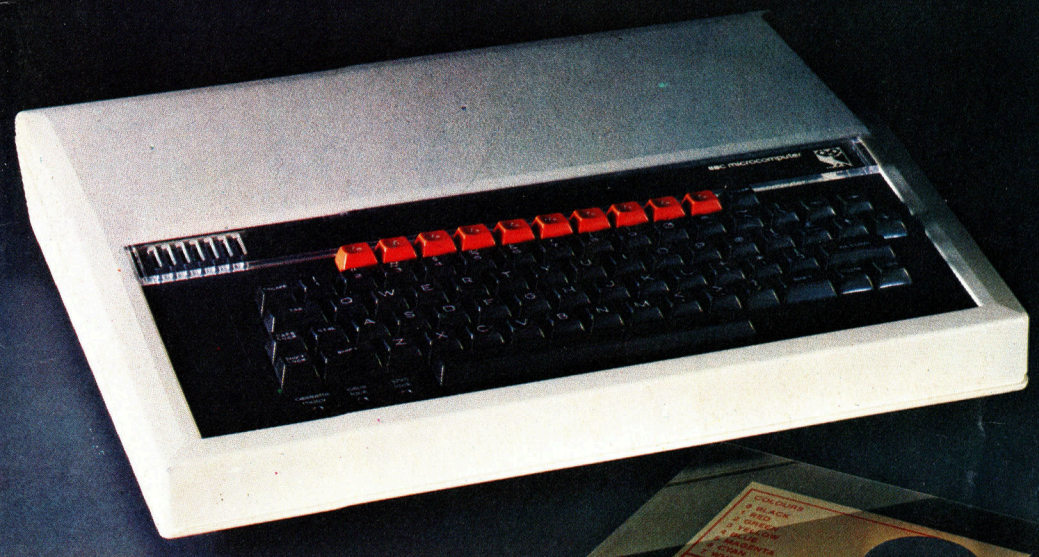
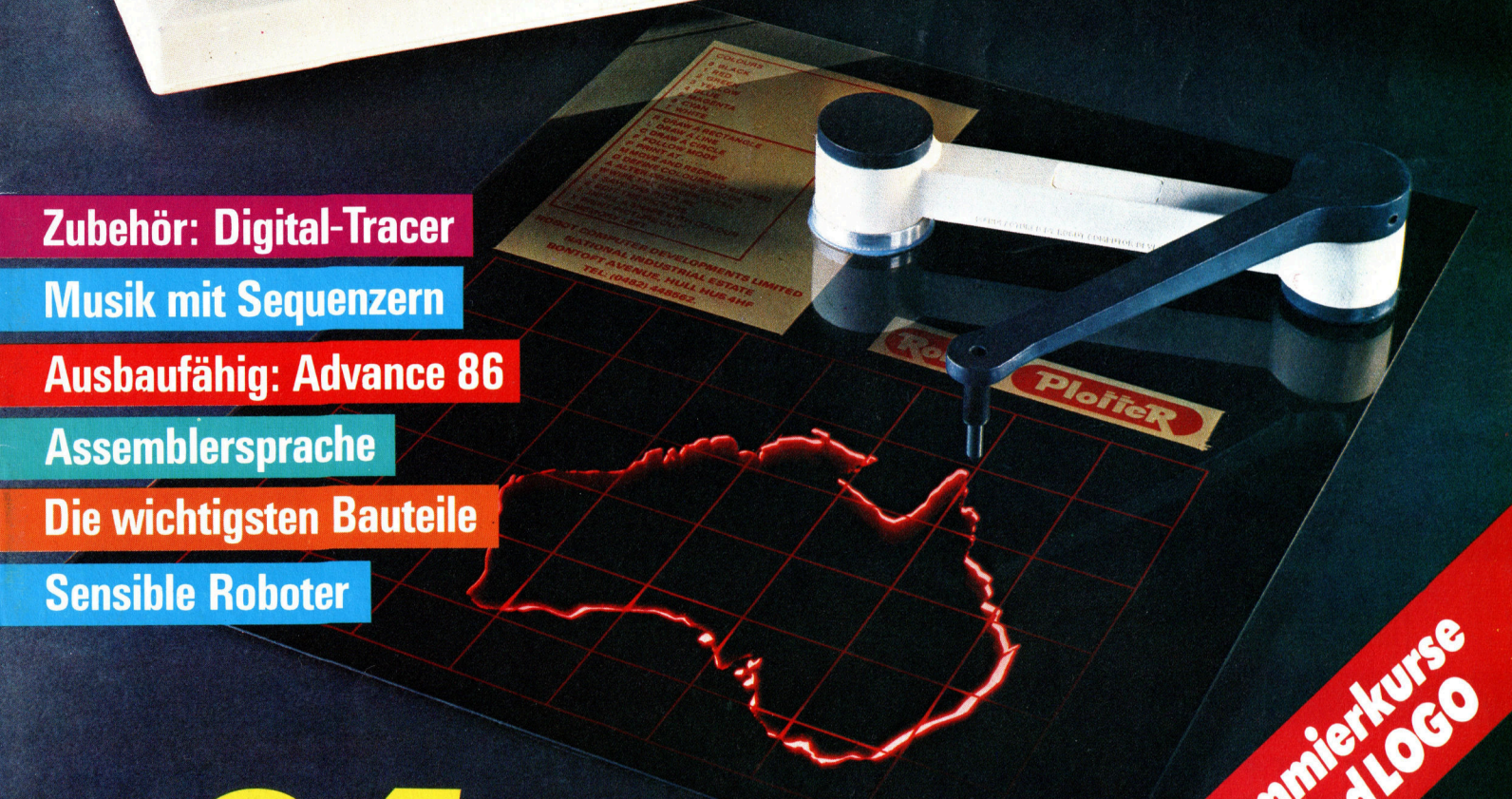
Einsteigen - Verstehen - Beherrschen

DM 3,80 öS 30 sfr 3,80

# computer kurs

Ein wöchentliches Sammelwerk






Zubehör: Digital-Tracer

Musik mit Sequenzern

Ausbaufähig: Advance 86

Assemblersprache

Die wichtigsten Bauteile

Sensible Roboter

Programmierkurse BASIC und LOGO

Heft 24

# computer kurs

## Heft 24

## 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

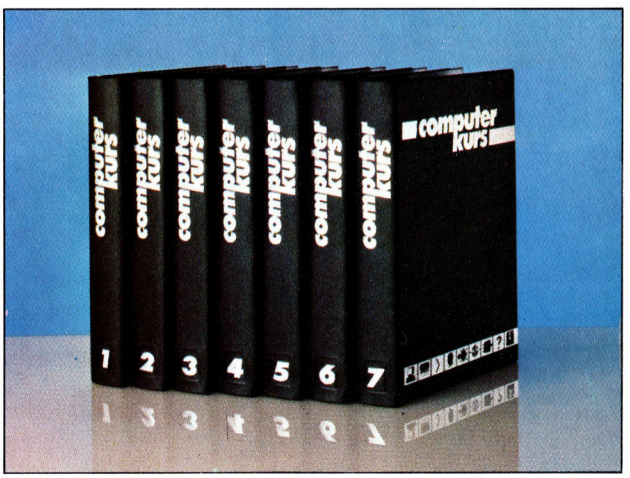
**Schweiz:** Der Sammelordner kostet sfr 15. Bitte wenden Sie sich an Ihren Kiosk; dort werden Sie jederzeit die gewünschten Exemplare erhalten.

