieses wird wiederum ein Sockel montiert. In ihn kann später nicht nur das ausgebaute BASIC-ROM sondern auch ein anderer Chip eingesetzt werden. Für diese Übung eignet sich der ZX 81 besonders gut, weil es für das Gerät die Sprache FORTH in ROM-Form gibt, das zusätzlich auch noch Multi-Tasking ermöglicht: Dadurch können mehrere Programme gleichzeitig, jedoch völlig unabhängig voneinander, ablaufen. Für diese respektable Leistung braucht der kleine ZX 81 allerdings mindestens 2 KByte RAM, muß also mit einem zusätzlichen Speicher-Modul aufgerüstet sein.

Die hier vorgeschlagene Aufgabe wird Sie sicherer im Umgang mit elektronischen Bauteilen machen – sie erfordert aber auch sauberes und genaues Arbeiten. Im nächsten Teil unseres Bastel-Kurses wird gezeigt, wie die Erweiterung mit einem Multimeter durchgeprüft werden kann. Mit dem Multimeter können Sie Spannung, Strom und Widerstand messen. Beim Test von Bauteilen und Stromkreisen ist es ein unentbehrliches Hilfsmittel.

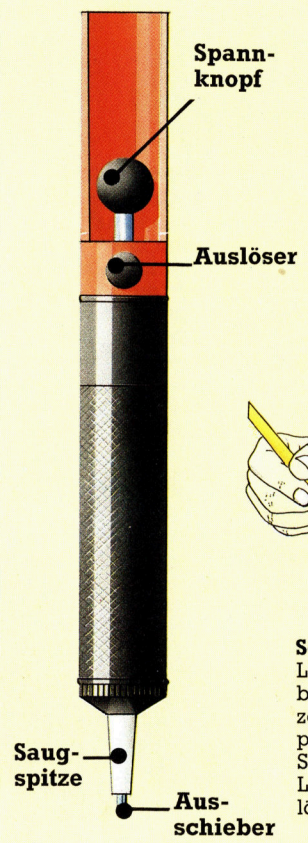
Zuletzt sollten alle Lötstellen noch einmal überprüft werden. Beim nächsten Mal zeigen wir, wie man mit einfachen Mitteln zum Ziel kommt, und den Umgang mit Meßgeräten.

Was Sie brauchen . . .

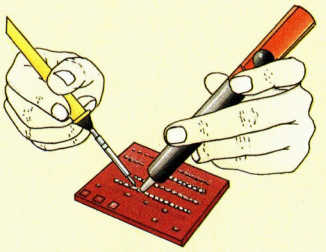
Wenn Chips in Ihrem Computer ausgetauscht, ersetzt oder durch zusätzliche Bauteile ergänzt werden sollen, empfiehlt sich der Einbau eines Chip-Sockels. Dadurch können integrierte Bauteile sehr schnell ausgewechselt werden. Neben den bereits beschriebenen Werkzeugen brauchen Sie außerdem eine Rolle Entlöt-Litze oder eine Lötsaugpumpe. In unserem Beispiel wird das BASIC-ROM des ZX 81 durch einen Standard-Sockel ersetzt, von dem aus ein Flachbandkabel zu einem Chip-Sockel außerhalb des Gehäuses führt. So kann der ZX 81 – je nach eingesetztem ROM – als BASIC- oder FORTH-Rechner verwendet werden. Außer dem Entlötgerät benötigen Sie noch eine kleine Lochplatine (Veroboard) und 28poliges Flachband-Kabel – alles Teile, die Sie in jedem Elektronik-Bastelgeschäft ohne Schwierigkeiten finden werden. Bei dem FORTH-ROM handelt es sich übrigens um eine Entwicklung von David Husband. Fragen Sie Ihren Sinclair-Händler danach.

Zuerst muß das Gehäuse des ZX 81 geöffnet und das BASIC-ROM gesucht werden. Von den insgesamt fünf Gehäuseschrauben sitzen drei unter den aufgeklebten Gummifüßchen am Boden des Rechners. Unter dem Gummifuß neben den EAR- und MIC-Anschlüssen befindet sich keine Schraube. Lösen Sie die Gummis vorsichtig, und drehen Sie alle fünf Kreuzschlitz-Schrauben los. Nun können Sie die Bodenplatte abnehmen, und die Unterseite der Platine kommt zum Vorschein. Nach Lösen der drei sichtbaren Kreuzschlitz-Schrauben kann die Platine aus dem Gehäuse herausgenommen werden. Zwei der Schrauben sitzen nahe beim Vielpol-Stecker, die dritte befindet sich neben dem Kühlkörper. Wenn Sie die Platine jetzt herumdrehen, sehen Sie das BASIC-ROM etwas links über dem Tastaturanschluß.

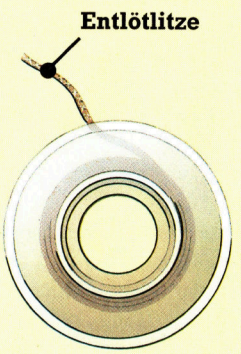
Werkzeug zum Entlöten

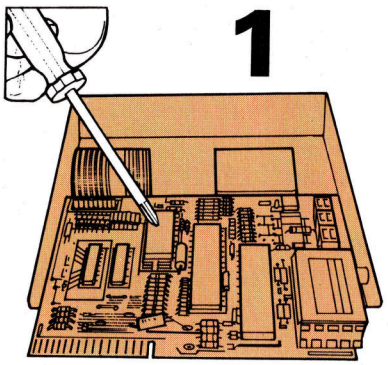


Beim Entlöten sind zwei verschiedene Techniken gebräuchlich: Wer nur selten Bauteile auswechselt, arbeitet mit der preiswerten Entlötlitze. Sie besteht aus feinsten Kupferdrähtchen, die mit Flußmittel überzogen sind. Durch den Kapillar-Effekt saugt die Litze flüssiges Lötzinn auf wie ein Lampendocht. Entlötlitze können Sie in unterschiedlichen Breiten und Längen kaufen. Das richtige Maß hängt von der Menge des Lötzinns ab, das entfernt werden soll. Ein Meter der nicht wiederverwendbaren Litze kostet ca. 4 Mark. Alternativ können Sie aber auch mit einer Lötsaug-Pumpe arbeiten, was auf die Dauer billiger ist. Das Gerät arbeitet nach dem umgekehrten Prinzip einer Fahrrad-Luftpumpe.

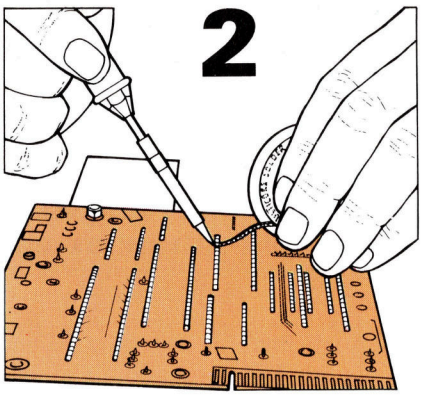


So wird entlötet: Lötpunkt mit dem Kolben bis zur Schmelze erhitzen, Feder der Lötsaugpumpe spannen und die Spitze des Gerätes am Lötpunkt ansetzen. Auslöseknopf drücken.



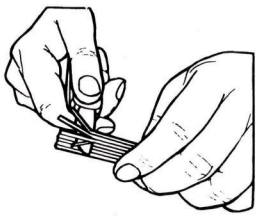


1



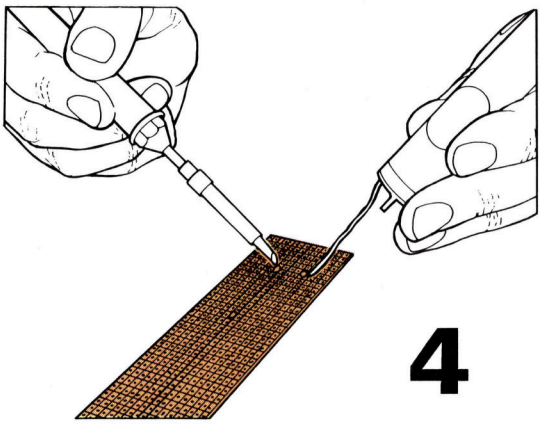
2

Wenn die Temperatur hoch genug ist, gibt es beim Entlöten keine Probleme. Einfach die Entlötlitze mit der Spitze des Kolbens auf die Lötstelle pressen, bis das Flußmittel der Litze schmilzt – dabei entsteht ein kleines Rauchwölkchen. Danach sollte die Litze das Lötzinn aufsaugen. Das vollgesogene Litzenende muß jeweils abgeschnitten werden.



3

Flachkabel gibt es in unterschiedlicher Breite. Bei diesem Beispiel wird ein 28poliges gebraucht. Sie können natürlich auch zwei Kabel aus jeweils 14 isolierten Drähten verwenden. Alle 28 Enden werden 10 mm lang abisoliert und verzinnt. Auf der Chipseite mit dem halbkreisförmigen Ausschnitt anfangen und die Enden nacheinander sorgfältig festlöten. Durch den Ausschnitt am Chip können Sie erkennen, wie das Bauteil richtig eingesetzt ist. Gelegentlich findet man auch aufgedruckte Punkte als Hilfe.

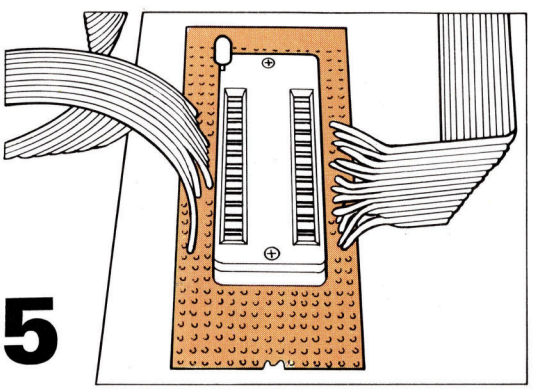


4

Jetzt können Sie das Flachkabel auf die gewünschte Länge schneiden und auf der freien Seite ebenfalls verzinnen. Beim Anlöten der Enden auf der Nebenplatine muß jede Leitung genau an den der Hauptplatine entsprechenden Anschlußpunkt führen. Vorsicht, hier darf nichts verwechselt werden!

Lochplatinen (Veroboard)

Vorgelochte Platinen mit kupfernen Leiterbahnen sind für den Selbstbau elektronischer Schaltungen gedacht. Die Leiterbahnen lassen sich je nach Bedarf durchschneiden oder -kratzen. Beim Einkauf sollten Sie darauf achten, daß das Rastermaß der Platine zum Rastermaß des geplanten Sockels bzw. ICs paßt.

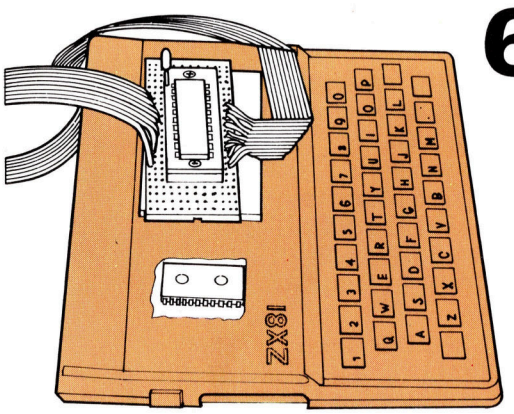


5

Den Sockel lötet man am besten auf ein Stück Lochplatine (Veroboard), das man in geeigneter Größe abbricht oder absägt. Der Sockel wird so eingesteckt, daß jeder Lötstift auf einer eigenen Leiterbahn festgelötet werden kann. Zuerst zwei Eckpunkte festlöten, dann sitzt der Sockel beim Arbeiten fest.
Auf „kalte" Lötstellen achten!

Achtung!

Während der Garantiefrist darf Ihr Heimcomputer nur vom Hersteller bzw. der Vertragswerkstatt geöffnet werden. Meist wird nach selbständigen Eingriffen in das Gerät jede weitere Garantieleistung verweigert.



6

Wenn alles fertig gelötet ist, sollten Sie jede einzelne Verbindung noch einmal genau überprüfen. Das gilt für die Platine des Rechners genauso wie für die neue Tochterplatine. Ist Lötzinn zwischen zwei Anschlußpunkte gelaufen? Gibt es abstehende Drähtchen, die einen Kurzschluß erzeugen? In Zweifelsfällen mit der Lupe nachprüfen. Sicherheitshalber können Sie auch mit einer Nadel zwischen zwei Anschlußpunkten hindurchkratzen.

Abenteuerspiele

Die zahlreichen Möglichkeiten der LOGO-Listenverarbeitung eignen sich ideal für Spielanwendungen. In diesem Teil des Kurses soll der Entwurf für ein Text-Adventure entwickelt werden.

Zuerst werden wir uns mit den generellen Aspekten, die zur Programmierung eines Adventures nötig sind, befassen. Die Einzelheiten und Spiel-Details werden später besprochen.

Adventures (Abenteuerspiele) basieren auf fünf grundlegenden Handlungsmustern: Der Spieler muß Gegenstände aufnehmen und ablegen und mitgeführte Dinge auflisten können. Ferner muß er in der Lage sein, die Umgebung zu betrachten und sich von einem Ort zum nächsten bewegen. Diese Aktivitäten bilden auch in dem LOGO-Adventure die Grundlage. Der Einfachheit halber werden die Anweisungen auf zwei Arten limitiert: einzelne Wörter (wie zum Beispiel SEHEN) und die Kombination Verb-Substantiv (wie etwa NEHMEN RING). Das Programm arbeitet ferner mit zwei Listen. Die erste heißt INVENTAR und enthält die Gegenstände, die der Spieler bereits gesammelt hat. Die zweite, genannt OBJEKTE, zeigt die Objekte, die in dem jeweiligen Raum versteckt sind.

Zunächst die INVENTAR-Liste:

```
TO INV
  PRINT [INVENTAR:]
  IF EMPTY? :INVENTAR THEN PRINT
    [NICHTS]
  ELSE PRINT :INVENTAR
END
```

Wie Sie sehen, wird in dieser Prozedur die komplette IF-Abfrage eingesetzt: IF ‹Bedingung› THEN ‹Aufruf 1› ELSE ‹Aufruf 2›. Der Befehl zum Aufnehmen von Gegenständen lautet NEHMEN.

```
TO NEHMEN :ITEM
  IF MEMBER? :ITEM :OBJEKTE
    THEN AUFNEHMEN :ITEM ELSE PRINT
    [UNMOEGLICH ES IST NICHT DA]
END
```

MEMBER? überprüft, ob der Gegenstand in der Liste enthalten ist. Um etwas „nehmen" zu können, sind zwei Dinge erforderlich: Es muß dem Inventar hinzugefügt und aus der Objekt-Liste entnommen werden.

```
TO AUFNEHMEN :ITEM
  RECHNE.ZU.INV :ITEM
  ENTFERNE.AUS.RAUM :ITEM
END
TO RECHNE.ZU.INV :ITEM
  MAKE "INVENTAR SENTENCE :ITEM
    :INVENTAR
END
TO ENTFERNE.AUS.RAUM :ITEM
  MAKE "OBJEKTE STREICHEN :ITEM
    :OBJEKTE
END
```

Die letzte dieser Prozeduren zeigt, wie ein Element aus einer Liste entfernt wird. Das Verfahren wurde in vorangegangenen Übungen bereits erläutert.

```
TO STREICHEN :ITEM :LIST
  IF :ITEM = FIRST :LIST THEN OUTPUT
    BUTFIRST :LIST
  OUTPUT SENTENCE FIRST :LIST
    STREICHEN :ITEM BUTFIRST :LIST
END
```

Der Befehl für das Ablegen eines Gegenstandes erfolgt ähnlich.

```
TO ABLEGEN :ITEM
  IF MEMBER? :ITEM :INVENTAR THEN
    LEGAB :ITEM ELSE PRINT [ABLEGEN
    NICHT ERFORDERLICH!]
END

TO LEGAB :ITEM
  ENTFERNE.AUS.INV :ITEM
  RECHNE.ZU.RAUM :ITEM
END

TO ENTFERNE.AUS.INV :ITEM
  MAKE "INVENTAR STREICHEN :ITEM
    :INVENTAR
END

TO RECHNE.ZU.RAUM :ITEM
  MAKE "OBJEKTE FPUT :ITEM :OBJEKTE
END
```

Nach Eingabe dieser Prozeduren ist nun eine Überprüfung ihrer Wirkungsweise erforderlich. Zuerst müssen die beiden globalen Variablen INVENTAR und OBJEKTE definiert und überprüft werden:

```
MAKE "OBJEKTE [SCHWERT SPEER
  FACKEL]
MAKE "INVENTAR [LATERNE]
NEHMEN "SCHWERT
ABLEGEN "LATERNE
```

Mit den folgenden Anweisungen sind INVENTAR und OBJEKTE zu überprüfen:

```
PRINT :OBJEKTE
PRINT :INVENTAR
```

Beachten Sie dabei bitte, daß vor den Objektnamen sowohl beim Befehl NEHMEN als auch beim ABLEGEN Anführungsstriche einzuset-

zen sind. Um NEHMEN SCHWERT verwenden zu können, muß SCHWERT wie folgt definiert werden:

```
TO SCHWERT
  OP "SCHWERT
END
```

Dies ist natürlich bei jedem benutzten Begriff zu tun.