## 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.

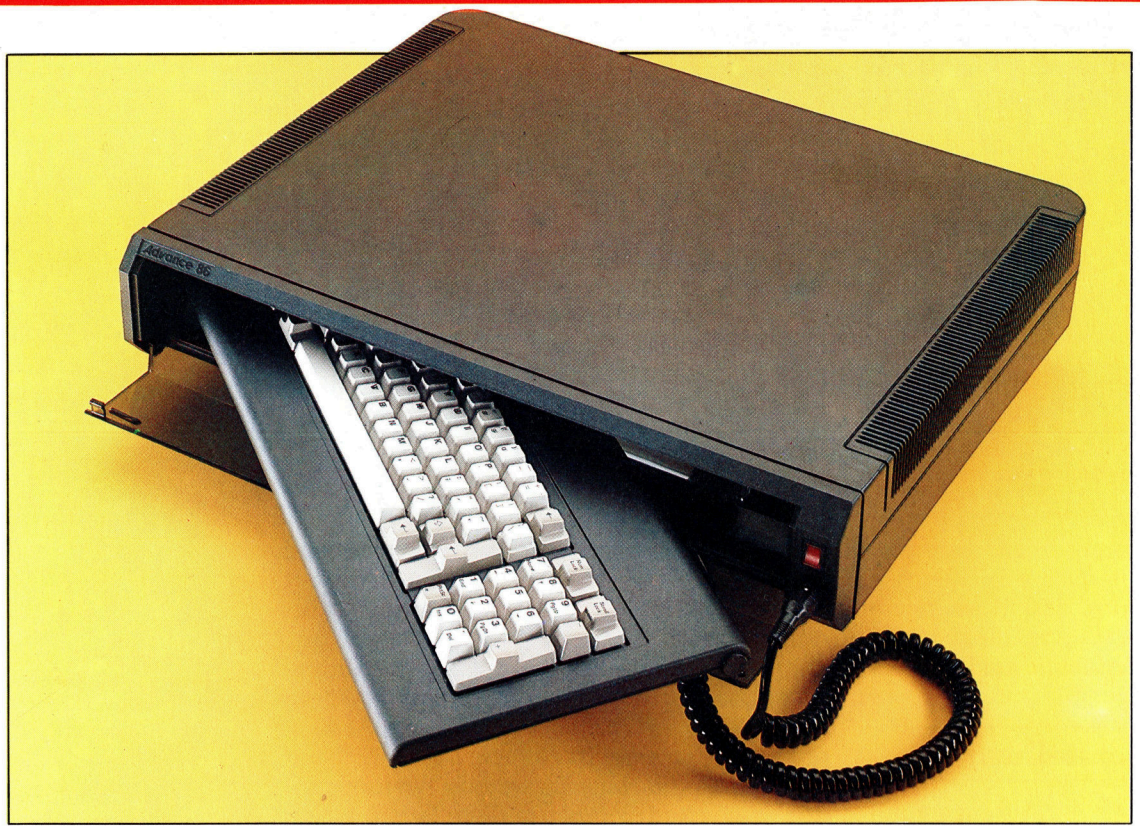

**Redaktion:** Winfried Schmidt (verantw. f. d. Inhalt), Joachim Seidel, Elke Leibinger, Susanne Brandt, Uta Brandl (Layout), Sammelwerk Redaktions-Service GmbH, Paulstraße 3, 2000 Hamburg 1

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




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




**Den Advance 86 gibt es in zwei Versionen – den als Heimcomputer gedachten Advance 86a für ca. 1200 Mark (siehe Bild) und den Advance 86b für ca. 6000 Mark, ein IBM-kompatibler kommerziell einsetzbarer Microcomputer.**

# Erfüllte Erwartung

**Seit Jahren werden billige Heimcomputer mit dem Versprechen angeboten, daß sie sich zu leistungsfähigen kommerziellen Computern ausbauen lassen. Die Firma Advance Technology ist eine der ersten, die dieses Versprechen mit dem Micro Advance 86 erfüllt.**

Den Advance 86 gibt es in zwei Ausführungen: Der 86a arbeitet mit Cassetten und besitzt einen Arbeitsspeicher von 128 KByte RAM. Der 86b hat die gleichen Eigenschaften, verfügt jedoch zusätzlich über zwei Laufwerke für 5 1/4-Zoll-Disketten und ein umfangreicheres BASIC. Der 86a läßt sich zum 86b aufrüsten, wobei das „Erweiterungspaket" mit den Diskettenlaufwerken einfach auf das Grundgerät aufgesteckt wird. Das Ergebnis ist ein IBM-kompatibler Computer für die Hälfte des Preises vom IBM PC.

Der 86b besteht aus zwei Teilen – der Tastatur und dem Hauptgehäuse mit dem Microprozessor. Alle Steckleisten und das Netzteil befinden sich in dem Gehäuse.

Die Tastatur ist per Koaxialkabel und einer fünfpoligen DIN-Buchse mit dem Hauptgerät verbunden. Über der Buchse liegt der Hauptschalter mit einer LED-Betriebsanzeige. Die 84 Tasten der ausgezeichneten Tastatur sind in drei Gruppen angeordnet: die Haupttastatur mit den alphanumerischen Zeichen, ein Block mit zehn Funktionstasten und ein Zehnerblock, der wahlweise für die Zahleneingabe oder die Cursorsteuerung einsetzbar ist.

Die Funktionstasten sind mit einigen Befehlen wie RUN, LIST, SAVE, LOAD etc. belegt, lassen sich aber leicht umdefinieren. Die Zehnertastatur läßt sich mit der Taste „Num Lock" zwischen Rechenfunktion und Cursorsteuerung umschalten.

## 16-Bit-Prozessor Intel 8086

Im Hauptgehäuse befindet sich eine verhältnismäßig kleine Schaltplatine mit 128 KByte RAM und dem 16-Bit-Prozessor Intel 8086. Der 8086 ist mit dem 8088 des IBM kompatibel, arbeitet aber schneller. Ebenfalls auf der Platine befinden sich Steckleisten, mit denen sich die RAM-Kapazität verdoppeln läßt. Obwohl der Arbeitsbereich des BASIC auf maximal 62 KByte begrenzt ist, reicht diese Kapazität für fast alle Anwendungen völlig aus.

Der Advance ist mit zahlreichen Steckkontakten und Schnittstellen ausgerüstet: ein Netzstecker, mit dem ein Monitor oder Fernseher über das Netzteil des Computers betrieben werden kann, ein Signalausgang für den Einsatz eines Fernsehers, RGB-Monitorausgang, eine Standard-Centronics-Schnittstelle für

**Intelligente Tastatur**
Die hochwertige Tastatur des IBM PC diente bei der Konstruktion des Advance als Vorbild, wobei sich jedoch die Tastenbelegungen der beiden Maschinen unterscheiden.

einen parallelen Druckeranschluß, zwei Joystickbuchsen und eine fünfpolige DIN-Buchse für den Cassettenrecorder. Die Erweiterung des Gerätes stellt außerdem eine RS232-Steckleiste für den seriellen Anschluß von Druckern oder Modems zur Verfügung.

## Bildschirmdarstellung

In der Textdarstellung zeigt der Advance wahlweise 25 Zeilen mit je 40 Zeichen oder 25 Zeilen mit je 80 Zeichen an, die jedoch ohne Monitor kaum lesbar sind. Die unterste Bildschirmzeile stellt normalerweise die Befehle der Funktionstasten dar, die Anzeige läßt sich jedoch für eine volle Ausnutzung des Bildschirms abschalten. In dieser Darstellungsart können 16 Farben (blinkend oder normal) auf den Schirm gebracht werden. Die mittlere Auflösung unterstützt den Einsatz von vier Farben und kann 320x200 Pixel oder 25 Zeilen x 40 Zeichen Text darstellen. In der hohen Auflösung ist eine Schwarzweißdarstellung mit 640x200 Pixeln oder 25 Zeilen x 80 Zeichen Text möglich. Im RAM können sieben Bildschirminhalte im 25x40-Format gespeichert und unmittelbar abgerufen werden. Bei der 80-Zeichen-Darstellung lassen sich vier Bildschirmseiten auf diese Weise bereithalten. Der Einsatz der Farben ist jedoch mit einigen Einschränkungen verbunden. In der Textdarstellung kann man für den Hintergrund nur eine von acht Farben benutzen, obwohl für Vordergrund und Umrahmung die gesamte Farbskala zur Verfügung steht. Noch komplizierter ist die mittlere Auflösung: Obwohl vier Farben dargestellt werden können, lassen sich dafür nicht alle Farben einsetzen, sondern nur eine von zwei Farbgruppen (sogenannte „Paletten").

Auf dem 86a gibt es nur zwei Grafikbefehle: PSET (mit dem sich die Farbe eines einzelnen Pixels bestimmen läßt) und LINE (ein Befehl zum schnellen Zeichnen von Linien und Kästen). Befehle wie CIRCLE, PAINT (für das Ausfüllen von gezeichneten Flächen), DRAW (mit dem sich Figuren definieren und zeichnen lassen), GET und PUT (mit denen Teilbereiche des Bildschirms in ein Datenfeld übertragen und dann in anderen Farben und Größen erneut dargestellt werden können) gibt es nur auf dem 86b mit der Diskettenversion des BASIC. Das ist als Nachteil zu werten, da die Besitzer von Heimcomputern erfahrungsgemäß großes Interesse an einem vollständigen Befehlssatz für Grafik haben. Der Zeichensatz des Advance enthält den normalen ASCII-Umfang, mathematische Zeichen, Blockgrafik mit Spielkartensymbolen, Musiknoten und griechische Schriftzeichen. Es besteht außerdem die Möglichkeit, selbst Zeichen zu definieren.

Das BASIC des Advance ist schnell und läßt sich leicht einsetzen. Es enthält Hilfsroutinen für die automatische Vorgabe von Zeilennummern und deren Reorganisation, PRINT USING zur Formatierung der Bildschirmausgabe und den SWAP-Befehl, mit dem der Inhalt zweier Variablen gegeneinander ausgetauscht werden kann. Die Soundmöglichkeiten sind gut, aber nicht außergewöhnlich. Der vollständige Befehlssatz ist auch hier nur bei der Diskettenversion des BASIC vorhanden. Die Ansteuerung der Cassette ist einfach: Das BASIC und Programme im Maschinencode werden mit dem einfachen Befehl LOAD geladen, wobei die Programme automatisch ablaufen, wenn nach dem LOAD ein „R" eingegeben wird.

Beeindruckend ist der ausgezeichnete Bildschirmeditor des Advance. Nach Umschalten der Zehnertastatur mit der Num-Lock-Taste kann der Cursor für Korrekturen oder Eingaben frei auf jede Position des Bildschirms bewegt werden.

Insgesamt scheint der Advance die Versprechung zu erfüllen, daß sich ein Heimcomputer auf ein kommerziell einsetzbares Gerät aufrüsten läßt. Wenn auch das Disketten-BASIC weitaus mehr Möglichkeiten bietet als das BASIC der kleineren Maschine, so besteht auch dieses Gerät den Vergleich mit Heimcomputern gleicher Größe.

**Centronics-Druckerschnittstelle**

**RGB-Monitorausgang**
Hier kann ein RGB-Farbmonitor angeschlossen werden.

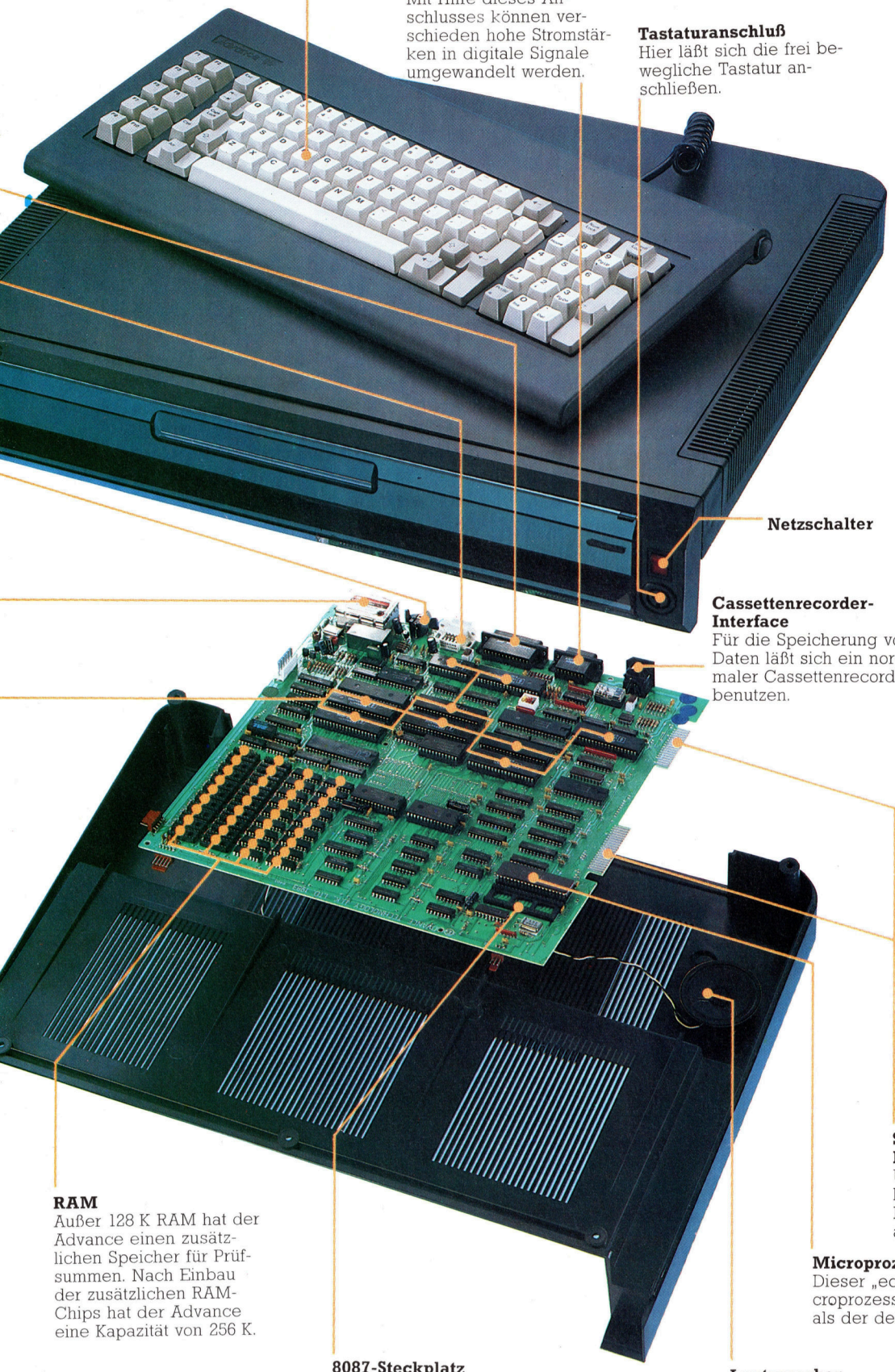
**Videoanschluß**
Signal für Monochrom- oder Farbmonitor.

**Moduliertes Fernsehsignal**
Anschlußmöglichkeit für einen normalen Fernseher.

**Ferranti-ULA-Chips**
Der Preis dieses Computers wird durch den Einsatz von neun ULA- (Uncommitted Logic Array) Chips niedrig gehalten, in denen viele Schaltungen zusammengefaßt sind.

## Angriff auf IBM

Hauptattraktion des Advance ist die Möglichkeit, das Gerät als Heimcomputer kaufen und später zu einem vollwertigen kommerziell einsetzbaren Gerät ausbauen zu können.
Das erweiterte System ist weitestgehend mit dem IBM kompatibel. Unser Bild zeigt den Flugsimulator von Microsoft auf dem Advance. Die Erweiterung kostet ca. 4400 Mark und läßt sich auf das Gehäuse des 86a aufsetzen. Der Ausbau sollte von einem Händler vorgenommen werden.



**IBM-ähnliche Tastatur**

**Buchse für Joysticks und Analog/Digital-Wandler**
Mit Hilfe dieses Anschlusses können verschieden hohe Stromstärken in digitale Signale umgewandelt werden.

**Tastaturanschluß**
Hier läßt sich die frei bewegliche Tastatur anschließen.

**Netzschalter**

**Cassettenrecorder-Interface**
Für die Speicherung von Daten läßt sich ein normaler Cassettenrecorder benutzen.

**RAM**
Außer 128 K RAM hat der Advance einen zusätzlichen Speicher für Prüfsummen. Nach Einbau der zusätzlichen RAM-Chips hat der Advance eine Kapazität von 256 K.