Der Befehl SEHEN gibt eine Beschreibung des jeweiligen Raumes, eine Liste der darin befindlichen Objekte und zeigt die möglichen Ausgänge. Dazu sind zwei weitere Listen erforderlich: Raumbeschreibung und Ausgänge. Da für ausreichend lange Beschreibungen mehr als eine Bildschirmzeile nötig ist, wird die Beschreibungsliste in Form einer Liste mit mehreren Listen definiert. Zum Beispiel:

```
MAKE "RAUMBESCHREIBUNG [[DU
  STEHST AM EINGANG] [EINER HOEHLE]]
```

Um zu verdeutlichen, wie die Räume zusammenhängen, wird jeder Raum mit einer Nummer gekennzeichnet. Die Ausgangsliste ist eine Liste von Unterlisten, die jeweils aus einer Richtung und einer Raumnummer bestehen. Folglich:

```
MAKE "AUSGANG.LIST [[N 4] [0 6]]
```

Jetzt kann man SEHEN definieren:

```
TO SEHEN
  PRINTL :RAUMBESCHREIBUNG
  PRINT "
  PRINT [DU SIEHST:]
  IF EMPTY? :OBJEKTE THEN PRINT
    [NICHTS BESONDERES] ELSE PRINT
    :OBJEKTE
  PRINT "
  PRINT [DU KANNST GEHEN NACH:]
    PRINT.AUSGAENGE :AUSGANG.LIST
  PRINT "
END
```

In dieser Prozedur wurden zwei besondere Print-Routinen verwendet, um den Text leichter lesbar zu machen. PRINTL wird zur Darstellung mehrerer Textzeilen verwendet.

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

PRINT.AUSGAENGE stellt die Wege aus dem Raum dar, ohne jedoch die Raumnummern anzugeben.

```
TO PRINT.AUSGAENGE :LIST
  IF EMPTY? :LIST THEN PRINT "STOP
  MAKE "AUSGANG FIRST :LIST
  PRINT1 FIRST :AUSGANG
  PRINT1 " ' '
  PRINT.AUSGAENGE BUTFIRST :LIST
END
```

Alles, was über einen Raum bekannt ist, läßt sich durch Zusammenfassung der drei Unterlisten aussagen: die Beschreibung, die Objekte und die Ausgänge. Zum Beispiel:

```
MAKE "RAUM.1 [[[DU STEHST AM
  EINGANG] [EINER HOEHLE]] [SCHWERT]
  [[N 4] [0 6]]]
```

Angenommen, daß RAUM.1 so definiert wurde, läßt er sich mit der folgenden Prozedur in mehrere Komponenten untergliedern:

```
TO ZUWEISEN.VARIABLEN
  MAKE "RAUM THING "RAUM.1
  MAKE "RAUMBESCHREIBUNG
    RAUMBESCHREIBUNG :RAUM
  MAKE "OBJEKTE OBJEKTE :RAUM
  MAKE "AUSGANG.LIST AUSGANG.
    LIST :RAUM
END
```

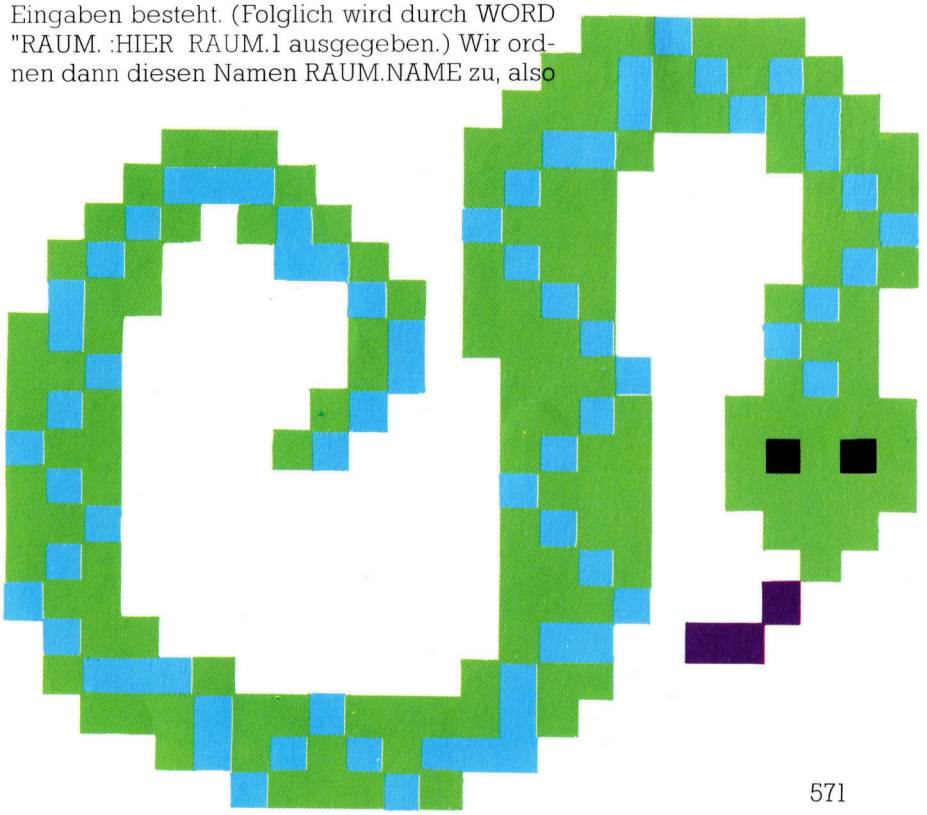
THING "RAUM.1 ist eine Alternative zu :RAUM.1 und bedeutet „Inhalt der Variablen RAUM.1". Der Grund für die Verwendung dieser Form wird später erläutert. Die Unterprozeduren sind wie folgt definiert:

```
TO RAUMBESCHREIBUNG :RAUM
  OUTPUT ITEM 1 :RAUM
END

TO OBJEKTE :RAUM
  OUTPUT ITEM 2 :RAUM
END

TO AUSGANG.LIST :RAUM
  OUTPUT ITEM 3 :RAUM
END
```

In dieser Form gilt die Prozedur nur für RAUM.1. Soll sie für jeden Raum gelten, muß sie erweitert werden. Dies geschieht mit der Verwendung der globalen Variablen HIER, die die gegenwärtige Raumnummer enthält. Angenommen, das wäre im Augenblick 2. Der LOGO-Befehl WORD liefert als Ausgabe ein Wort, das aus einer Kombination seiner beiden Eingaben besteht. (Folglich wird durch WORD "RAUM. :HIER RAUM.1 ausgegeben.) Wir ordnen dann diesen Namen RAUM.NAME zu, also



```
:RAUM.NAME ist RAUM.2.
  TO ZUWEISEN.VARIABLEN
    MAKE "RAUM.NAME WORD "RAUM.
      :HIER
    MAKE "RAUM.THING :RAUM.NAME
    MAKE "RAUMBESCHREIBUNG
      RAUMBESCHREIBUNG :RAUM
    MAKE "OBJEKTE OBJEKTE :RAUM
    MAKE "AUSGANG.LIST AUSGANG.
      LIST :RAUM
  END
```

Es ist jetzt möglich, eine Karte der Räumlichkeiten des Adventures zu zeichnen und die Beschreibungen aufzulisten, einschließlich der Objekte und Ausgänge.

LOGO-Dialekte

In einigen LOGO-Versionen gibt es die Befehle EMPTY?, ITEM, COUNT und MEMBER? nicht. Die Erläuterungen für die ersten drei Anweisungen wurden in der letzten Folge gegeben. Die Definition von MEMBER? lautet:

```
TO MEMBER? :ITEM :LIST
  IF :LIST = [] THEN OUTPUT "FALSE
  IF :ITEM = FIRST :LIST THEN OUTPUT
    "TRUE
  OUTPUT MEMBER? :ITEM BUTFIRST
    :LIST
END
```

In einigen LOGO-Versionen sind folgende Anweisungen zu ersetzen:

EMPTYP statt EMPTY?
LISTP statt LIST?
MEMBERP statt MEMBER?
TYPE statt PRINT1

Ferner gibt es den Befehl EQUALP, der überprüft, ob seine beiden Eingaben identisch sind. Man verwendet ihn statt =, um Listen und Worte zu vergleichen. Das folgende Beispiel zeigt die variierte Schreibweise des IF-Befehls:

```
IF EMPTYP :OBJEKTE [PRINT [NICHTS
  BESONDERES]] [PRINT :OBJEKTE]
```

Die erste Liste wird ausgeführt, wenn die Bedingung „wahr" ist, die zweite, wenn sie „falsch" ist.

Lösungen

1. Drucken einer Liste in umgekehrter Reihenfolge:

```
TO PRINTR :LIST
  IF EMPTY? :LIST THEN PRINT "STOP
  PRINT1 LAST :LIST
  PRINT1 " ' '
  PRINTR BUTLAST :LIST
END
```

Ausgabe einer Liste in umgekehrter Reihenfolge:

```
TO REVERSE :LIST
  IF EMPTY? :LIST THEN OUTPUT []
  OUTPUT SENTENCE LAST :LIST
    REVERSE
    BUTLAST :LIST
END
```

2. Auf den vorhergehenden Seiten wird gezeigt, wie die Prozedur lautet, mit der man ein Element aus einer Liste löscht.

Dichten mit LOGO

Im letzten Teil wurde gezeigt, wie man mit LOGO „dichten" kann. Eine Möglichkeit wäre, das „Gerüst" einer Satzstruktur, wie zum Beispiel Substantiv-Verb-Substantiv, zu schaffen und Worte aus Unterlisten dieser Satzteile zu wählen. Z. B.:

```
GEDICHT2 [SUBSTANTIV VERB
  SUBSTANTIV]
```

Das Ergebnis lautet:

```
TO GEDICHT2 :GERUEST
  IF EMPTY? :GERUEST PRINT "STOP
  GEDICHT2.1 FIRST :GERUEST
  GEDICHT2 BUTFIRST :GERUEST
END
TO GEDICHT2.1 :WRD
  IF :WRD = "SUBSTANTIV
    (PRINT1 " ' ' GETRANDOM
    :SUBSTANTIV)
  IF :WRD = "VERB (PRINT2 " ' '
    GETRANDOM :VERB)
END
```

Werden weitere Satzteile benutzt, erweist sich dieses Verfahren als sehr umständlich. Betrachtet man die Variablen genauer, läßt sich das verbessern. Zunächst geben wir ein:

```
MAKE "ROSE "SUESS
MAKE "ANDERERNAME "ROSE
```

Nun zeigt sich, daß

```
PRINT :ROSE SUESS ausgibt
PRINT :ANDERERNAME ROSE ausgibt
PRINT THING :ANDERERNAME SUESS
  ausgibt
```

THING stellt den mit einem Namen verknüpften Wert dar. Im letzten Fall ist der auf THING folgende Name der Wert der Variablen ANDERERNAME, also ROSE.

```
TO GEDICHT 2.1
  (PRINT1 " ' 'GETRANDOM THING
    :WRD)
END
```

Der Aufruf des Befehls GEDICHT2.1 "SUBSTANTIV weist den Inhalt von SUBSTANTIV der Variablen WRD zu. THING :WRD ist folglich der mit SUBSTANTIV verbundene Wert, die Liste der Substantive also.

```
GEDICHT2 [ART ADJ SUBSTANTIV
  VERB PRAEP ART ADJ SUBSTANTIV]
```

in Verbindung mit einem entsprechenden Vokabelschatz, erhält man folgendes Resultat:

```
EIN GRUEN PLANET FLIEGEN UNTER
  EIN PARANOID HIMMEL
```

Verlassene Mine

Der „Manic Miner", programmiert von Matthew Smith, ist in der Welt der Computerspiele zu einem Kultprogramm geworden. Als Grund dafür ist die Kombination von hintergründigem Humor mit exzellenter Grafik zu sehen, die das Programm zu einem echten Hit machten.

Manic Miner:
Für 48K-Spectrum, ca. 30 Mark
Für Commodore 64, ca. 35 Mark
Hersteller/ Vertrieb:
Software Projects, Fachhandel
Autor:
Matthew Smith
Joysticks:
In beiden Versionen
Programm:
Cassette

Der Hauptdarsteller, Bergmann Willy, hat alle Chancen, zu einem Dauerstar zu werden – in Fortsetzungen dieses Spiels nämlich, die gar nicht ausbleiben können. Im Prinzip ist „Manic Miner", entwickelt für den ZX Spectrum mit 48 K und den Commodore 64, ein recht einfaches Spiel, das auf einem anderen Bestseller („Kong") basiert. Die Aufgabe in diesem Programm bestand darin, über Leitern und Träger aufwärts zu klettern und dabei Hindernissen auszuweichen, um eine hilflose Schöne, die sich in der Gewalt des berüchtigten Riesenaffen befand, zu befreien.

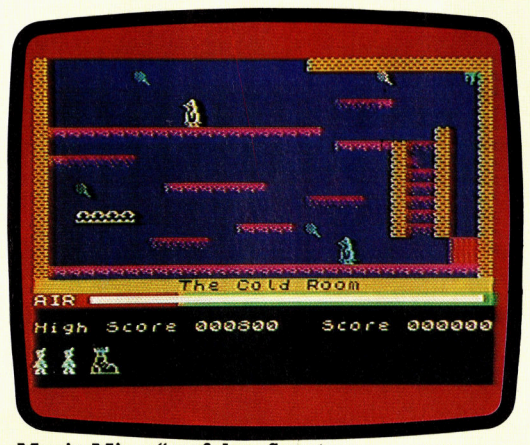
Beim „Manic Miner" übernimmt der Spieler die Rolle des Bergarbeiters (englisch: Miner, daher der Titel) Willy, der in der Bergwerksstadt Surbiton zu Hause ist. Willy bewegt sich suchend durch ein verlassenes Bergwerk, in dem eine längst untergegangene Zivilisation nach Gold und anderen wertvollen Erzen suchte. Unglücklicherweise haben die früheren Minenbesitzer vergessen, ihre Schürfroboter abzuschalten. Das hat zur Folge, daß die Schatzsuche recht kompliziert wird, da diese Roboter aggressiv sind.

Das Bergwerk besteht aus zwanzig Höhlen. In jeder muß Willy zunächst vier Schlüssel finden, bevor er die Tür öffnen kann, die zur nächsten Höhle führt. In jeder Höhle befinden sich überdies verschiedene Plattformen, auf die Willy springen muß, um an die Schlüssel heranzukommen. Einige dieser Plattformen sind ziemlich brüchig (wohl eine Alterungserscheinung) und geben nach, wenn Willy darauf tritt. Die Höhlen selbst sind einfallsreich betitelt: beispielsweise „Eugene's Lair" (eine Ehrung für den ausgezeichneten Programmierer Eugene Evans, seinerzeit bei Imagine), „Miner Willy Meets the King Beast", „Attack of the Mutant Telephones" (ein weiterer Insider-Scherz, der auf den Tick des Programmierers Jeff Minter anspielt, dessen Spezialität mutierte Lamas sind) und „Skylab Landing Bay". All diese Höhlen sind von einer Vielzahl fremdartiger Kreaturen bevölkert, deren Berührung tödlich ist. Das gilt sogar für die geheimnisvollen Pflanzen.

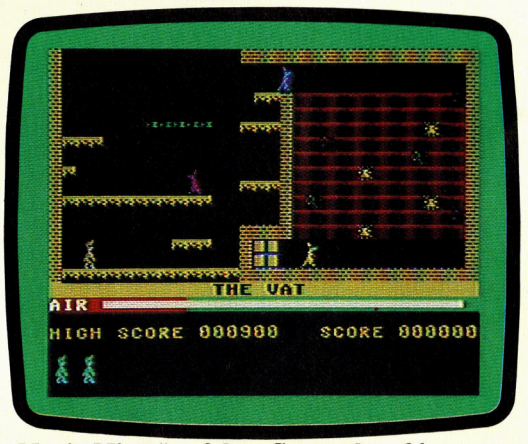
Um all diesen Problemen aus dem Wege zu gehen, wird Willy mit den drei einfachen Kommandos „Links", „Rechts" und „Springen" gesteuert. Das macht das Spiel so attraktiv: Da man die Steuerung auf Anhieb versteht, gibt es keine lange Lernperiode. Dazu kommt, daß zur Steuerung die Tasten benutzt werden können, die einem am meisten zusagen.

Willy hat drei Leben und verfügt in jeder Phase über einen begrenzten Luftvorrat, der auf dem Bildschirm angezeigt wird. Der Verlust des dritten Lebens bringt Willy automatisch in die erste Höhle zurück, was natürlich sehr enttäuschend ist. So überrascht es kaum, daß viele Spieler so lange herumprobiert haben, bis es ihnen gelang, an jeder beliebigen Stelle (= Höhle) ins Spiel einzusteigen.

Die Commodore-Adaption ist eine fast exakte Kopie der Spectrum-Version, wobei die vielseitigen Sound- und Grafikmöglichkeiten des Commodore 64 nicht voll genutzt wurden.



„Manic Miner" auf dem Spectrum



„Manic Miner" auf dem Commodore 64

Die verrückten und wundersamen Gegenstände, die in der Unterwelt des Manic Miner zu finden sind, haben das Programm zum Kultspiel gemacht. Selbst erfahrene Spieler entdecken in den Höhlen immer wieder neue, ungewöhnliche Dinge.

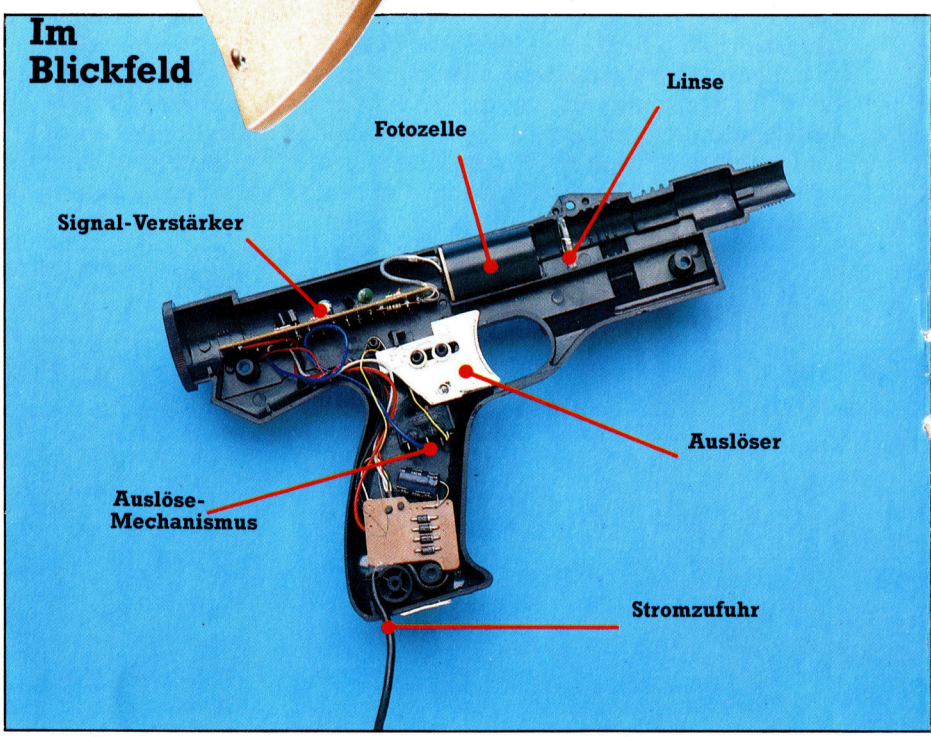
Licht-pistole

Die Stack Light Rifle (SLR) oder – frei übersetzt – Stacks Licht-Pistole wurde entwickelt, um Computer-Schießspiele noch realistischer zu gestalten. Dabei wurde der eigentliche Gewehrkörper mit einem optischen System verbunden, das auch in Kameras Verwendung findet.

Hauptbestandteil des „Stack Light Rifle System" ist eine elektronische Pistole, die durch ein Kabel mit dem Computer verbunden wird. Je nach Ausführung befindet sich am Ende des Kabels ein an den Computer passender Stecker oder eine Buchse. Bei der Version für den ZX Spectrum enthält diese Verbindung zwei Chips, sowie einfache weitere Komponenten, die als Interface zum Computer selbst dienen. Zwecks größerer Genauigkeit der Pistole wurde sie mit einem Schulterstück ausgestattet, das ganz genau auf den Pistolengriff paßt.

Das elektronische Innenleben der nach einem Gewehr aussehenden Pistole besteht aus einer Fotozelle sowie einem kleinen Verstärker und einem Buffer. Das beim „Schuß" eindringende Licht wird durch eine Plastiklinse auf die Fotozelle fokussiert, die so empfindlich ist, daß sie auch kleine Veränderungen der Lichtstärke registriert. Nach der Verstärkung löst das Signal eine Digital-Schwingung aus, die dann in den Computer eingelesen wird. Die in diesem Augenblick gezeigte Bildschirmposition wird abgetastet, und zwar genau der Punkt, auf den das Gewehr gerichtet ist. Da der Computer die Schwingung über das Lichtgewehr erhält, vergleicht er intern den ermittelten Wert mit dem Positionswert auf dem Bildschirm. Stimmen die beiden überein, erhält der Spieler einen Punkt.

Die Light Rifle wurde für den ZX Spectrum, den Commodore VC 20 sowie den C 64 produziert, konnte sich jedoch aufgrund mangelnder Software-Unterstützung nicht durchsetzen. Zwar hatten viele unabhängige Software-Häuser die Produktion von Programmen angekündigt, die mit dieser Art Peripherie beeinflußbar wären, doch nur wenige realisierten dieses Vorhaben. Eine der Ausnahmen war Micromania. Noch nachteiliger auf die Verkaufschancen wirkte sich die Tatsache aus, daß Stack den Anwendern keine Hilfestellung bot, ei-



gene Programme für die Rifle zu schreiben. Da außerdem Dokumentation und Arbeitshilfen nicht zur Verfügung gestellt wurden, die technische Details vermittelt und die Funktionsweise erläutert hätten, stellt die Light Rifle keine echte Alternative zum Joystick dar.

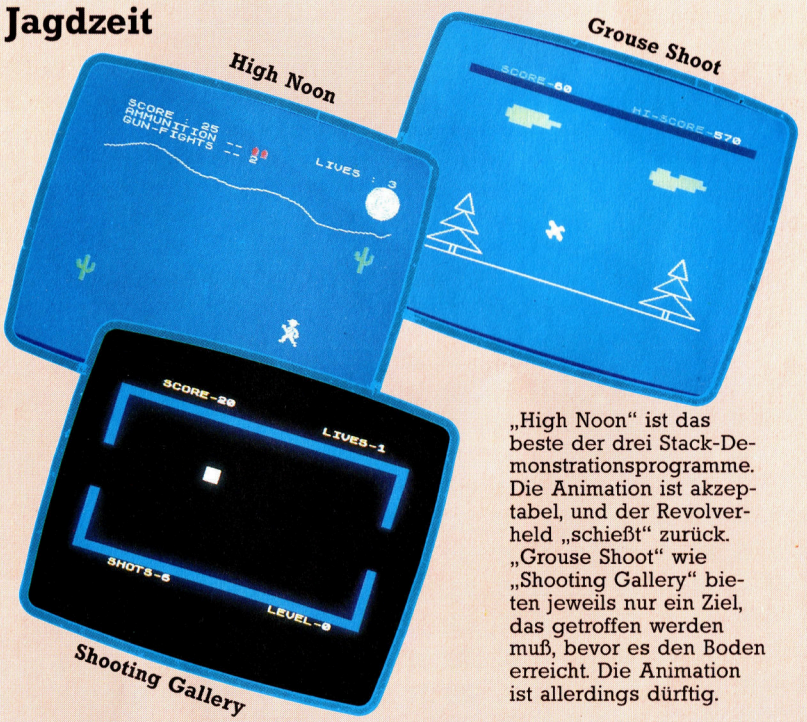
Die Lichtpistole funktioniert nach demselben Prinzip wie der Lichtgriffel, ist aber größer und so angelegt, daß sie bis auf eine Entfernung von drei Metern Reaktionen hervorruft. Eine Berührung des Bildschirms ist also nicht erforderlich. Um den Einfluß von Fremdlicht weitgehend auszuschließen, hat man die Light Rifle mit einem langen dunklen Rohr (dem Lauf) und einer Linse ausgestattet. Damit läßt sich eine gewisse, wenn auch nicht perfekte Genauigkeit erreichen, die es dem Benutzer gestattet, vom Sessel aus zu „schießen". Die vorhandenen Spiele sind ein dürftiges Beispiel für die vielfältigen Möglichkeiten, die das Verfahren prinzipiell bietet.

Eines der Hauptprobleme bei der Programmierung von Lichtgriffeln oder eben einer „Lichtpistole" liegt in der Beherrschung der speziellen Programmiertechnik. Das Spiel wird nach dem Auslösen des Abzugs unterbrochen. Würde der Bildschirm ständig abgetastet, wie es bei Verwendung des Lichtgriffels der Fall ist, wäre das Spiel zu langsam. Folglich muß das Programm „den Schirm einfrieren", sobald der Auslöser des Lichtgewehrs betätigt ist, damit eine Überprüfung stattfinden kann, ob Ziel und Light Rifle in der Linie übereinstimmen. Erst nach diesem internen Check kann das Spiel fortgesetzt werden. Theoretisch wäre dieser Vorgang von sehr kurzer Dauer, da der Abtast- und Überprüfungsvorgang nicht viel Zeit in Anspruch nimmt. Die Spiele zeigen allerdings, daß es in der Praxis doch noch anders aussieht.

Schuß im Dunkel

Die Fotozelle in der Lichtpistole erkennt die zeilenweise Erneuerung des Bildes auf dem Schirm. Mit Hilfe der Software wird der Zeilenverlauf des Screens ständig computerintern überprüft, damit das Raster des Bildschirms gleichbleibend ist. Signalisiert die Lichtpistole, daß ein bestimmter Rasterbereich erkannt wurde, setzt die Software in die X- und Y-Bildschirmkoordinaten um.

Jagdzeit



„High Noon" ist das beste der drei Stack-Demonstrationsprogramme. Die Animation ist akzeptabel, und der Revolverheld „schießt" zurück. „Grouse Shoot" wie „Shooting Gallery" bieten jeweils nur ein Ziel, das getroffen werden muß, bevor es den Boden erreicht. Die Animation ist allerdings dürftig.

Such-Routinen

Die zum Suchen eines bestimmten Verzeichnisses benötigte Zeit kann durch Verwendung einer „binären Suchmethode" erheblich reduziert werden – vorausgesetzt, die Datei wurde zuvor bereits sortiert.

Die drei wesentlichsten Funktionen des Adreßbuch-Programms sind nun fertiggestellt. Dies sind: Hinzufügen eines neuen Verzeichnisses, Speichern auf Cassette oder Diskette und Einlesen der Daten beim Programmstart. Ein Adreßbuch hat jedoch keinen Nutzen, wenn man nur Verzeichnisse hinzufügen und jedoch keine herauslesen kann. Aus diesem Grund ist eine Routine zum Finden eines Verzeichnisses notwendig.

Das Auffinden eines Verzeichnisses anhand des Namens ist wohl die meistgebrauchte Funktion. Aus diesem Grund ist dies auch die erste Option im Auswahlmenü. Das Suchen ist in vielen Computerprogrammen eine der wichtigsten Funktionen, ganz besonders in Datenbank-Programmen, wo ständig auf spezielle Daten Zugriff genommen werden muß. Allgemein gesagt gibt es zwei Suchmethoden – die lineare und die binäre Methode. Bei der linearen Methode wird jedes Element und Verzeichnis untersucht, bis die gestellte Vergleichsbedingung zutrifft und somit das gesuchte Verzeichnis gefunden worden ist. Sind die einzelnen Verzeichnisse nicht in eine bestimmte Reihenfolge gebracht worden, so ist die lineare Suchmethode die einzige, die garantiert zum Erfolg führt. Bei ihr verhält sich die zum Finden eines bestimmten Verzeichnisses in einer Datei mit N Verzeichnissen benötigte Zeit proportional zu N/2. Wenn sich die Gesamtanzahl der Verzeichnisse in Grenzen hält, so hat diese Formel zweifelsohne Gültigkeit. Erhöht sich jedoch die Anzahl der Verzeichnisse, kann die zum Suchen notwendige Zeit sehr lang werden.

Die binäre Suchmethode

Wenn die Daten der Datei bereits in einer bestimmten Reihenfolge vorliegen, kann man wesentlich effizienter suchen: mit der binären Suchmethode. Stellen Sie sich vor, Sie wollten in einem Lexikon die Definition für den Begriff „liquide" finden. Sie werden dann kaum auf der ersten Seite zu suchen beginnen, sondern das Buch ungefähr in der Mitte aufschlagen. Befindet sich auf der Seite zum Beispiel der Begriff „Mond", so steht fest, daß Sie das Lexikon zu weit hinten aufgeschlagen haben. Das gesuchte Wort kann sich somit nicht in der zweiten Hälfte des Buches befinden, sondern muß irgendwo in der ersten Hälfte stehen. Entsprechend werden Sie jetzt das Lexikon ungefähr in der Mitte der ersten Hälfte aufschlagen und die Eintragungen mit dem sogenannten Suchschlüssel, der in diesem Fall das Wort „liquide" ist, vergleichen. Da hier alle Begriffe alphabetisch geordnet sind, läßt sich leicht feststellen, ob der aufgeschlagene Abschnitt zu „hoch" oder zu „niedrig" liegt. Auf ähnliche Weise vergleicht auch das Programm die Eintragungen mit den zu findenden Begriffen.