**8087-Steckplatz**
Hier kann ein Prozessor für komplizierte mathematische Berechnungen eingesetzt werden.

**Lautsprecher**

**Microprozessor 8086**
Dieser „echte" 16-Bit-Microprozessor ist schneller als der des IBM PC.

**Steckleisten für Erweiterungen**
Über diese Steckleisten läßt sich der 86a zu dem IBM-kompatiblen 86b ausbauen.

## Advance 86a

**PREIS**
Advance 86a ca. 1200 Mark
Advance 86b ca. 6000 Mark

**ABMESSUNGEN**
95x400x520 mm

**ZENTRALEINHEIT**
Intel 8086, 4,77 MHz

**SPEICHERKAPAZITÄT**
128 K RAM, auf 256 K erweiterbar, 64 K ROM

**BILDSCHIRM-DARSTELLUNG**
25 Zeilen mit je 40 Zeichen oder 25 Zeilen mit je 80 Zeichen, Grafik 320x200 (4 Farben) oder Grafik 640x200 (schwarz/weiß). 16 Farben in der Textdarstellung.

**SCHNITTSTELLEN**
RGB- und Videoausgang, 2 Joysticks, Centronics-Druckerschnittstelle, Cassettenrecorderanschluß, Netzausgang, RS232-Ausgang (nur 86b)

**PROGRAMMIERSPRACHEN**
BASIC im ROM (86a), BASIC auf Diskette (86b)

**TASTATUR**
84 Schreibmaschinentasten, darunter 10 Funktionstasten und Zehnerblock

**HANDBÜCHER**
Benutzerhandbuch und BASIC-Handbuch werden mitgeliefert.

**STÄRKEN**
Ausgezeichnete Tastatur und Bildschirmeditor. Die Diskettenversion des BASIC ist umfassend.

**SCHWÄCHEN**
In der ROM-Version des BASIC fehlen Befehle, mit denen sich Grafik- und Tonerzeugung voll ausschöpfen lassen.

# Klangfolgen

**Im ersten Teil dieser Serie wurde verdeutlicht, welche entscheidende Rolle die Sequenzer für den Musiker von heute spielen. Exakte Steuerung ist nicht nur auf der Bühne wichtig, sondern im Studio von ebensolcher Bedeutung, wo exaktes Timing Grundvoraussetzung für erfolgreiches Mischen ist.**

Sowohl auf der Bühne wie im Studio hat der Sequenzer das Musikschaffen wesentlich beeinflußt. Doch auch hier gibt es Nachteile: Arbeitet ein Keyboard-Spieler mit zwei Synthesizern, von denen der eine über vielfältige Steuerungsmöglichkeiten verfügt, der andere aber einen guten Klang hat, so gibt es keine Möglichkeit, die beiden Einheiten miteinander bei der Bedienung zu koordinieren. Der Zweck einer digitalen Schnittstelle für Musikinstrumente besteht darin, diese Verbindung herzustellen und das System insgesamt über eine leistungsfähige Steuereinheit regulieren zu können. MIDI ist ein Versuch, dies zu standardisieren. Das heißt: Bei der Musikproduktion kann jedes digitale System ein anderes kontrollieren bzw. beeinflussen oder steuern.

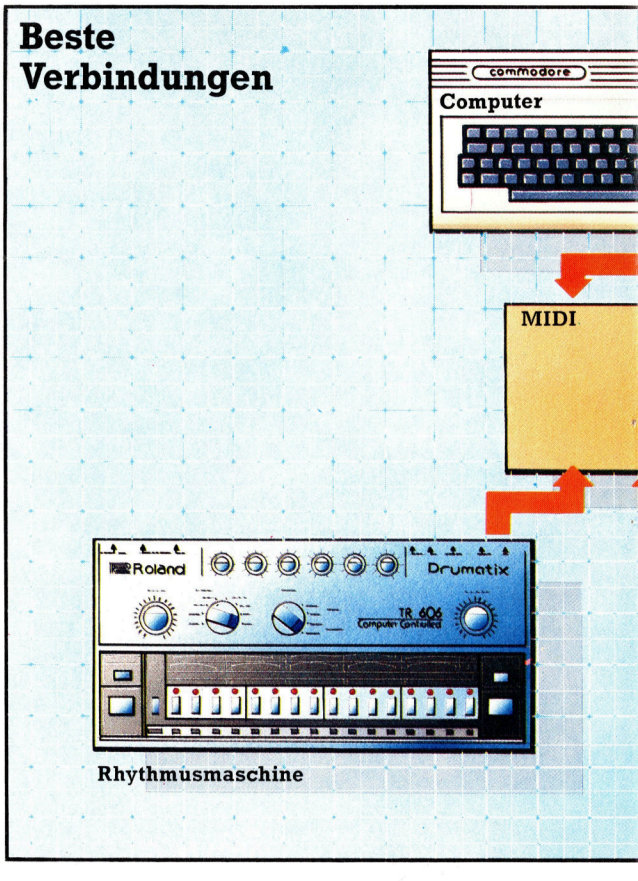
MIDI arbeitet mit einem Acht-Bit-Prozessor. Die mit der Entwicklung beauftragten Techniker hatten die Wahl zwischen zwei Übertragungsmöglichkeiten – der parallelen und der seriellen. Nachteil des parallelen Verfahrens sind die relativ hohen Kosten und der Umstand, daß ein mindestens achtadriges Kabel mit einem 25poligen D-Stecker verbunden werden muß. Bei der seriellen Übertragung werden lediglich zwei Leitungen benötigt. Über die eine sendet man die Daten hintereinander. Die zweite Leitung wird dazu benutzt, vom empfangenden Instrument Fehlermeldungen an das Hauptinstrument bzw. den Microcomputer zurückmelden zu können. Folglich verläuft die serielle Übertragung langsamer als die parallele, hat aber den Vorteil der einfacheren Verbindung und ist dadurch sehr viel kostengünstiger.

Hauptgrund für die MIDI-Entwicklung war das Bestreben, eine Kontrolleinheit zu haben, die die Bandbreite synthetisch erzeugter Musik vergrößerte und zugleich eine exakte Tonkontrolle erlaubte. Der Preis dafür sollte sich im Rahmen dessen bewegen, was Heimcomputerbesitzer, aber auch Musiker, die mit Synthesizern und Rhythmusmaschinen arbeiteten, sich leisten konnten.

Die Übertragung bei MIDI verläuft asynchron und bedient sich dabei desselben Prinzips wie das RS232-Interface. Bei der asynchronen Übertragung von Daten muß jedes Byte für das empfangende Instrument definiert werden. Bei MIDI bewirkt das ein Motorola 6850 ACIA(Asynchronous Communications Interface Adaptor)-Chip, der jedes vom Steuerinstrument ausgesandte Byte um zwei Bits ergänzt. Das erste ist eine „0", das Start-Bit, worauf die acht seriellen Datenbits folgen. Abgeschlossen wird die Übertragung mit „1" als Stop-Bit. Dieses serielle 10-Bit-Wort wird an das empfangende Instrument übertragen, worin ein zweiter ACIA-Chip die „Sendung" wieder in echte Acht-Bit-Daten umwandelt.

In einem sehr wichtigen Punkt unterscheidet sich MIDI von der RS232-Schnittstelle. Bei der RS232 beträgt die Übertragungsrate 1920 serielle Worte pro Sekunde. Das entspricht einer Rate von 19,2 KBaud. MIDI ist fast doppelt so schnell. Das beeinträchtigt die Kompatibilität mit Heimcomputern allerdings nicht, da die interne Logik des MIDI die Geschwindigkeit auf 3125 Worte pro Sekunde oder, anders ausgedrückt, mit 31,25 KBaud taktet.

MIDI wurde so gestaltet, daß es mit mehreren Instrumenten „kommunizieren" kann, vorausgesetzt, daß die richtigen Daten zum ent-



**Das „Glissando" (Veränderung der Tonhöhe von einer Note zur nächsten) ist auf Saiteninstrumenten leicht durchführbar. Dieser Effekt erfordert auf dem Synthesizer sorgfältige Programmierung, da sonst die Noten als Einzeltöne hörbar sind.**

sprechenden Gerät gesendet werden. Andernfalls könnte eine Rhythmusmaschine plötzlich versuchen, eine sorgfältig komponierte Melodie zu spielen, oder ein polyphoner Synthesizer sähe sich plötzlich veranlaßt, auf dem mittleren C ein Baßtrommel-Rhythmus-Muster zu erzeugen. MIDI-kompatible Instrumente bedürfen einer sogenannten numerischen Identifikation als Zuweisungscode, kurz ID genannt.