Jedesmal, wenn ein Verzeichnis untersucht wird, wird der „Suchschlüssel" (die Eintragung, die auf der Datei herausgesucht werden soll) mit dem „Verzeichnisschlüssel" verglichen, um festzustellen, ob sie übereinstimmen. Trifft der Vergleich zu, so erhält man die gewünschte Information.

Das System des Durchsuchens einer Computer-Datei nach einem bestimmten Datensatz bzw. Verzeichnis ist offensichtlich, vorausgesetzt, die einzelnen Datensätze wurden zuvor in alphabetischer Reihenfolge sortiert.

Die für das Adreßbuch-Programm benötigte Suchroutine wird nun doch erheblich komplizierter, da verschiedene Bedingungen erst jetzt zum Tragen kommen. Die erste Aufgabe der Suchroutine ist, nach dem zu suchenden Namen zu fragen (wir wollen die Suchroutine *SUCHVER* nennen). Dies wird der Suchschlüssel genannt. Stellen Sie sich vor, daß sich irgendwo in der Datei ein Verzeichnis für eine Person mit dem Namen Peter Fischer befindet. Das Verzeichnis für diese Person wird ein Feld mit dem Inhalt FISCHER PETER enthalten (mit dem Namen in standardisierter Form). Die Suchroutine wird Sie also z. B. als erstes fragen "NACH WELCHEM VERZEICHNIS SUCHEN SIE?", und Sie könnten dann mit PETER FISCHER, P. FISCHER oder Pete Fischer antworten. Nach Eingabe des Namens PETER FISCHER wird die Suchroutine diesen in die Standardform FISCHER PETER umwandeln. Danach wird sie Ihre Eingabe, den Suchschlüssel, mit den verschiedenen Inhalten der MODNAM$-Felder vergleichen. Wenn das Programm die lineare Suchmethode verwenden sollte, so würde jedes MODNAME$-Feld so lange mit dem Suchschlüssel verglichen, bis eine Übereinstimmung gefunden wäre oder aber feststünde, daß keine exakte Übereinstimmung vorhanden ist.

Wie wir jedoch schon feststellten, ist die lineare Suchmethode im Vergleich zur binären

nicht sehr effizient, vorausgesetzt, die Verzeichnisse wurden bereits sortiert. Dies kann direkt in der Suchroutine überprüft werden, indem der Vergleich IF VMOD=1 THEN GOSUB *SRTVER* integriert wird. Das Programm weiß, daß das niedrigste Element im Bereich von MODFLD$(1) und das höchste bei MODFLD$ (GROSS-1) liegt. Für die Suchroutine braucht man drei Variablen: BTM für die untere Grenze des Bereiches (MODFLD$(1) als Startwert); TOP für die obere Grenze des Bereiches (MODFLD$(GROSS-1) als Startwert); und MID für den Wert, der das mittlere Element kennzeichnet.

Vergleich der Einträge

Mit dem Setzen der Variablen BTM=BEREICH(1) und TOP=BEREICH(GROSS-1) wird festgelegt, daß die Suche im Bereich zwischen dem kleinsten und dem größten Element stattfinden soll. Daher kann man auch sagen LET BTM=1 und LET TOP=GROSS-1.

Stellen Sie sich vor, daß sich 21 gültige Eintragungen im Adreßbuch befinden. GROSS hat einen Wert von 22. BTM wird den Wert 1 haben, TOP entsprechend den Wert 21. Der Wert von MID, die Position des mittleren Elementes, kann in BASIC mit INT((BTM+TOP)/2) bestimmt werden. Wenn der Wert von BTM gleich 1 ist und der Wert von TOP gleich 21, enthält MID den Wert 11.

Um die binäre Suche einzuleiten, nehmen wir als erstes an, daß die gesamte Datei gültig ist. Dann finden wir den Mittelpunkt mit INT((BTM+TOP)/2) in einer Schleife heraus, die erst dann abgebrochen wird, wenn der gesuchte Name gefunden wurde oder feststeht, daß der gesuchte Name nicht vorhanden ist. Danach ist zu überprüfen, ob der Suchschlüssel (SUSCHL$) mit dem Inhalt des durch MID angegebenen Verzeichnisses übereinstimmt. Ist der MID-String zu klein, steht fest, daß BEREICH(MID) der kleinste Teil des zu durchsuchenden Bereiches ist. Der Wert von BTM kann also auf den von MID gesetzt werden. Noch besser wäre, wenn man den Wert von BTM auf den Wert MID+1 setzen würde, da der Inhalt von BEREICH(MID) ja ohnehin nicht mit dem Suchschlüssel übereinstimmt. Ähnlich verhält es sich bei IF BEREICH(MID) › SUSCHL$. Hier kann TOP auf den Wert MID-1 gesetzt werden.

Als kleinen Schritt in Richtung einer vollständigen Programmroutine kann das gezeigte Programm eine Testeingabe annehmen (die jedoch exakt in der Form sein muß wie MODFLD$) und gibt bei erfolgloser Suche die Meldung VERZEICHNIS NICHT GEFUNDEN oder bei erfolgreicher Suche die Meldung VERZEICHNIS IST NR.(MID) aus. Da die Routine mit Zeilennummer 13 000 beginnt, kann sie an das Ende des Programms angefügt werden. Zeile 4040 muß dann allerdings in IF WAHL=1 THEN GOSUB 13 000 abgeändert werden.

Zeile 13 240 beinhaltet eine STOP-Anweisung. Diese Anweisung unterbricht die Programmausführung, sobald die Meldung VERZEICHNIS NICHT GEFUNDEN oder VERZEICHNIS IST NR. (MID) dargestellt wird. Der Programmlauf kann dann ohne Datenverlust mit derselben Zeilennummer fortgesetzt werden, indem man CONT eingibt und RETURN drückt. Ohne die STOP-Anweisung würde das Programm sofort zu der RETURN-Anweisung in Zeile 13 250 kommen, und die Meldung würde zu kurz dargestellt, um sie vollständig lesen zu können.

In Zeile 13 100 wird BTM auf 1 gesetzt, entsprechend dem Wert des niedrigsten Elementes der Datei. TOP wird in Zeile 13 110 auf den Wert von GROSS-1 gesetzt. Dies ist die Position innerhalb des MODFLD$-Bereiches, an dem sich das höchstgelegene Element befindet. Zeile 13 120 beinhaltet eine Schleife, die erst dann abgebrochen wird, wenn entweder ein Vergleich zutrifft oder keine exakte Übereinstimmung möglich ist.

In Zeile 13 130 wird der Mittelpunkt des Bereiches errechnet, indem die Gesamtanzahl der Verzeichnisse durch 2 dividiert wird (INT wird zum Runden verwendet, damit nicht beispielsweise ein Wert 1,5 als Ergebnis heraus-

BASIC-Dialekte

SPECTRUM

Beim Spectrum sind die folgenden Modifikationen notwendig:

```
13000 REM *FNDVER* TEST-
      VERSION
13010 IF VMOD=1 THEN GOSUB
      11200
13020 INPUT "EINGABE SUCH-
      BEGRIFF ";S$

13100 LET BTM=1
13110 LET TOP=GROSS-1
13120 FOR L=1 TO 1
13130 LET MID=INT
      ((BTM+TOP)/2)
13140 IF M$(MID)‹›S$ THEN LET
      L=0
13150 IF M$(MID)‹S$
      THEN LET BTM=MID+1
13160 IF M$(MID)›S$
      THEN LET TOP=MID-1
13170 IF BTM › TOP THEN LET
      L=1
13180 NEXT L
13200 IF BTM › TOP THEN PRINT
      "VERZEICHNIS NICHT
      GEFUNDEN"
13210 IF BTM ‹ = TOP THEN
      PRINT
      "VERZEICHNIS IST NR. ";
      MID
13240 STOP
13250 RETURN
```

Beachten Sie noch einmal das Problem der Benennung von String-Variablen beim Spectrum. Da immer nur ein Buchstabe verwendet werden kann, ist hier S$ als Ersatz für SUSCHL$ verwendet worden.

kommt). Beim ersten Durchlauf besteht selbstverständlich eine gewisse Chance, daß der Inhalt von MODFLD$(MID) mit dem Suchschlüssel (SUSCHL$) übereinstimmt. Ist dies jedoch nicht der Fall, so wird L auf den Wert 0 gesetzt, um sicherzustellen, daß die Schleife nicht abgebrochen wird. Wenn der Vergleich in Zeile 13 140 fehlschlägt, hat MODFLD$ entweder einen höheren oder einen niedrigeren Wert als der Wert von SUSCHL$. Der Wert von BTM wird dann entsprechend um eins erhöht, oder der Wert von TOP wird um eins reduziert. Der Grund, weshalb der Wert von MID nicht verwendet wird, ist, daß der Vergleich in Zeile 13 140 bereits gezeigt hat, daß MODFLD$(MID) nicht den gesuchten Inhalt hat. Deshalb ist es nicht sinnvoll, dieses Element beim nächsten Schleifendurchlauf noch einmal zu untersuchen.

Wenn kein Vergleich zutreffend war, wird der Wert von BTM eventuell größer werden als der von TOP. Danach kann die Schleife abgebrochen (Zeile 13 170) und die Meldung VERZEICHNIS NICHT GEFUNDEN ausgegeben werden (Zeile 13 200).

Standardisierte Eintragungen

Das hier gezeigte Programm-Fragment dient zum Test der Suchroutine. Wichtig ist die Darstellung des gesamten Verzeichnisses, wie es ursprünglich eingegeben wurde. Wenn die Nummer des Verzeichnisses bekannt ist, ist es nicht mehr schwierig, weitere benötigte Informationen zu erhalten (z. B. NAMFLD$, STRFLD$ usw.). Unter der Darstellung des Verzeichnisses sollen Meldungen wie beispielsweise LEERTASTE, UM FORTZUFAHREN (zurück zum Hauptmenü) und eventuell weitere Optionen, wie z. B. DRUECKE "P" ZUM AUSDRUCKEN ausgegeben werden.

Nun muß noch die Entscheidung getroffen werden, wie die Eingabe von *FNDVER* zu handhaben ist. In dem gezeigten Programmfragment muß die in Zeile 13 020 erwartete Eingabe in standardisierter Form sein – FISCHER PETER beispielsweise. Es wird jedoch kaum jemand den Namen in der Reihenfolge Familienname/Vorname oder aber komplett in Großbuchstaben schreiben. Ferner würde die geringste Abweichung vom Originalformat die Meldung VERZEICHNIS NICHT GEFUNDEN ergeben. Die beiden erstgenannten Probleme könnten, zumindest dem Anschein nach, durch die Routine *MODNAM* bewältigt werden. Das dritte Problem ist viel interessanter, aber auch schwieriger in den Griff zu bekommen.

Bevor wir uns mit diesem Problem befassen, wollen wir untersuchen, warum *MODNAM* die ersten beiden Probleme nicht lösen kann. Wenn Sie *MODNAM* noch einmal betrachten (beginnend bei Zeile 10 200), werden Sie einen der meistgemachten Fehler bei der Programmierung entdecken – mangelnde allgemeine Verwendbarkeit einer Programmroutine. Diese Unterroutine sollte in der Lage sein, „normale" Namen in „standardisierte" Namen umzuwandeln. Es wird angenommen, daß der umzuwandelnde Name immer in NAMFLD$(GROSS) steht und der umgewandelte Name in MODFLD$(GROSS) abgelegt wird. Mit dieser Situation vor Augen hat der Programmierer drei Möglichkeiten zur Auswahl: Entweder kann er die Routine vollkommen neu schreiben, um sie allgemein verwendbar zu machen. Das würde jedoch sicherlich weitere Änderungen an anderen Routinen erfordern. Genauso aber könnte er eine ähnliche Routine schreiben, um die Eingabe von *FNDVER* zu handhaben. Damit verbunden ist natürlich ein erheblicher Mehraufwand an Arbeitszeit und unnötiger Speicherplatzverbrauch. Als letztes könnte er einige umständliche Programmiertechniken anwenden, um die bestehende Routine „hinzubiegen". Die zuletzt genannte Alternative ist die unattraktivste Lösung. Das Problem wird zwar gelöst, aber die Routine wird dadurch schwierig zu verstehen, selbst für den Programmierer. Und es dürfte zu einem Alptraum für einen anderen User werden, wenn einmal etwas am Listing zu ändern ist.

Als Leitregel sei deshalb empfohlen: Schreiben Sie Unterroutinen so allgemein wie irgend möglich, so daß sie von jedem Teil des Programms aus aufgerufen werden können.

Um Ihnen einen Eindruck von schlechten Programmiertechniken zu geben und um Ihnen zu zeigen, wie unklar ein Programm dadurch werden kann, betrachten Sie Zeile 13 020 des Programms. Dort steht INPUT "EINGABE SUCHBEGRIFF:"; SUSCHL$. Betrachten Sie dann die Modifikationen, die die Routine *MODNAM* einsetzbar machen sollen:

```
13020 INPUT "EINGABE SUCHBEGRIFF:";
      NAMFLD$(GROSS)
13030 GOSUB 10200:REM *MODNAM*
      UNTERROUTINE
13040 LET SUSCHL$=MODFLD$(GROSS)
13050 . . .
```

Glücklicherweise ist der Wert von GROSS immer um eins höher als die Zahl der gültigen Verzeichnisse. Mit anderen Worten, befindet sich an dieser Position kein Verzeichnis, so daß ein bestehendes Verzeichnis auch nicht versehentlich verändert werden kann. Aber stellen Sie sich vor, wie verwirrend dieses Programm ohne einige erklärende REM-Anweisungen für jemanden ist, der nicht mit der Entwicklung des Programms vertraut ist!

Doch zurück zu dem Problem der Handhabung von „leichten Abweichungen" vom vorgeschriebenen Eingabeformat. Stellen Sie sich vor, Sie hätten in der ADDVER-Routine einen Namen in der Form Pete Fischer und als Suchschlüssel dann Peter Fischer eingegeben. Die

Eingabe würde dann jeweils in die Standardform FISCHER PETE und entsprechend FISCHER PETER umgewandelt. Wegen des Unterschiedes würde kein exakter Vergleich möglich sein und das Verzeichnis nicht gefunden, obwohl es in der Datei enthalten ist. Wir werden nicht versuchen, dieses Problem zu lösen, da eine zufriedenstellende Lösung einen erheblichen Programmieraufwand erfordern würde. Für experimentierfreudige, interessierte Leser geben wir hier nur einige Anregungen, die Sie bei Ihren selbstgeschriebenen Programmen berücksichtigen können:

```
BEGIN (STARTE) (durchsuche Bereich nach
  exakter Übereinstimmung)
  IF (WENN) exakte Übereinstimmung
    gefunden
    THEN PRINT (DANN DRUCKE)
      gesamtes Verzeichnis
    ELSE (SONST) durchsuche Bereich
      nach annähernder Übereinstimmung
      IF (WENN) annähernde
        Übereinstimmung gefunden
        THEN PRINT (DANN DRUCKE)
          entsprechendes Verzeichnis
        ELSE PRINT (SONST DRUCKE)
          "VERZEICHNIS NICHT
            GEFUNDEN"
      ENDIF (SO LANGE BIS FERTIG)
  ENDIF (SO LANGE BIS FERTIG)
END (ENDE)
```

Die Routine zum Suchen einer annähernden Übereinstimmung könnte wie folgt aussehen:

```
BEGIN (STARTE) annähernde
  Übereinstimmung)
  Durchsuche Bereich nach einer exakten
  Übereinstimmung des Familiennamens
  IF (WENN) exakte Übereinstimmung des
    Familiennamens gefunden
    THEN (DANN) durchsuche Vornamen
      nach entsprechendem Vergleich
    PRINT (DRUCKE) entsprechendes
      Verzeichnis
    ELSE (SONST) durchsuche
      Familiennamen nach größter
      Übereinstimmung
      IF (WENN) Familienname mit
        maximaler Übereinstimmung
        gefunden
        THEN PRINT (DANN DRUCKE)
          entsprechendes Verzeichnis
      ENDIF (SO LANGE BIS FERTIG)
  ENDIF (SO LANGE BIS FERTIG)
END (ENDE)
```

Die Routine für „maximale Übereinstimmung" kann ungefähr so definiert werden, als wolle man das Element finden, das die maximale Anzahl gemeinsamer Zeichen mit dem Suchschlüssel hat. Genauso kann eine Situation angenommen werden, in der der Suchschlüssel vollständig in einem Element vorhanden ist.

Bei dem hier gezeigten Programm-Fragment gibt es jedoch noch ein Problem. Stellen Sie sich vor, die folgenden Ereignisse würden eintreten: In der Datei befinden sich zehn Verzeichnisse. Sie starten das Programm und verwenden *ADDVER* zum Eingeben eines neuen Verzeichnisses. Direkt danach verwenden Sie *FNDVER* zum Lokalisieren eines Verzeichnisses. Wenn dann abschließend *ENPROG* ausgeführt wird (um die Datei zu sichern und den Programmlauf abzubrechen), wird das neu eingegebene Verzeichnis nicht gespeichert (obwohl alle anderen Verzeichnisse gespeichert werden). Dies resultiert direkt aus einem Fehler, der bei der Ausführung von *FNDVER* passierte. Erkennen Sie, weshalb das neu eingegebene Verzeichnis nicht mit abgespeichert wird?

Programmoptimierung

Im nächsten Teil dieses Kurses werden wir Ihnen zeigen, wie man einen derartigen Datenverlust verhindern kann. Vielleicht machen Sie sich bereits einige Gedanken darüber, wie man ein Verzeichnis löschen oder modifizieren könnte. Weitere Optionen des Hauptmenüs (beispielsweise *FNDSTD* usw.) sind bereits entwickelten Routinen sehr ähnlich. Wir überlassen diese Aufgaben Ihnen. Wenn Sie diese Routinen benötigen, so können Sie sie in das Programm integrieren.

Abschließend noch eine Aufgabe: Überlegen Sie, was wohl passiert, wenn die Datei genau 50 Verzeichnisse beinhaltet und die *FNDVER*-Routine verwendet wird. (Hinweis: GROSS hat dann den Wert 51.)

```
13000 REM TESTFASSUNG VON *FNDVER*
13010 IF VMOD =1 THEN GOSUB 11200
13020 INPUT "EINGABE SUCHBEGRIFF ";SUSCHL$
13030 REM
13040 REM
13050 REM
13060 REM
13070 REM
13080 REM
13090 REM
13100 LET BTM=1
13110 LET TOP=GROSS-1
13120 FOR L=1 TO 1
13130 LET MID=INT((BTM+TOP)/2)
13140 IF MODFLD$(MID) ‹› SUSCHL$ THEN L=0
13150 IF MODFLD$(MID) ‹ SUSCHL$ THEN BTM=MID+1
13160 IF MODFLD$(MID) › SUSCHL$ THEN TOP=MID-1
13170 IF BTM › TOP THEN L=1
13180 NEXT L
13190 REM
13200 IF BTM › TOP THEN PRINT "VERZEICHNIS NICHT GEFUNDEN"
13210 IF BTM ‹ = TOP THEN PRINT "VERZEICHNIS IST NR. ";MID
13220 REM
13230 REM
13240 STOP
13250 RETURN
```

Geschäftsberichte

Kostenanalysen und Listenverarbeitung im Betrieb werden durch kommerzielle Buchhaltungsprogramme auf Cassetten oder Disketten für Microcomputer erheblich erleichtert.

Elektronische Buchhaltungssysteme wie „Cash Trader" von Quick Count, „Microledger" von Lewis Ashley und „Accountant" von Compact Accounting Services sollen nicht nur Geld- und Warenbewegungen speichern können, sondern auch Listen und Analysen liefern. Diese Listen lassen sich grob in zwei Bereiche unterteilen: Das Management benötigt zunächst Listen für den internen Arbeitsablauf, während für Organisationen und Institutionen außerhalb der Firma Bilanzen, Mehrwertsteuervoranmeldungen etc. erstellt werden müssen.

Wie bei allen Computersystemen können die Listen aber nur so ausführlich sein wie die Daten, die in das System eingegeben bzw. gespeichert wurden. Die Programme Cash Trader und Accountant bieten beispielsweise nicht die Möglichkeit, für jeden Kunden und Lieferanten ausführliche Dateien anzulegen, und können daher nur einen kleinen Teil der Listen produzieren, die ein normales Buchhaltungsprogramm liefert. Microledger dagegen kann umfassende Informationen speichern. Alle drei Programmpakete liefern jedoch zur Überprüfung der Dateneingaben einen „Protokollausdruck".

Microledger unterhält für die laufende Abrechnungsperiode einen Überblick über alle Vorgänge in den Kunden- und Lieferantenkonten. Es kann Rechnungen mit dem aktuellen Stand eines Kundenkontos drucken und dabei alle Bestellungen und Käufe aufführen. Außerdem druckt es bei Lieferantenschecks eine Liste der Lieferungen, die der Scheck begleichen soll. Microledger liefert auch Kontenlisten in numerischer und alphabetischer Reihenfolge und kann auf der Grundlage der Primanotadatei eine aktuelle Kontenstandsliste, eine Zwischenbilanz und eine betriebswirtschaftliche Analyse nach unterschiedlichen Gesichtspunkten erstellen.

Mit dem Cash Trader lassen sich eine Zusammenstellung der Wocheneinnahmen, die Primanotadatei und eine Zwischenbilanz drukken, außerdem Kontoauszüge aller Bargeld- und Kontenbewegungen, die Zwischensummen der Mehrwertsteuerkonten aufgeschlüsselt nach Steuersätzen und der Monats- bzw. Quartalsabschluß.

Das Programm Accountant liefert Aufstellungen aller Geschäftsvorgänge, listet die Einzelheiten eines bestimmten Kontos und druckt eine Zwischenbilanz. Über die Tageskasse kann ein Protokolldruck der einzelnen Kontenklassen (alle Vorgänge eines bestimmten Kontos) angefertigt werden. Ein Ausdruck der Summenkonten, der die Einträge in das Hauptbuch zeigt, und eine Mehrwertsteuerliste mit den Steuersätzen der einzelnen Waren sind ebenfalls möglich.

Die Mahnliste ist ein gutes Beispiel für Informationen, die für die Geschäftsführung wichtig sein können. Sie kann per Hand nur mit großem Aufwand erstellt werden, wird von einem Computersystem aber automatisch geliefert. In einer Mahnliste sind die ausstehenden und die geschuldeten Beträge nach ihrer (Über-) Fälligkeit von 30, 60 und 90 Tagen aufgeschlüsselt. Das Programm untersucht dabei die Daten der einzelnen Käufe und Verkäufe und sortiert sie nach Datum geordnet in eine der drei Kategorien ein. Ein als bezahlt gekennzeichneter Betrag wird automatisch aus der Liste gelöscht.

Automatisches Summieren

Alle drei Systeme können eine Zwischenbilanz erstellen. Darin werden alle Kontenbezeichnungen und der jeweilige Gesamtsaldo ausgedruckt. Außer der Zwischenbilanz sind für eine Geschäftsführung die Gewinn- und Verlustrechnung und die endgültige Bilanz interessant. Der Cash Trader vereinfacht seine Gewinn- und Verlustrechnung, indem er Einnahmen und Ausgaben summiert und die Differenz bildet.

Nicht alle Listen lassen sich so einfach erstellen wie die Zwischenbilanz und die Mahnliste. Wie auch bei einigen anderen Listen (z. B. den Kontenbewegungen) werden dabei nur die einzelnen Datensätze mit einer gemeinsamen Überschrift versehen und auf dem Drucker ausgegeben. Listen mit mehr Detailinformationen und genauerer Aufschlüsselung sind schwieriger zu erstellen, da die Zusammenstellung der Auswahlkriterien (unabhängig von dem eingesetzten Programmpaket) um so komplizierter wird, je präziser die Analyse sein soll.

Sowohl der Accountant als auch der Microledger verfügen über Möglichkeiten der verfeinerten Analyse und Planung. Der Einsatz dieser Programmteile ist für den Anwender allerdings mit viel Arbeit verbunden.

Bei dem Accountant läßt sich die Primanota-

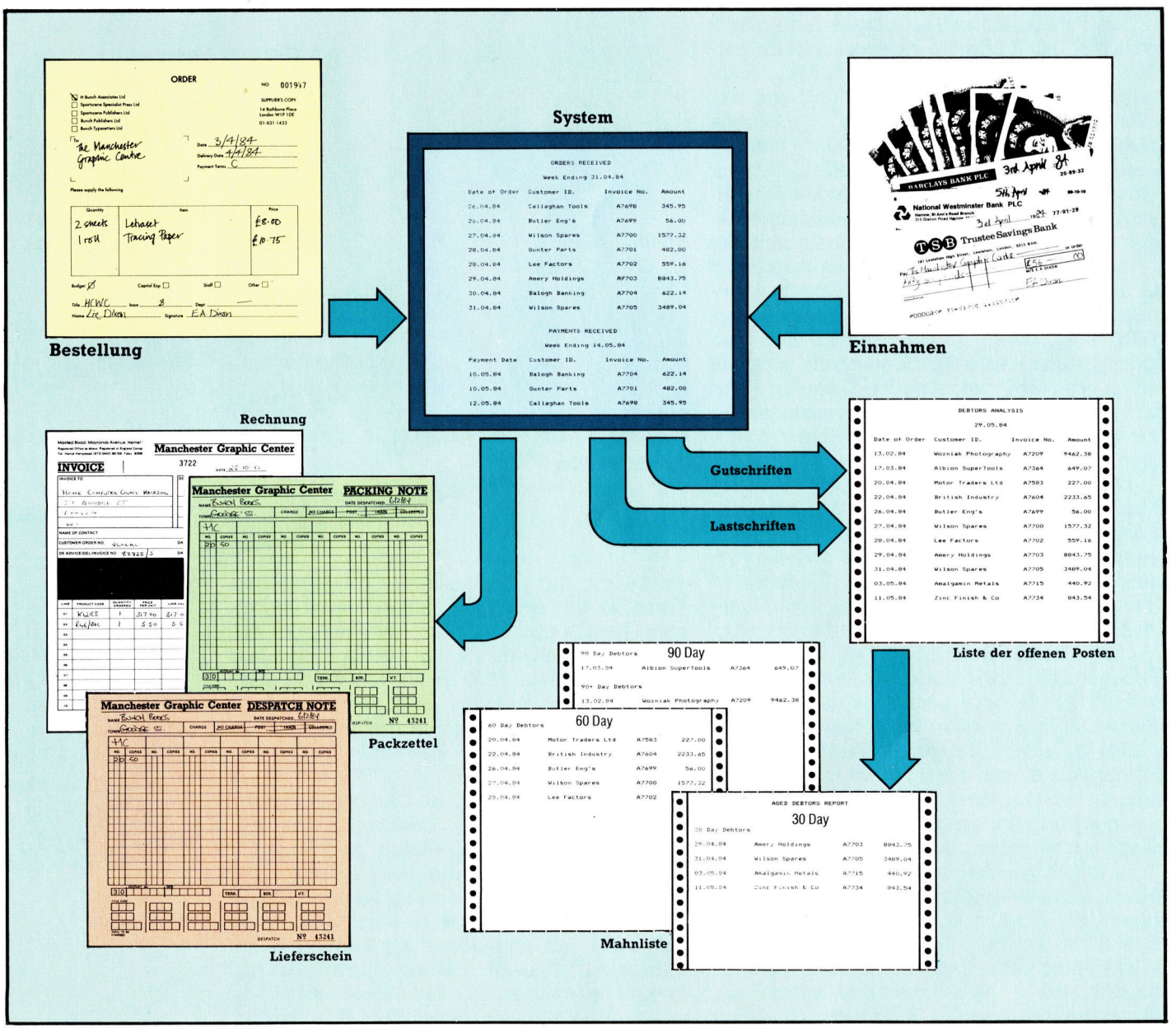
datei über einen Gruppenschlüssel analysieren. Nehmen wir an, ein Geschäft hat drei Abteilungen: Fabrikation, Vertrieb und Lagerhaltung bzw. Einkauf. Über die Gruppenschlüssel können Sie herausfinden, welche Kosten jede Abteilung verursacht. Zwar gibt es Ausgaben, die alle Abteilungen gemeinsam haben, aber mit der Zuteilung des Gruppenschlüssels 01 an den Einkauf, 02 an die Fabrikation und 03 an den Verkauf lassen sich die Anteile der einzelnen Abteilungen an den Gesamtkosten leicht identifizieren. Es kann so beispielsweise festgestellt werden, daß die Fabrikation vier Fünftel der gesamten Elektrizität verbraucht. Der Accountant kann so eine aufgeschlüsselte Zwischenbilanz erstellen, deren Konten die Teilsummen der einzelnen Abteilungen enthalten.

Integrierte Gruppenschlüssel

Auch Microledger hat diese Möglichkeit. Bei diesem Programm sind die Gruppenschlüssel in die Kontonummern integriert. Jedem Konto des Microledger ist eine dreistellige Nummer zugeordnet. Es gibt zwei Ebenen der Analyse. Die erste Ebene wird von einer zweistelligen Zahl gekennzeichnet, die der Kontonummer vorangestellt ist. Die zweite Ebene wird von einer dreistelligen Nummer gesteuert, die zwischen der ersten Analysenummer und der Kontonummer steht (Beispiel: aa/bbb/111).

Für eine Geschäftsanalyse könnten diese beiden Ebenen folgendermaßen eingesetzt werden: Nehmen wir an, es gibt nur ein Eingangskonto, aber die Firma hat vier Verkäufer. Die erste Analysenummer dient zur Unterscheidung des Eingangskontos von allen anderen Kontenklassen, während sich mit der zweiten Analysennummer der Umsatz der einzelnen Verkäufer aufschlüsseln läßt. Anhand dieser Liste kann die Geschäftsleitung feststellen, ob der Verkäufer sein Soll erfüllt hat.

Eine Buchhaltung per Computer kann ohne viel Aufwand einen Überblick über den aktuellen Stand einer Firma liefern. Das im Bild gezeigte System speichert Informationen über alle Geschäftsvorgänge und stellt automatisch eine Mahnliste für überfällige Zahlungen zusammen. Größere Systeme schreiben Rechnungen, Lieferscheine und Packzettel auf speziell dafür abgestimmten Formularen.



Firmengründungen

In der Geschichte der Microcomputer sind die Entwicklungen von Hard- und Software untrennbar miteinander verbunden.

Technische Veränderungen haben in der Vergangenheit die Menschen häufig in Erstaunen versetzt. Bis heute aber kann keine dieser Entwicklungen in ihrer Geschwindigkeit mit der der Microelektronik-Revolution Schritt halten – nicht einmal die Fliegerei, die binnen kurzer Zeit von den Gebrüdern Wright zur ersten Mondexpedition führte. In nur einem Jahrzehnt verwandelten sich die ersten primitiven Microprozessoren in leistungsstarke 16-Bit-Rechner.