In einem Musik-System kann der Computer zwei mögliche Funktionen erfüllen, die von der verwendeten MIDI-Einheit abhängen: Erstrangig muß er das MIDI aktivieren und zugleich den Speicherplatz zur Aufzeichnung von Musik zur Verfügung stellen. Überdies kann er die MIDI-Software enthalten bzw. steuern, falls diese nicht ins MIDI selbst integriert ist. MIDI kann sowohl ein einfaches digitales Interface zwischen Musikinstrumenten und dem Computer sein, aber auch eine Schnittstelle, die aktiv die zu übertragenden Daten kontrolliert. Es gibt zwei Übertragungsarten: Aufzeichnung und Wiedergabe. Bei der Aufzeichnung wird die auf den Instrumenten erzeugte Musik über das MIDI in den Speicher übertragen – entweder in den des Computers oder in den Buffer des MIDI. Bei der Wiedergabe wird diese digitale Information durch das MIDI für die entsprechenden Instrumente interpretiert: Je nach Anweisung steuert es das Timing, die Synchronisation und kontrolliert die Ausführung.

Einer der 16 zur Verfügung stehenden MIDI-Kanäle wird diesem Code zugewiesen, so daß nur dieser Kanal Daten für das betreffende Instrument annimmt. Erster Teil einer vollständigen MIDI-Übertragung ist deshalb ein Status-Byte, das diesen ID beinhaltet.

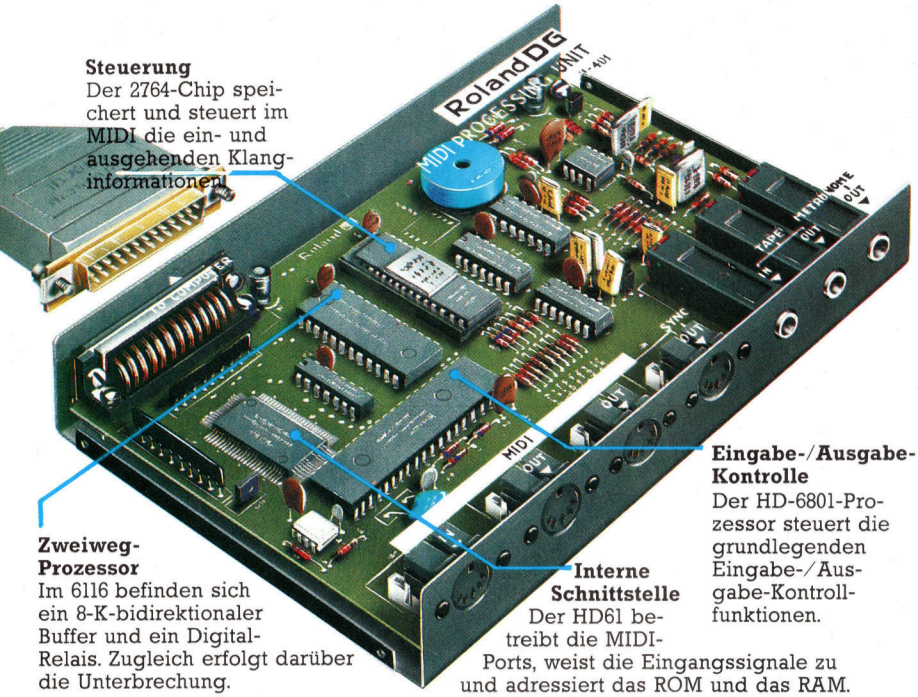
Die Einheit, etwa 100x120x45 mm groß, verfügt über zwei fünfpolige DIN-Buchsen, als „MIDI IN" und „MIDI OUT" gekennzeichnet. „MIDI IN" nimmt die Anweisungen eines Computers oder Mastersynthesizers entgegen, die dann über „MIDI-OUT" als modifizierter Bitfluß an das empfangende bzw. ausführende Instrument weitergegeben werden. Einige Modelle verfügen zusätzlich über eine zweite Ausgangsbuchse, bezeichnet als „MIDI THRU", die lediglich den an „MIDI IN" geleiteten, originalen – also unmodifizierten – Bitfluß weiterleitet.

## Eingabe-Möglichkeiten

Versetzen wir uns einmal in die Rolle eines Anwenders, der erstmals mit MIDI zu tun hat. Er möchte eine kurze Melodie ausprobieren. Die Melodie beginnt mit dem mittleren C, hebt sich dann auf das E, weiter auf das G und so fort. Wie er die Anweisungen eingibt, hängt von der verwendeten Software ab. Er kann die Anweisung mit einem Lichtgriffel über den Bildschirm, aber auch mit einer MCL (Music Composition Language) über die Tastatur seines Computers eingeben. Eine andere Möglichkeit wäre, die Noten der Melodie auf einer speziellen Musiktastatur zu spielen, die als Peripherie zur Verfügung steht. Wie immer die Musik eingegeben wird, die MIDI-Übertragung bleibt dieselbe: Damit die Melodie gespielt werden kann, gibt das erste Byte – als serielles Wort mit seinen beiden Extra-Bits vom ACIA-Chip übertragen – die Anweisung PLAY/ON, CHANNEL 6; das zweite Byte beinhaltet die erste Note.

Diese kurze Anweisung erzeugt das mittlere C auf dem empfangenden Instrument. Der Synthesizer wird das mittlere C so lange spielen, bis die Anweisung erfolgt, daß der Ton abzubrechen ist. Erfolgt diese nicht, und wird der Rest der Melodie – das E, G usw. – ohne Parameter für die Klangdauer eingegeben, klingen alle Noten der Melodie kontinuierlich weiter. Das Ergebnis wäre ein Dauerakkord, der aus den Noten der Melodie besteht.

Selbst wer die herkömmliche Art der Notation und ihre Bildschirmdarstellung nur flüchtig kennt, kann unschwer feststellen, daß aus der Melodie ein Akkord geworden ist. Der Benutzer, dem dieser Fehler unterlaufen ist, wird unproblematisch diese Sequenz spielen können, indem er einfach eine MONOPHONIC-Anweisung an den Synthesizer gibt. Monophonie, die Einstimmigkeit, steht gegen Polyphonie, Viel- bzw. Mehrstimmigkeit. Der empfangende Synthesizer kann nur jeweils einen Ton erzeugen,




**Steuerung**
Der 2764-Chip speichert und steuert im MIDI die ein- und ausgehenden Klanginformationen.

**Zweiweg-Prozessor**
Im 6116 befinden sich ein 8-K-bidirektionaler Buffer und ein Digital-Relais. Zugleich erfolgt darüber die Unterbrechung.

**Interne Schnittstelle**
Der HD61 betreibt die MIDI-Ports, weist die Eingangssignale zu und adressiert das ROM und das RAM.

**Eingabe-/Ausgabe-Kontrolle**
Der HD-6801-Prozessor steuert die grundlegenden Eingabe-/Ausgabe-Kontrollfunktionen.

**Die MIDI-Schnittstelle koordiniert die Eingabe-/Ausgabe-Anweisungen der mit ihr verbundenen Computer und Musikinstrumente genau wie jede andere Schnittstelle. Sie verarbeitet überdies die durch sie geleiteten digitalisierten Töne und fügt der Synthesizer-Eingabe vorgegebene Parameter wie Abfolge, Synchronisation und Takt hinzu.**

wenn er auf MONOPHONIC geschaltet ist. Die unkorrekt eingegebene Sequenz wird folglich als Abfolge einzelner Noten ausgeführt – eben als Melodie und nicht als Akkord.

Unterstellen wir einmal, daß der Erstanwender nach mehreren Versuchen und Fehlern die Melodie richtig eingegeben hat und der angeschlossene Synthesizer nun spielt. Die Melodie läuft, und der Rhythmus – definiert durch die Abfolge der Tondauer – ist stimmig.