1971 kamen mehrere Chip-Hersteller in Kalifornien zu der Erkenntnis, daß die wesentlichen Funktionen eines Computers in einem winzigen Stück Silizium zusammengefaßt werden könnten. Damals gab es noch keine großartigen Pläne für eine „Revolution", und das Wort „Informationstechnologie" existierte nicht. Grundgedanke war, einen kleinen, preiswerten Computer herzustellen, der zur Steuerung von Fertigungsmaschinen oder Aufzügen imstande war. Die ersten Microprozessoren wurden diesen Aufgaben auch gerecht.

Intel, einer dieser Chiphersteller, steht heute in dem Ruf, den ersten Microprozessor überhaupt entwickelt zu haben. Seine Bezeichnung: 4004. Die „Vieren" in der Angabe beziehen sich auf die Kapazität. Es handelte sich um einen Vier-Bit-Prozessor, der Daten in Form von vier Binäreinheiten verarbeiten konnte. Für die Steuerung eines Aufzugs reichte die winzige Speicherkapazität völlig.

1972 entwickelte Intel einen Acht-Bit-Microprozessor, den 8008-Chip. Amateurelektroniker machten sich zu diesem Zeitpunkt erstmals Gedanken darüber, wie man mit diesem Chip Computer für den Eigenbedarf bauen könnte. In verschiedenen amerikanischen Elektronikzeitschriften wurden entsprechende Artikel veröffentlicht. Die vorgestellten Rechner waren zwar weder mit Monitoren, professioneller Tastatur oder anderen nützlichen Elementen ausgestattet, doch stellten sie die ersten Vorläufer der eigentlichen Heimcomputer dar. Aus diesen von Bastlern gebauten Prototypen entstand der erste „kommerzielle" Microcomputer mit dem Namen Altair 8800. Allerdings war er nur als Bausatz erhältlich.

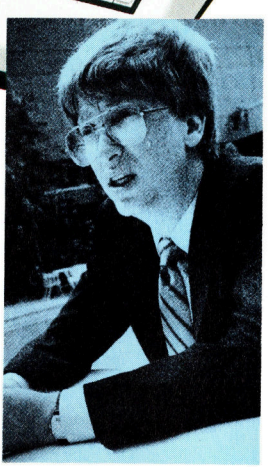
Im folgenden Jahr wurde der erste „echte" Microprozessor präsentiert, der 8080, wiederum ein Produkt von Intel. Er verarbeitete Acht-Bit-Datenblocks und vermochte bis zu 64 KByte Speicherkapazität zu steuern, was bedeutete, daß größere Programme nutzbar gemacht werden konnten. Zu diesem Zeitpunkt stiegen auch andere Chip-Hersteller ins Geschäft ein. Motorolas 6800-Chip war dem 8080 fast ebenbürtig. Die Hardware-Eigenschaften waren weitgehend ähnlich, aber die Eingaben mußten auf eine andere Art erfolgen. Hier begann das Problem mit der Software-Kompatibilität: Für den 8080 geschriebene Programme liefen auf dem 6800 nicht, und umgekehrt.

Gleichzeitig hatten andere Firmen ähnliche Prozessoren entwickelt, so etwa National Semiconductor, Signetics und Advanced Micro Devices. Der entscheidende nächste Schritt wurde von MOS Technology vollzogen, jener Firma, in der seinerzeit Chuck Peddle arbeitete. Peddle gehörte dem Unternehmen zu einem Zeitpunkt an, als dort ein Prozessor mit der Bezeichnung 6500 entwickelt wurde, der dem 6800 von Motorola so ähnlich war, daß man gezwungen war, einige Änderungen einzubauen. Das Ergebnis bezeichnete man als 6502.

In der Büromaschinen- und Taschenrechner-Branche Kanadas hatte Commodore derzeit bereits einen guten Ruf. Peddle wechselte zu diesem Unternehmen über, weil er eine Idee verwirklichen wollte: Er beabsichtigte, einen „persönlichen" Computer zu entwickeln. Dieser sollte mit folgendem ausgestattet sein: Bildschirm, Tastatur, Cassettenlaufwerk für Programmspeicherung und alles, was ein „richtiger" Computer sonst benötigt – gebaut

Chuck Peddle entwikkelte sowohl den Commodore PET wie dessen Herzstück, den 6502-Microprozessor. Doch Bill Gates' Beitrag, das in PETs ROM integrierte Microsoft-BASIC, war ebenso wichtig.

Chuck Peddle

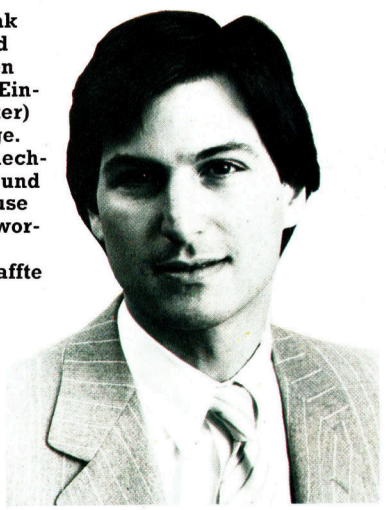


Bill Gates

um den 6502-Prozessor. 1976 kam der Rechner als PET 2001 auf den Markt.

Kaum war der erste PET lieferbar, bereiteten zwei andere Erfinder in einer kalifornischen Garage einen eigenen Computer vor. Stephan Wozniak wollte schon immer einen Rechner haben und lernte alles darüber im Homebrew Computer Club. Er entwickelte einen Einplatinencomputer, den er gemeinsam mit seinem Freund Steven Jobs baute und verkaufte. Sie nannten das System Apple I. Nachdem man die Platine in ein Gehäuse verpackt und mit einer Tastatur versehen hatte, entwickelte sich daraus der überaus erfolgreiche Apple II. Dieser Rechner kam kurz nach Peddles PET auf den Markt und war eine Art Grundstein für die Soft- und Hardwareindustrie.

Die Tandy Corporation in Forth Worth, Texas, hatte eigene Ideen für den noch kleinen Computer-Markt. Bis heute ist das Unternehmen ein Filialist, der in seinen Läden alle Arten elektronischer Geräte anbietet, von HiFi über Synthesizer bis hin zum einfachen Radio. Die Einbeziehung des Computers war also nur folgerichtig. Ergebnis war der TRS-80 Modell 1,



Stephan Wozniak entwickelte und baute den ersten Apple I (einen Einplatinencomputer) in seiner Garage. Nachdem der Rechner modifiziert und in einem Gehäuse untergebracht worden war – der Apple II –, schaffte er mit seinem Freund Steven Jobs den geschäftlichen Durchbruch.

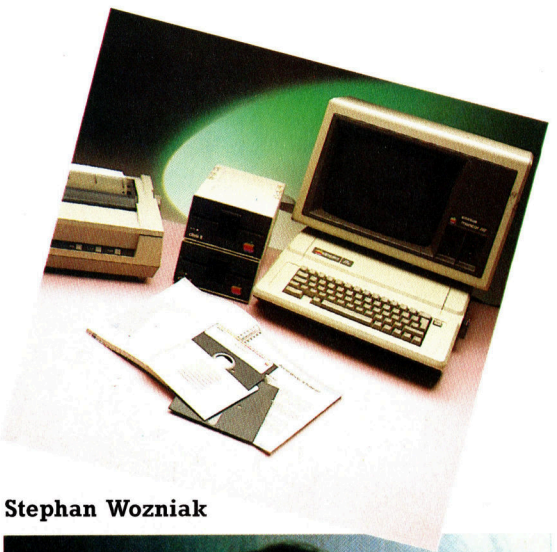
Steven Jobs

ein großer Erfolg im US-Markt. TRS ist die Abkürzung für Tandy Radio Shack, die 80 bezieht sich auf den verwendeten Microprozessor Zilog 80. Zilog war ein weiterer neuer Chip-Hersteller und hatte einen Chip entwickelt, der dem Intel 8080 ähnlich war, doch entscheidende Verbesserungen aufwies.

Mit dem TRS-80 Modell 1, dem Apple II sowie Commodore PET kündigte sich bereits die Hardware-Vielfalt an. Hand in Hand gingen damit die Probleme der Nicht-Kompatibilität und der uneinheitlichen Software. Denn vom Chip hängt ab, welche von Drittanbietern offerierte Software auf dem Rechner funktionsfähig ist.

1972 arbeitete bei Intel ein junger Mann namens Gary Kildall als Berater. Sein Unternehmen, Microprozessor Application Associates, entwickelte eine Computersprache, die es den Ingenieuren ermöglichen sollte, Software-Pakete für die neuen Intel-Microprozessoren zu schreiben.

In einer Garage bauten Kildall und sein Freund John Torode ein eigenes Computersystem. Torode entwickelte die Hardware, um eine Diskettenstation mit dem Prozessor lauffähig zu machen, und Kildall schrieb die Software, die es dem Prozessor ermöglichte, die Floppy zu steuern. Das Programm wurde CP/M benannt (Control Program/Microcomputers). Der Name rührte aus Kildalls Arbeit mit der Intel-Programmiersprache her, die die Bezeichnung PL/M (Programming Language/Microcomputers) trug. Dieses erste Disketten-



Stephan Wozniak

Betriebs-System fand das Interesse aller Hardware-Hersteller, die ihre Rechner mit Diskettenstationen ausstatten wollten. Die Software beeinflußte aber auch die Konstruktion: CP/M konnte nur auf den Prozessoren 8080 und dem schnelleren 8085 von Intel sowie dem Z80 von Zilog laufen. So wurde der Z80 zum Standardchip für jeden CP/M-Rechner und CP/M-Kompatibilität zum großen Ziel aller Softwareanbieter.

Neben dem Betriebssystem waren Programmiersprachen erforderlich, mit denen Anwender eigene Programme schreiben konnten.

Gary Kildall
Neuere Betriebssysteme werden von großen Programmiererteams geschrieben, doch Gary Kildall entwickelte CP/M allein. Selbst in verbesserten Versionen wird deutlich, daß dieses Programm für die frühen einfachen Rechner vorgesehen war.

Adam Osborne
Adam Osborne war lange Jahre einer der führenden Microcomputer-Journalisten, bevor er sein eigenes Unternehmen gründete und den ersten tragbaren Computer der Welt entwickelte.

BASIC, am amerikanischen Dartmouth College entwickelt, bewies sich als leicht erlernbare Sprache und wurde bald Standard.

Bill Gates, der in Seattle promoviert hatte, entwickelte einen BASIC-Interpreter für Microprozessoren, der in Heimcomputer integriert werden konnte. Gates' Unternehmen Microsoft wurde zum maßgebenden Hersteller von Computersprachen.

Auf diese Entwicklungen folgten rasch weitere Fortschritte bei Hardware- und Software-Anwendungen. Dan Bricklin und Bob Frankston entwarfen das erste Micro-Spreadsheet-Programm, VisiCalc, in ihrem Unternehmen Software Arts. Das Programm wurde von Personal Software vertrieben und entwickelte sich zum bestverkauften Software-Paket für den Apple II. Aufgrund dieses Erfolges änderte Personal Software die Firmenbezeichnung in VisiCorp um. WordStar, von Seymour Rubinsteins Haus MicroPro entwickelt, wurde das meistverkaufte CP/M-Textverarbeitungsprogramm.

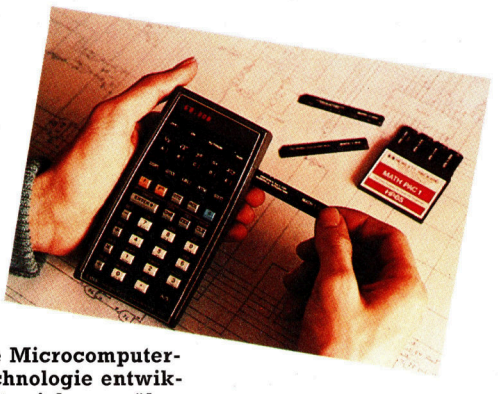
Die Hardware, auf der diese Programme liefen, wurde im Laufe der Zeit leistungsfähiger

Sir Clive Sinclair
Nach seinen zukunftsträchtigen Produkten, wie HiFi, Taschenrechner, Miniatur-Radios und Mini-Fernsehern, brachte ihm der unvergleichliche Erfolg seiner Microcomputer (ZX 80, ZX 81, ZX Spectrum) 1983 den Titel „Sir" ein.

und zugleich billiger. Adam Osborne, der seine Karriere als technischer Autor, Journalist und Software-Verleger begonnen hatte, produzierte in England einen erfolgreichen Bürocomputer, dessen knapp kalkulierter Kaufpreis ein umfangreiches Paket sonst teurer Software beinhaltete. Nicht zu vergessen ist Sir Clive Sinclair, der mit den Rechnern ZX 80, ZX 81 und ZX Spectrum neue Preismaßstäbe setzte und so den Heimcomputer Millionen Einsteigern nahebrachte.

In den vergangenen Jahren setzte IBM mit seinem IBM PC den Standard für Microcomputer. 1982 auf den Markt gebracht, erfreut sich der Rechner steigender Popularität.

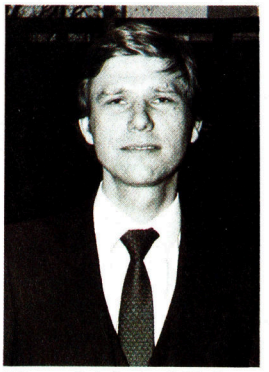
Viele Pioniere aus den Anfangstagen der Microcomputer-Industrie haben zum Erfolg des IBM PC beigetragen. So liefert Intel den Microprozessor, und Betriebssystem und Programmiersprache entstanden in der Zusammenarbeit von Microsoft und Digital Research. Die ersten beiden Software-Pakete für die Maschine waren VisiCalc und WordStar.



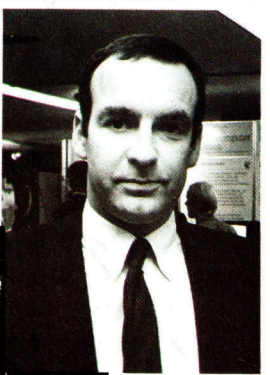
Die Microcomputer-Technologie entwickelte sich, was überraschen mag, aus den programmierbaren Taschenrechnern (wie diesem Hewlett-Packard HP 65).

Auch Chuck Peddle gründete sein eigenes Unternehmen, Sirius, das in England große Erfolge hatte, bevor IBM seinen Rechner auf den Markt brachte. Seitdem hat das Unternehmen jedoch beträchtliche finanzielle Schwierigkeiten. Doch Peddle wird sicher auch künftig eine Rolle spielen. Die kurze Geschichte des Micro-Geschäftes beweist, daß die Pioniere auch die Überlebenden sind, selbst wenn die Multis versuchen, „das Spiel zu machen".

Herman Hauser



Wenngleich preislich weniger käuferfreundlich als Sinclair, ist die Arbeit von Chris Curry und Herman Hauser (als Designer und Entwickler der Acorn Computer) nicht zu vernachlässigen. Der Acorn B und der Electron sind als Meilensteine zu sehen.



Chris Curry

Erst 1982 stieg IBM ins Microcomputer-Geschäft ein. Und das hatte Folgen. Fast jeder Neuling auf dem Microcomputer-Markt legt Wert auf IBM PC-Kompatibilität, um etwas von dem reichhaltigen Softwareangebot zu haben.

Speicheradressen

Im Vergleich zu unserem vertrauten Dezimalsystem sieht das Hexadezimalsystem umständlich und kompliziert aus. In Verbindung mit dem Acht-Bit-System des Rechners zeigt es sich jedoch als äußerst brauchbar und leicht verständlich.

An diesem Punkt unseres Kurses über Maschinensprache lohnt es sich, auf die Darstellung von Zahlen genauer einzugehen. Es wurde bereits festgestellt, daß die gleichen Zahlen in den verschiedenen Systemen mit unterschiedlichen Methoden dargestellt werden. Mathematische Vorgänge haben jedoch immer die gleichen Ergebnisse, unabhängig von Sprache oder System, in dem die einzelnen Bestandteile eines mathematischen Ausdrucks beschrieben werden.

Der Wert der Zahl 5 bleibt gleich, egal ob sie als Fünf oder Five bezeichnet wird. Genauso verändert sie sich auch nicht, wenn sie 5 oder 101b genannt wird. (Das b hinter 101 bedeutet, daß die Zahl als Binärzahl verstanden werden muß.) Der Grund für die Verwendung eines Zahlensystems ergibt sich aus der Rechengewohnheit oder der Brauchbarkeit des Systems für einen definierten Anwendungsbereich.

Wir empfinden das Dezimalsystem als praktisch, weil wir in unserem täglichen Leben ständig damit umgehen. Das Dezimalsystem ist aber längst nicht das einzige Zahlensystem, das wir wie selbstverständlich einsetzen. So haben zum Beispiel Digitaluhren ein kombiniertes mathematisches System: teilweise dezimal, teilweise Modulo 60 (resultierend auf der Basis 60 Minuten bzw. Sekunden) und teilweise Modulo 24 (die 24 Tagesstunden). Ein weiteres Beispiel: Vor 1971 hatte englisches Geld folgende Einheiten: Ein Shilling enthielt zwölf Pence und ein Pfund 20 Shilling. Obwohl die Beherrschung dieses Systems Schwierigkeiten mit sich brachte, versuchte niemand, mit Pence und Shilling in Form von Dezimalbruchteilen des Pfundes zu rechnen.

Die Zahlensysteme

Für Computer ist das Binärsystem von Vorteil, da sich die internen elektrischen Vorgänge (Folgen der Schaltstellungen EIN und AUS = 1 und 0) damit am besten darstellen lassen. Wenn wir es dabei immer nur mit einzelnen Bytes zu tun hätten, wäre das Binärsystem die perfekte Darstellungsmethode. Im Normalfall sind jedoch besonders Speicheradressen und auch andere oft verwandte Zahlen zu groß, um noch in ein Byte zu passen. Nach jahrelangem Arbeiten mit dem Binärsystem entwickelten Programmierer und Systemingenieure daher ein Zahlensystem, das die logischen Vorteile des Binärcodes hatte, aber wie das Dezimalsystem leicht zu überblicken war. Es entstanden zwei Zahlensysteme: das Hexadezimal- und das Oktalsystem. Das Hexadezimalsystem – auch „Hex" genannt – entwickelte sich zum Standard der Microcomputer. Das auf acht aufbauende Oktalsystem wurde meistens auf Großrechnern eingesetzt, wird jetzt aber mehr und mehr von dem Hexadezimalsystem verdrängt.

Die Basiszahl legt die Anzahl der einstelligen Zahlensymbole des Systems fest und ist außerdem der Multiplikationsfaktor für die einzelnen Stellenwerte. So gibt es z. B. im Dezimalsystem zehn einstellige Zahlensymbole (0–9), und der Wert der einzelnen Stellen erhöht sich von rechts nach links jeweils um den Faktor Zehn.

Das Hexadezimalsystem verfügt über 16 einstellige Zahlensymbole: die Zahlen 0 bis 9 und die Buchstaben A bis F. Beim Aufwärtszählen in Hex werden alle einstelligen Zahlensym-

Dezimal	Binär	Hex
0	00000000	0
1	00000001	1
2	00000010	2
3	00000011	3
7	00000111	7
8	00001000	8
9	00001001	9
10	00001010	A
11	00001011	B
12	00001100	C
13	00001101	D
14	00001110	E
15	00001111	F
16	00010000	10
17	00010001	11
24	00011000	18
25	00011001	19
26	00011010	1A
27	00011011	1B
31	00011111	1F
32	00100000	20
33	00100001	21

Diese Tabelle zeigt die Dezimalzahlen 0 bis 33 mit ihren binären und hexadezimalen Äquivalenten.

bole eingesetzt. Dann geht man auf mehrstellige Darstellungen über, bei denen wie im Dezimalsystem der Wert einer Zahl von seiner Position bestimmt wird. Die Hexzahlen nach 9 sind daher A (dezimal 10), B (dezimal 11), C (dezimal 12), D (dezimal 13), E (dezimal 14) und F (dezimal 15). Da jetzt alle einstelligen Zahlensymbole einmal eingesetzt wurden, ist die auf F folgende Hexzahl 10 (gesprochen: „Eins-Null Hex" = dezimal 16). Der Wert einer Stelle im Hexadezimalsystem erhöht sich bei jeder Bewegung nach links jeweils um den Faktor 16. Im Dezimalsystem bezeichnen wir die einzelnen Stellen als Einer, Zehner, Hunderter und Tausender. Im Hexadezimalsystem repräsentieren die entsprechenden Stellen Einer, Sechzehner, Zweihundertsechsundfünfziger und Viertausendsechsundneunziger.

Zahlenumwandlungen

Wenn Sie jetzt die Binärstellen mit den Hexstellen vergleichen, können Sie leicht den Hauptvorteil der Hexzahlen feststellen: Eine vierstellige Binärzahl läßt sich durch eine einstellige Hexzahl darstellen.
Der Bereich einer einfachen Acht-Bit-Zahl (ein Byte), die im Binärsystem aus acht Stellen und im Hexsystem aus zwei Stellen besteht, ist daher:

```
        0 bis 255        dezimal
00000000 bis 11111111 binär
       00 bis FF         hex
```

Für die Umwandlung einer Hexzahl in eine Binärzahl muß jede Stelle der Hexzahl als vierstellige Binärzahl dargestellt werden. Dabei entsprechen die linken vier Stellen der Binärzahl der äußersten linken Ziffernstelle der Hexzahl und die vier rechten binären Bits der rechten Stelle der Hexzahl. Eine derartige Aufteilung eines Bytes ergibt zwei „Nybbles" (ein Nybble ist die Hälfte eines Bytes). Das Nybble auf der linken Seite des Bytes wird „höherwertiges Nybble" genannt, während man das rechte als das niederwertige Nybble bezeichnet. Hier ein Beispiel:

```
 höherwertiges                niederwertiges
      Nybble                       Nybble
         ↓                            ↓
206  =  1100                  1110  =  C E
 ↑                ↑                    ↑
dezimal          binär              hexadezimal
```

Im Hexadezimalsystem lassen sich die aus acht Bits bestehenden Bytes weitaus einfacher berechnen als im Binärsystem. Eine Vertrautheit mit dem System wird sich schnell einstellen, wenn Sie Ihre ersten Erfahrungen mit Hexzahlen, Speicheradressen und Speicherinhalten machen.

In dieser Folge haben wir Programme für den Acorn B, den Commodore 64 und den Spectrum abgedruckt, mit denen sich Byteinhalte des Speichers darstellen lassen. Diese Programme fragen zuerst nach der Anfangsadresse (d. h. der Nummer des ersten Bytes) und dann nach der Anzahl der Bytes, die Sie sich ansehen wollen. Wenn Sie z. B. Byte 1953 als Anfangspunkt angegeben haben und die Inhalte der nächsten vier Bytes abfragen wollen, zeigt die linke Spalte die Dezimalzahl 1953, während der Inhalt von Byte 1953, Byte 1954, Byte 1955 und Byte 1956 in den vier rechten Spalten erscheint.

Wenn die Maschine anzeigt, daß das Byte 1956 die Dezimalzahl 175 enthält, sollten wir uns vor Augen führen, was das eigentlich bedeutet: In einem der Speicherchips existiert ein Bereich, den die Maschine als Byte 1956 bezeichnet und der acht elektrische Zustände enthält. Wenn wir 0 Volt als 0 darstellen und 5 Volt als 1, dann enthält Byte 1956 das Voltmuster 1010111. Wir entscheiden uns dafür, dieses Muster als Binärzahl zu interpretieren, deren dezimales Äquivalent 175 beträgt.

Der Bildschirm kann den „echten" Inhalt eines Bytes nicht anzeigen. Wir sehen nur Zeichen, die wir den Voltmustern der einzelnen Bytes zugeordnet haben: Wir interpretieren die Voltmuster als Binärzahlen, die der Rechner wiederum in Dezimalzahlen darstellt. Nun gehen wir einen Schritt weiter und wandeln die Dezimalzahlen in ASCII-Zeichen (American Standard Code for Information Interchange) um. Diese Zeichen werden in der Spalte am rechten Bildschirmrand dargestellt.

ASCII-Codierung

ASCII ist ein international anerkannter Code, mit dem die meisten Computer arbeiten und der allen Zeichen der Tastatur (ursprünglich der des Teleprinters) Dezimalzahlen zwischen 0 und 127 zuordnet. Dabei entspricht die Zahl 65 dem großen „A", 66 bedeutet „B", 67 „C" etc. Bei den nicht im Alphabet enthaltenen Zeichen entspricht 32 dem Leerzeichen, 42 dem Stern, 13 der Return-Taste usw.

Drucken lassen sich die ASCII-Zeichen zwischen 32 und 127, alle anderen Codes sind entweder nicht fest definiert, Funktionszeichen oder nur auf bestimmten Maschinen vorgesehen. In den Demonstrations-Programmen erscheint auf dem Bildschirm daher für jedes Byte, dessen Codezahl außerhalb des druckbaren Bereiches liegt, ein Stern. In der nächsten Folge werden wir den vollständigen ASCII-Zeichensatz (0 bis 127) abdrucken.

Für das Verständnis des Maschinencodes stellt der ASCII-Zeichensatz eine wichtige Hintergrundinformation dar: Zunächst wird damit

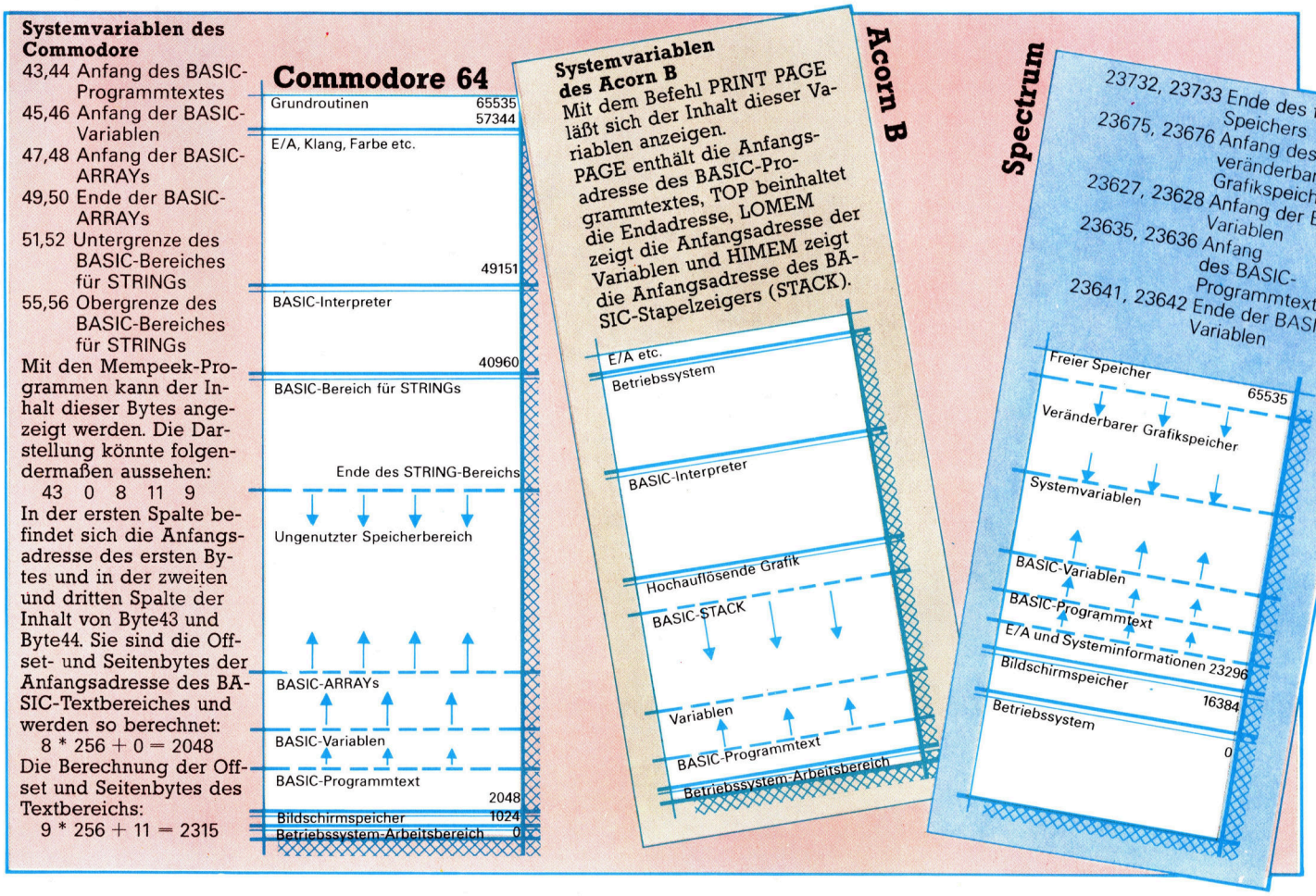
klar, daß der Inhalt eines Speichers unterschiedlich interpretiert werden kann. Sie können den Inhalt eines Bytes als Zahl behandeln, als Zeichen eines Codes oder als Befehl. Weiterhin gewinnen Sie dadurch eine klarere Vorstellung von den Speicherbereichen, die Zeichen enthalten können und vom Betriebssystem der Maschine oder von Ihnen genutzt werden.

Alle Meldungen des Betriebssystems – z. B. Fehlermeldungen, READY, oder START TAPE THEN PRESS RETURN, kurz alles was auf dem Bildschirm erscheint – müssen in ASCII codiert und im Speicher untergebracht sein. Wir erhalten damit einen Eindruck von den Grenzen einer „intelligenten" Maschine. Unsere Intelligenz arbeitet anders: Wir brauchen nicht alle verfügbaren Meldungen zu speichern, wir fassen einen Gedanken und formulieren dann die Worte, die diesen Gedanken ausdrücken.

Memory Maps

Eine Memory Map (Speicheratlas) zeigt in Diagrammform, in welche Bereiche der Speicher aufgeteilt ist und welche Adressen die einzelnen Bereichsgrenzen haben. Einige Speicherbereiche führen immer die gleichen Aufgaben aus. Zum Beispiel werden auf dem Commodore 64 die Bytes 0 bis 1024 von den Variablen des Betriebssystems belegt. Andere Bereiche übernehmen unterschiedliche Aufgaben – abhängig von der Größe und Art des Programms. Die Grenzen sind entweder festgelegt (in unseren Diagrammen als durchgehende Linien angezeigt) oder fließend (gebrochene Linien). Feste Grenzen verändern sich nicht, während die Größe der Bereiche mit fließenden Grenzen je nach Bedarf variiert. In der Memory Map des Commodore 64 sind die Grenzen des Bildschirmspeichers auf Byte 1024 bis 2048 festgelegt, während der Bereich für BASIC-Variablen von der Anzahl der eingesetzten Variablen bestimmt wird.