Der Komponist dieser Melodie hört sich das Ergebnis mehrere Male an und bemerkt, daß sie, aufgrund der begrenzten Definition, „steif" klingt. Er entschließt sich, die Tonhöhe des C zum E gleitend zu verändern, statt die Töne nacheinander erklingen zu lassen. Diese Art des Tonwechsels nennt man „glissando" oder Tonhöhenverschiebung. Das entspricht in etwa dem, wie ein Mensch die Melodie pfeifen würde. So klingt die Darbietung auf dem Synthesizer lebhafter und fließender. Der ursprüngliche Befehl für das mittlere C wird ersetzt. Dies erfolgt durch Hinzufügen eines Extra-Bytes.

## Mögliche Einschränkungen

Damit kommen wir zu einem wichtigen Punkt bei unserer MIDI-Gesamtbetrachtung. Verfügt der empfangende Synthesizer nicht über die Möglichkeit, ein „Glissando" zu erzeugen, so kann er diese letzte Anweisung auch nicht ausführen! Er wird das mittlere C so spielen, als hätte er die ursprüngliche Anweisung empfangen, oder er spielt etwas völlig anderes. Enthalten die Befehle Anweisungen zur Erzeugung polyphoner Musik, und der empfangende Synthesizer ist monophon, wird er eine nicht genau voraussagbare Auswahl aus der polyphonen Komposition treffen und diese monophon wiedergeben. Kurz gesagt: Durch den Einsatz eines MIDI-Interfaces in Verbindung mit einem Microcomputer und einem einfachen Synthesizer werden dessen technische Möglichkeiten nicht erweitert. Durch MIDI wird aus einem einfachen Synthesizer kein kostspieliger wie etwa ein Fairlight.

Umgekehrt gelten diese Einschränkungen auch. Mag es sich bei dem empfangenden Instrument auch um einen Synthesizer handeln, der an die 40 000 Mark kostet, – sind nicht genügend musikalische Parameter eingegeben und die entsprechenden Kontrollen des Synthesizers nicht richtig eingestellt, kann ein vergleichbares Ergebnis auch mit einem einfachen, klangerzeugenden Taschenrechner erzielt werden!

In der Praxis bedeutet das: Die zweite aufgezeigte Situation läßt sich leicht ändern. Unter Verwendung der Kontrolleinheiten des Synthesizers können beliebig viele Parameter als Konstanten fixiert werden. Im Rahmen dieser Parameter sind die MIDI-Anweisungen durchführbar.

Bisher wurden nur die Beeinflussungsmöglichkeiten von Tonhöhe und -dauer berücksichtigt, doch theoretisch ermöglicht MIDI insgesamt 128 verschiedene Kontrollfunktionen, darunter Filterung, Verzerrung, „weißes Rauschen" (eine Mischung aller möglichen Frequenzen auf Zufallsbasis) und „rosa Rauschen" (Frequenzen im mittleren Klangbereich), deren Werte sich wiederum jeweils zwischen 0 und 128 bewegen können. Das ist weit mehr, als bei den meisten Synthesizern zur Einstellung verfügbar ist. Und eben diese Kontrollmöglichkeiten sind für Heimcomputerbesitzer besonders interessant.

In diesem Zusammenhang erweist sich die Übertragungsrate des MIDI als wichtiger Faktor. Wie dargelegt, benötigt man zur Formulierung eines Befehls, der lediglich eine Note darstellt und zwei Parameter definiert, drei serielle Worte. Basierend auf einer Übertragungsgeschwindigkeit von 31,25 KBaud, wird zur Übermittlung fast eine Millisekunde benötigt. Bei vielen Kompositionen legt man sechsstimmige Akkorde zugrunde. Ein solcher Akkord benötigt zur Übertragung 5,76 Millisekunden. Wird dieser Akkord unter Verwendung der MIDI-Steuerung definiert, erfolgt die Übertragung so langsam, daß das menschliche Ohr die Veränderungen der Klang-Charakteristika, bedingt durch die Verzögerung bei der Übertragung, als solche wahrnimmt. Diese Veränderungen sind vor allem dann hörbar, wenn Töne – insbesondere gleichklingende – zur selben Zeit abgespielt werden.

Aus diesem Grunde ist die serielle Übertragungsweise des MIDI-Interfaces kritisiert worden. Eine parallele Übertragung wäre effektiver gewesen. Immerhin stellt die gegenwärtig genutzte Technologie einen Kompromiß zwischen Kosten und Effizienz dar. Es kann also durchaus sein, daß das MIDI verändert wird.

# Lageranalyse

**Auch einfache Lagerprogramme wie das Dragon's Data für den Dragon 64 erzeugen genaue Listen mit erstaunlich vielen Einzelheiten. In dieser Folge über kommerzielle Software untersuchen wir die unterschiedlichen Methoden der Datenanalyse.**


Jeder Lagermeister muß sich ständig über den aktuellen Lagerbestand informieren können. Es gibt dafür zwei Methoden: die direkte Abfrage über den Bildschirm oder den Ausdruck von Listen. Das Menü des Dragon's Data zeigt eine Anzahl Möglichkeiten, Daten auszuwählen und nach unterschiedlichen Gesichtspunkten darzustellen.

Eine der wesentlichen Analysen betrifft Artikel, die schwer abzusetzen sind: die Ladenhüter. Da sämtliche Lagerbewegungen bei der Eingabe mit einem Datum versehen werden, hat das Programm bereits alle Daten, die es zur Zusammenstellung dieser Liste benötigt. Es muß nun die Vorgänge durchsuchen und Daten vergleichen. Da Firmen unterschiedliche Umsatzgeschwindigkeiten haben (der „langsame" Umsatz einer Firma kann für eine andere Firma schon eine ausgezeichnete Umsatzgeschwindigkeit bedeuten), muß der Anwender dabei die entsprechenden Vorgaben selbst festlegen können.

Beim Lagersystem des Dragon wurde dieser Vorgang gut gelöst: Der Anwender gibt ein Datum vor (z. B. 150485 für den 15. April 1985), das das Programm als Markierung für seine Suche einsetzt. Alle Artikel, bei denen nach diesem Datum keine Bewegungen mehr verzeichnet wurden, erscheinen in einer Liste. Mit der Eingabe von mehreren unterschiedlichen Daten läßt sich eine ganze Anzahl von Listen erzeugen. Das eingegebene Datum muß in den Zeitraum der vorhandenen Bewegungsdaten fallen, da sonst der Bildschirm „NO ITEMS WITHIN SELECTION CRITERION" anzeigt.

Diese Art der Datenanalyse hat den Nachteil, daß nur Artikel angezeigt werden, die keinen Umsatz hatten. Eine Firma kann aber einen Artikel, der in sechs Monaten nur zweimal umgesetzt wurde, durchaus als Ware mit schleppendem Absatz ansehen.

## Abfrage anhand anderer Kriterien

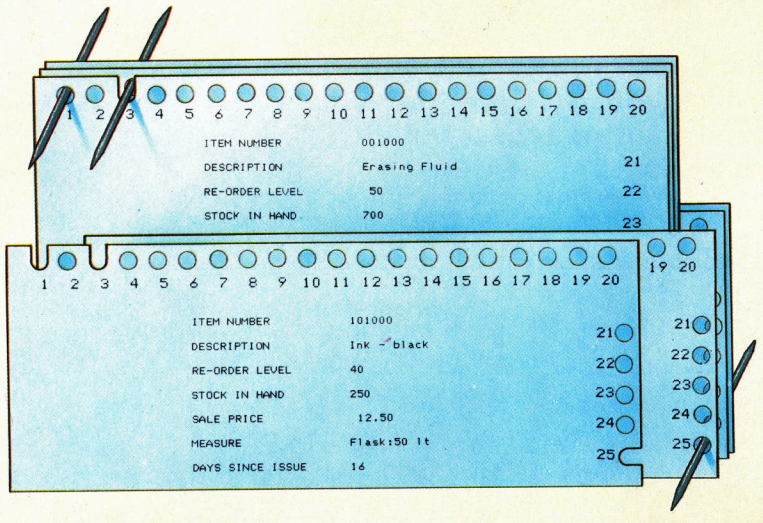
Ein höher entwickeltes System bietet da größere Flexibilität. Es gibt zwei Möglichkeiten, genauere Listen zu erzeugen: Zunächst einmal können außer dem Datum noch andere Auswahlkriterien eine Rolle spielen, wie etwa die Anzahl der Bewegungen. Dabei würden alle Waren, deren Umsatzzahlen unter einem bestimmten Wert liegen, in der Liste mit aufgeführt. Oder das System vergleicht automatisch die Bewegungen aller Artikel miteinander und zeigt erst die 50 Artikel an, die den geringsten Umsatz erzielten, dann die 50 Artikel mit dem nächsthöheren Umsatz etc.

Da auf dem Bildschirm alle Artikel einzeln angezeigt werden, dauert die komplette Ausgabe auf dem Bildschirm entsprechend lange. Aufwendigere Programmpakete bieten daher die Möglichkeit, die Daten mit Hilfe des Drukker auszugeben.

Eine Lageranalyse kann dem Anwender außer der Umsatzgeschwindigkeit aber noch andere wichtige Informationen liefern. So zeigt z. B. die Zusammenstellung aller Warenwerte an, wieviel Kapital im Lager investiert ist. Wei-

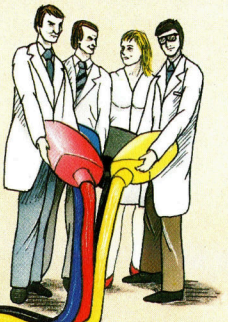
**Kommerzielle Lagerhaltungsprogramme gibt es von vielen Firmen. Im Bild sehen Sie verschiedene Softwarepakete englischer Hersteller für Heimcomputer: für den Dragon 64, den Acorn B, die Atari-Rechner, den 48K-Spectrum und den Commodore 64. Auf dem deutschen Markt sind sinnvolle Programme hauptsächlich für Personal-Computer erhältlich.**

## Lochkarten und Druckerschwärze




**Kantenlochung**
Die Kantenlochung ist eine vereinfachte Form der Datenbank. Die Löcher an der Kante werden wie Binärzahlen angesehen – ein Loch bedeutet Null, ein Schlitz eine Eins. In unserem Bild wird die Lagerhaltung einer Firma mit diesen Karten durchgeführt. Die Löcher 1 bis 3 stellen Warengruppen dar, 4 bis 6 Katalognummern und ergeben zusammengefaßt die Artikelnummer.

**Ladenhüter**
Da hohe Lagerbestände mit wenig Umsatz viel Geld kosten, untersucht eine Firma die Warengruppe 101 nach Artikeln, die in den letzten 16 Tagen nicht umgesetzt wurden. Die Analyse zeigt, daß hohe Bestände an Druckerschwärze auf Lager sind, die sich nur langsam verkaufen. Ein Sonderangebot soll diese Bestände schnell abbauen.



terhin gibt es Auskünfte über den durchschnittlichen Monatsumsatz, augenblicklich vorhandene Lagermengen und das letzte Bewegungsdatum einzelner Artikel. Auf der Grundlage dieser Daten kann der Anwender seine Geschäftsstrategie festlegen und Waren zum Beispiel mit Nachlaß anbieten.

Für eine Analyse der Bewegungsdaten sind die Darstellungsmöglichkeiten und die Breite der Selektionskriterien außerordentlich wichtig. Es sollte sich aber auch anzeigen lassen, was in einem bestimmten Zeitabschnitt mit einzelnen Artikeln geschehen ist. Zu diesem Zweck kann der Anwender beim Dragon-Programm einen bestimmten Bereich von Lagernummern angeben und die Grenzdaten einer Periode festlegen (z. B. vom 010185 bis 010485).

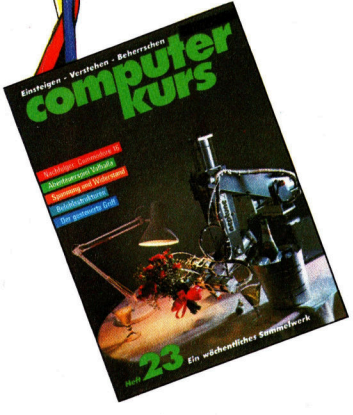
### Verschiedene Übersichten

Außer den Einzelbewegungen einer bestimmten Warengruppe hat der Anwender die Möglichkeit, eine Übersicht nach bestimmten Kriterien abzurufen: Bewegungsart, Höhe des Umsatzes, Nachbestellungen etc. Das Programm enthält ein Untermenü, in dem eine bestimmte Periode auf drei Arten festgelegt werden kann: alle Bewegungen, die Bewegungen der laufenden Abrechnungsperiode und alle Bewegungen in einem bestimmten Zeitraum. Mit weiteren Untermenüs läßt sich außerdem die Art des Vorgangs angeben. Dieses Untermenü enthält zusätzlich die Möglichkeit, über "99 ALL TYPES" alle Bewegungen innerhalb eines bestimmten Zeitraumes auszudrucken.

In dem Programm können einzelne Artikel bestimmten Warengruppen zugeordnet werden. Der Anwender hat daher auch die Möglichkeit, Warengruppen und Beschreibungen einzeln aufzurufen und festzustellen, wie hoch beispielsweise die Lagermenge für bestimmte Warengruppen ist. Bildschirmanzeigen sind dabei eine große Hilfe. Da die Analysen jedoch häufig für spätere Vergleiche benötigt werden, können fast alle Bildschirmlisten auch als „Hardcopy" auf Papier gebracht werden.

Trotz der detaillierten Analysemöglichkeiten ist die Dragon-Lagerhaltung nicht für höhere Ansprüche geeignet. Da sie als Einzelsystem konzipiert wurde, gibt es keine Möglichkeit, die Daten mit anderen kommerziellen Programmen auszutauschen oder Bestellungen mit dem Lager abzugleichen. Die wichtige Frage, ob genügend Waren am Lager sind, um eine bestimmte Bestellung ausführen zu können, kann der Computer zwar beantworten, bei größeren Bestellungen mit vielen Artikeln ist diese manuelle Abfrage jedoch sehr mühsam.

In dieser Serie haben wir den Einsatz von Heimcomputern für kleinere kommerzielle Anwendungen untersucht. Dabei wurde deutlich, daß zwar integrierte Programmpakete angeboten werden, diese aber oft nur auf einen bestimmten Teilbereich ausgerichtet sind, z. B. Geldverkehr, Lagerhaltung, Bestellwesen etc. Durch die Einschränkung werden andere Teile des Paketes oft benachteiligt. Programme dieser Art lassen sich am besten für zusätzliche Information der Geschäftsleitung einsetzen, denn trotz ihres begrenzten Wirkungsbereiches können kommerzielle Programme auf Heimcomputern die Abläufe in kleinen Firmen wesentlich verbessern. Größere Aufgaben kann nur ein Personal-Computer übernehmen.



# Andere Sprachen

**Als Nachtrag zu unserem BASIC-Programmierkurs wollen wir einige der Stärken und Schwächen von BASIC im Vergleich zu anderen Programmiersprachen aufzeigen.**

BASIC ist eine Weiterentwicklung von FORTRAN, einer der ersten Programiersprachen. Im Gegensatz zu den meisten anderen Sprachen ist BASIC eine Interpretersprache. Das bedeutet, daß bei der Ausführung eines BASIC-Programmes ein anderes spezielles Programm irgendwo im Speicher des Computers den Code Zeile für Zeile interpretiert und die BASIC-Anweisungen in Maschinensprache übersetzt. Betrachten Sie das folgende kurze BASIC-Programm:

```
 10 CLS
 20 PRINT "GEBEN SIE EINE ZAHL EIN"
 30 INPUT X
 40 PRINT "GEBEN SIE EINE ZWEITE ZAHL
    EIN"
 50 INPUT Y
 60 PRINT "DAS PRODUKT DER BEIDEN
    ZAHLEN IST: ";
 70 PRINT X*Y
 80 PRINT
 90 PRINT "WOLLEN SIE EINEN WEITEREN
    DURCHLAUF?"
100 PRINT "DRUECKEN SIE 'J' FUER
    EINEN NEUEN VERSUCH"
110 PRINT "ODER 'N' UM
    AUFZUHOEREN"
120 FOR X=1 TO 1
130 LET A$=INKEY$
140 IF A$ <> "J" AND A$ <> "N"
    THEN X=0
150 NEXT X
160 IF A$="J" THEN GOTO 10
170 END
```

Wenn der BASIC-Interpreter Zeile 10 untersucht, arbeitet er den Maschinencode aus, der zum Löschen des Bildschirmes notwendig ist. Für Zeile 20 ruft er die notwendigen Maschinencode-Anweisungen auf, um die Meldung GEBEN SIE EINE ZAHL EIN auf dem Bildschirm darzustellen. Bei Zeile 30 würde er den Speicherplatz zum Speichern einer realen Zahl reservieren, auf eine Eingabe von der Tastatur warten und dann die eingegeben Zahl in eine Binärzahl umwandeln und an dem für die Variable X reservierten Speicherplatz ablegen. Das ganze wird dann für die Zeilen 40 bis 60 wiederholt. Wenn der Anwender das Programm noch einmal verwenden möchte (indem er J eingibt), würde der Interpreter wieder zu Zeile 10 verzweigen und alle eben genannten Berechnungen erneut ausführen.