Mit den Mempeek-Programmen lassen sich auch die aktuellen Grenzen der variablen Speicherbereiche anzeigen. Der Commodore besitzt sechs Zeiger (Pointer) für die Systemvariablen. Wir haben diese Variablen in unseren Beispielen unten aufgeführt und können aus den Bytepaaren die einzelnen Speicheradressen erfahren. Das BBC-BASIC hat übrigens vier Systemvariablen und der Spectrum fünf.



Acorn B

```
7 REM*****************************
8 REM*     BBC       MEMPEEK 1    *
9 REM*****************************
20 MODE 7
30 *TV 255
40 CLS
50 REPEAT
100 INPUT"START ADDRESS ",SA
200 INPUT"NUMBER OF BYTES (0 TO QUIT)",B
N
250 PRINT "*****************************
**********"
300 FOR B%=SA TO (SA+BN-1) STEP 4
350 H$="":@%=6
400 PRINT TAB(0);B%;TAB(8);
450 @%=4
500 FOR C=0 TO 3
550 PK%=?(B%+C):PK$="."
600 PRINT PK%;
650 IF PK%=13 THEN PK$=CHR$(124)
700 IF (PK%>31) AND (PK%<128) THEN PK$=C
HR$(PK%)
750 H$=H$+PK$
800 NEXT C
850 PRINT TAB(32);H$
900 NEXT B%
950 UNTIL BN=0
1000 REM*************************
```

Spectrum

```
7 REM*************************
8 REM*  SPECTRUM  MEMPEEK 1   *
9 REM*************************
30 DIM H$(4)
50 FOR L=0 TO 1 STEP 0
100 INPUT"START ADDRESS ";SA
200 INPUT"NUMBER OF BYTES (0 TO
QUIT)";BN
250 PRINT "*********************
*********"
300 FOR B=SA TO (SA+BN-1) STEP 4
350 LET H$="...."
400 PRINT B;TAB 7;
500 FOR C=0 TO 3
550 LET PK=PEEK(B+C)
600 PRINT PK;" ";
650 IF (PK>31) AND (PK<128) THEN
 LET H$(C+1)=CHR$ PK
700 IF PK=13 THEN LET H$(C+1)="■
"
800 NEXT C
850 PRINT TAB 26;H$
900 NEXT B
950 IF BN=0 THEN LET L=2
1000 NEXT L
1050 REM*************************
```

Commodore 64

```
7 REM*****************************
8 REM*    COMMODORE MEMPEEK 1    *
9 REM*****************************
30 PRINT CHR$(147)     :REM CLEAR SCREEN
40 PRINT CHR$(142)     :REM UPPER CASE
50 FOR LP=0 TO 1 STEP 0
100 INPUT"START ADDRESS ";SA
200 INPUT"NUMBER OF BYTES (0 TO QUIT)";B
N
250 PRINT "*****************************
**********"
300 FOR B=SA TO (SA+BN-1) STEP 4
350 H$=""
400 PRINT B;TAB(8);
500 FOR C=0 TO 3
550 PK=PEEK(B+C):PK$="."
600 PRINT TAB(8+5*C);PK;
650 IF PK=0 THEN PK$=CHR$(122)
700 IF (PK>31) AND (PK<128) THEN PK$=CHR
$(PK)
750 H$=H$+PK$
800 NEXT C
850 PRINT TAB(32);H$
900 NEXT B
950 IF BN=0 THEN LP=1
1000 NEXT LP
1050 REM*************************
```

Wie funktionieren die „Mempeek"-Programme?

Bevor Sie die Mempeek-Programme ausprobieren, sollten Sie diese mit SAVE speichern, da Schreibfehler zu einem Abbruch des gesamten Systems führen können.

Das Programm fragt zunächst nach der Anfangsadresse und dann nach der Anzahl der Bytes, die Sie sich ansehen wollen. Beide Eingaben sollten ganze Zahlen zwischen 0 und 65535 sein. Die Eingabe von 0 bei der Byteanzahl beendet das Programm. Wenn Sie z. B. die Anfangsadresse 230 eingeben, kann der Bildschirm folgendes anzeigen:

```
ANFANGSADRESSE? 230
ANZAHL BYTES (0 = ENDE) ? 8

230 193 32 65 49 . A1
234 129 64 93 98 . @]b
```

Die linke Spalte zeigt die Dezimaladresse des ersten Bytes an, die nächsten vier Spalten bringen den Inhalt der vier Bytes von dieser Adresse an als Dezimalzahlen auf den Schirm, während die rechte Spalte das dem Byteinhalt entsprechende Zeichen (wenn möglich) darstellt. Läßt sich das Zeichen nicht darstellen, wird ein „." angezeigt.

Beginnen Sie am besten damit, im Speicher ein wenig „herumzuwandern", sehen Sie sich die interessanten Adressen an und versuchen Sie, die Meldungen des Betriebssystems und die Schlüsselworte des BASIC zu finden. Das Benutzerhandbuch kann Ihnen dabei helfen.

Wenn Sie die Zeiger gefunden haben, die die Grenzen der Speicherbereiche angeben, dann können Sie in das Programm ein paar REM-Zeilen einfügen und die Auswirkungen auf die Zeigerwerte beobachten. Fügen Sie auch am Anfang des Programms einige Zeilen für die STRING-Bearbeitung ein und untersuchen Sie, wie sich dies auf die Zeiger und den Inhalt des Speicherbereichs für Variablen auswirkt.

Fachwörter von A bis Z

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.

BASIC = BASIC

Dieser Begriff ist jedem Computer-Besitzer vertraut, und die meisten wissen auch, daß dies ein Akronym von „Beginners All-purpose Symbolic Instruction Code" (Anfänger-Allzweck-Symbolsprache) darstellt. Weniger geläufig ist, daß BASIC am Dartmouth College (USA) als Sprache für den Informatikunterricht entstanden ist.

BASIC wurde von FORTRAN abgeleitet, der damals in Wissenschaft und Technik verbreitetsten Sprache. Dabei wurden umständliche Anweisungen wie zum Beispiel WRITE und FORMAT durch den einfachen PRINT-Befehl ersetzt.

Der eigentliche Fortschritt besteht aber darin, daß BASIC schon vom Konzept her eine Dialogsprache ist: Alle Eingaben erfolgen direkt durch den Benutzer am Terminal und nicht, wie früher üblich, durch Einlesen vorgestanzter Lochkarten. Programmfehler sind daher im Dialog durch Überschreiben einzelner Zeilen leicht zu korrigieren.

BCD = BCD

Beim BCD-Code (Binary Coded Decimal) wird eine Dezimalzahl Stelle für Stelle binär verschlüsselt, z. B. zum Abspeichern auf Diskette. Die meisten Heimcomputer arbeiten zur besseren Speichernutzung allerdings mit Fließkommadarstellung (Floating Point). Dabei wird die Dezimalzahl als Ganzes in eine Binärzahl umgewandelt und anschließend mit dem Dezimalpunkt versehen. Der Punkt wird dabei in der „Mantisse" ganz nach links geschoben und der Stellenwert in einen Exponent übernommen. Beide werden in einer festen Anzahl von Bytes abgespeichert.

Beim BCD-Code wird jede Ziffer der ursprünglichen Dezimalzahl in eine vierstellige Dualzahl (d. h. 1/2 Byte) umgesetzt, so daß insgesamt halb so viel Bytes wie dezimale Stellen erforderlich sind. Das BCD-System behandelt jede Ziffernstelle einzeln. Dagegen wird beim Fließkommaverfahren die Zahl als Ganzes verarbeitet.

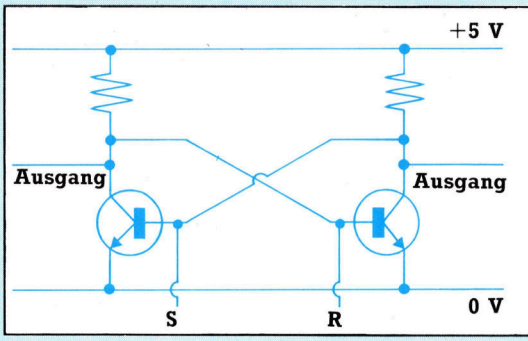
Benchmark = Bewertungsstandard

Als die ersten Personal Computer wie der PET, der Apple II und der TRS-80 von Tandy auf den Markt kamen, wurde ein Bewertungssystem entwickelt, um die Verarbeitungsgeschwindigkeit und die Effizienz der BASIC-Interpreter zu vergleichen. Es bestand aus zehn einfachen Programmen, die unterschiedliche Aspekte berücksichtigten (Schleifenbearbeitung, Fließkommarechnung, Winkelfunktionen usw.). Damit erzielte Testzeiten finden Sie in technisch orientierten Computerzeitschriften, etwa in der Form „BM1–10,2 s; BM2–3,87 s" und so fort.

Es ist nicht gelungen, ein entsprechendes System für moderne Bürorechner einzuführen – hauptsächlich deshalb, weil die Effizienz solcher Anlagen entscheidend vom speziellen Zuschnitt der Programme auf den jeweiligen Rechner abhängt. Der 8-Bit-Rechner Osborne 1 z. B. gilt nicht gerade als flink, trotzdem wird er von vielen Journalisten für die Textverarbeitung bevorzugt, weil er mit dem WordStar schneller arbeitet als die meisten neuen 16-Bit-Rechner.

Bistable = Bistabile Kippstufe

Als „bistabil" bezeichnet man die sog. „Flip-Flop"-Schaltungen mit ihren zwei stabilen Zuständen. Am einfachsten ist das RS-Flip-Flop realisierbar. Der Schaltzustand des Flip-Flops wird durch die Ausgangsspannung angezeigt, die z. B. 5 V („High"-Pegel) oder 0 V („Low"-Pegel) betragen kann.



Ein RS-Flip-Flop ist eine bistabile Kippschaltung, die als „Gedächtnis" für ein einzelnes Bit dient. Mit Kombinationen von solchen Flip-Flops können Binärzahlen gespeichert werden.

Das RS-Flip-Flop eignet sich als 1-Bit-Speicher, der je nach Schaltzustand eine „1" oder eine „0" enthält. Die ersten Halbleiterspeicher bestanden aus Serien von solchen Flip-Flops, und die heute noch üblichen statischen RAMs sind dasselbe in kompakter Bauweise. Der Trend geht allerdings zu dynamischen RAMs, bei denen die Information in Form elektrischer Ladungen auf winzigen Kondensatoren gespeichert wird. Wegen des Spannungsabfalls benötigen dynamische RAMs spezielle (in die Bausteine integrierte) Auffrisch-Schaltungen. Sie sind aber schneller und brauchen weniger Strom als die statischen RAMs.

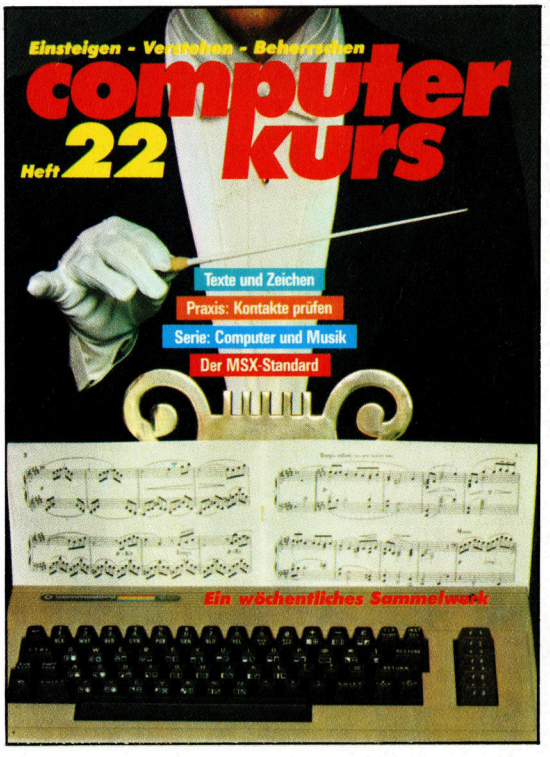
Flip-Flops sind weiterhin als logische Einzelschaltkreise zu finden. Es gibt unterschiedliche Ausführungsformen. Das oben abgebildete RS-Flip-Flop hat zwei Eingänge: Ein Impuls am S-Eingang „kippt" das Flip-Flop in den einen Zustand, während es durch einen Impuls am R-Eingang „zurückgekippt" wird.

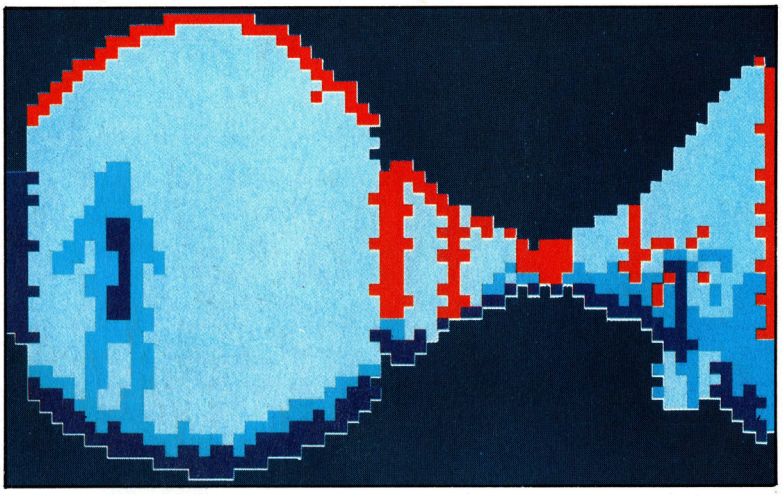
Bildnachweise

561, 563, 564, 565, 573, 575: Ian McKinnell
566, 577: Kevin Jones
574: Chris Stevens
579: Tony Lodge
581, U3: Liz Dixon
582: Commodore, Tony Sleep
583: Apple, Osborne
584: Sinclair Research, Hewlett Packard, Judy Goldhill

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

computer kurs Heft 22



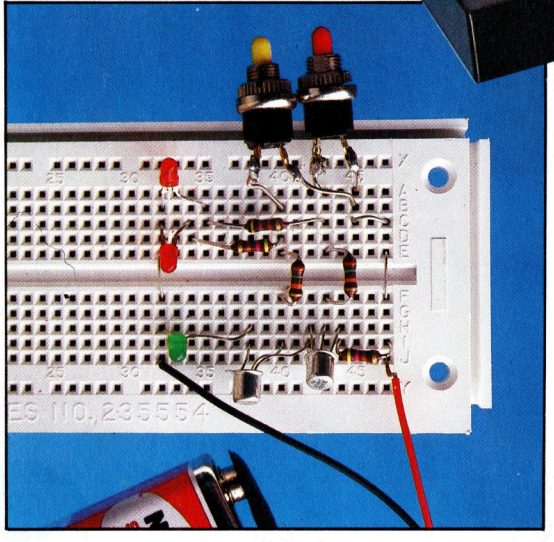


Infrarotbild

Ein Roboter kann auf diese Weise Lebewesen erkennen: links der Hitzeschatten eines Menschen, rechts eine Pflanze.

Kleine Reparaturen

Nach dem Aus- und Einbau elektronischer Teile werden alle Lötstellen noch einmal zur Sicherheit geprüft.



Neuer Standard

Mehr als ein Dutzend japanischer Elektronik-Hersteller hat sich auf die MSX-Norm für Computer geeinigt – Kompatibilität durch einheitliche CPUs und Schnittstellen.

+++ Musik aus dem Rechner +++ Lagerhaltungsprogramme +++ Space Invaders +++ Der Microdrive von Sinclair +++ ASCII-Code für Funktionen +++ Zoltoths Schrein +++ BASIC-Adreßbuch +++