Die meisten anderen Sprachen sind „Compilersprachen". Das bedeutet, daß das Programm, nachdem es geschrieben wurde, durch einen sogenannten Compiler bearbeitet werden muß, bevor man es starten kann. Der Compiler ist ein separates Programm, das den „Source Code" durcharbeitet (das Original-Programm) und eine zweite Fassung in Maschinensprache erstellt. Wenn das compilierte Programm gestartet wird, ist es erheblich schneller als ein interpretiertes Programm, da der gesamte zeitaufwendige Prozeß der Übersetzung in Maschinensprache bereits abgeschlossen ist.

Abgesehen von der größeren Schnelligkeit compilierter Programme bieten interpretierte Sprachen einige Vorteile. Diese resultieren daraus, daß interpretierte Sprachen interaktiv sind. Das bedeutet, daß man das Programm während der Entwicklung testen und bearbeiten kann. BASIC gestattet beispielsweise, daß man an jedem beliebigen Punkt eines Programmes eine STOP-Anweisung integrieren kann. Wenn der Interpreter das STOP erreicht, unterbricht er die Interpretierung des Programmes und erlaubt die weitere Eingabe von Befehlen über die Tastatur.

## Leicht erlernbar

Befehle sind Anweisungen, die vom Interpreter direkt ausgeführt werden können. Wenn ein BASIC-Programm ausgeführt wurde (wenn z. B. der Interpreter die END-Anweisung erreicht hat) oder wenn der Interpreter eine STOP-Anweisung findet, ist es möglich, alle Variablenwerte auszudrucken.

BASIC wird oft als die ideale Sprache für den unerfahrenen Programmierer bezeichnet, da man Fehler direkt über die Tastatur beheben kann. In unserem BASIC-Programmierkurs haben wir bereits alle fundamentalen und einige der fortgeschrittenen Aspekte von BASIC behandelt. Syntax-Fehler, wie etwa 40 PRNT A(12) ergeben beim Programmlauf normalerweise leicht verständliche Fehlermeldungen, wie SYNTAX ERROR IN 40. Der Hinweis auf die Zeilennummer, in der der Fehler aufgetreten ist, erleichtert das Erkennen des Fehlers. Sobald der Interpreter einen Fehler in der Syntax

oder Logik findet, unterbricht er die Programmausführung und gibt eine Fehlermeldung ab. Die Beseitigung der Ursache ist dank der zahlreichen Editier-Befehle genauso einfach wie die Eingabe einer neuen Programmzeile. Vorausgesetzt, die Fehlerquelle wurde herausgefunden. Nach der Korrektur wird das Programm mit RUN erneut gestartet.

## Langsame Verarbeitung

Trotz aller Vorteile hat die Programmiersprache BASIC auch einige Nachteile, geringfügige und schwerwiegende. Da BASIC interpretiert wird (obwohl es auch einige Compiler für BASIC gibt), ist es in der Ausführung sehr langsam. Wenn Geschwindigkeit wichtig ist (wie z. B. bei einem Programm mit bewegter Grafik oder bei der Steuerung einer Uhr bei Laborexperimenten), ist interpretiertes BASIC viel zu langsam.

Benötigen Sie in Ihren Programmen hohe Geschwindigkeiten, gibt es nur zwei Möglichkeiten: Entweder Sie programmieren in Maschinensprache oder Assembler-Sprache – ein schwieriger und zeitaufwendiger Vorgang – oder Sie verwenden eine Compiler-Sprache wie beispielsweise PASCAL oder FORTH. Compiler-Sprachen sind nicht sehr schwer zu erlernen, doch ist es so gut wie sicher, daß der Source Code Fehler enthält, die der Compiler während des Compilierungsvorganges findet. Im Gegensatz zu BASIC sind Fehler relativ schwer zu korrigieren. Nach der Berichtigung muß der Source Code vollkommen neu compiliert werden. Die meisten Compiler brauchen zwei oder drei Durchgänge (passes) durch den Source Code, wobei bei jedem Durchgang Fehler auftreten können.

## Was kommt nach BASIC?

Ein korrekt compiliertes Programm zu erstellen ist ein außerordentlich zeitaufwendiger Prozeß. Andererseits verleitet BASIC dazu, sich schlechte Programmiertechniken anzugewöhnen, was bei höheren, strukturierten Sprachen wie PASCAL von Anfang an unmöglich ist. BASIC gestattet dem Programmierer nachlässig zu programmieren (z. B. durch die Verwendung vieler GOTOs). Diese schlechten Angewohnheiten können den Übergang zu fortgeschrittenen, strukturierten Sprachen schwierig machen.

BASIC ist eine flexible Sprache, die nicht schwer zu erlernen ist. Sie bietet ein sehr komfortables STRING-Handling, verhindert jedoch, die Fähigkeiten eines Heimcomputers voll auszunutzen. Modernere Programmiersprachen, wie PASCAL und FORTH, offerieren dagegen Programmiermöglichkeiten, die entweder zu schwer sind oder gar nicht in BASIC nachvollzogen werden können.

PASCAL wurde als Lehrsprache entwickelt und sollte speziell die Entwicklung gut konstruierter, strukturierter Programme unterstützen. PASCAL ist eine Compiler-Sprache, was für den Anwender bedeutet, daß er die durch den Compiler aufgedeckten zahlreichen Fehler jeweils einzeln beseitigen muß. Neulinge tendieren dazu, einige Punkte der Programmiersprache PASCAL als Einschränkung einer freien und flexiblen Programmierung zu empfinden, wie etwa die Notwendigkeit, alle Variablen am Anfang des Programmes festzulegen (um welchen Variablentyp es sich handelt – real, Integer usw.).

Außerdem verlangt PASCAL, daß sich der Programmierer bereits vor dem Schreiben des Ablaufs Gedanken über die Logik macht. Programme in PASCAL enthalten im Source Code fast zwangsläufig zahlreiche Syntax-Fehler. Dagegen sind logische und fundamentale Fehler relativ unwahrscheinlich.

FORTH ist als Programmiersprache für Heimcomputer eine populäre Alternative geworden. Obwohl FORTH eine höhere Programmiersprache ist, kommt ihre Ausführungsgeschwindigkeit aufgrund der einzigartigen Arbeitsweise der Maschinensprache sehr nahe. Wo bei anderen Programmiersprachen wie BASIC eine feste Anzahl an Anweisungen und Befehlen zur Verfügung steht, kann man bei FORTH, ähnlich wie bei LOGO, ein eigenes Vokabular definieren.

## Vor- und Nachteile

Das Schlüsselwort PRINT in BASIC bedeutet, daß alle nachfolgenden Zeichen in Anführungszeichen auf dem Bildschirm ausgegeben werden. Daran kann der Programmierer nichts ändern. In FORTH dagegen läßt sich PRINT so definieren, daß beispielsweise eine Liste der hexadezimalen Äquivalente der ASCII-Codes eines Strings in einer vertikalen Spalte auf dem Bildschirm ausgegeben wird.

FORTH bietet dem Programmierer die Möglichkeit, jedes Wort wunschgemäß zu definieren und ihm jede beliebige Funktion zuzuordnen. FORTH ist in dieser Richtung nicht nur sehr flexibel, sondern produziert auch Programme, die im Object Code compiliert werden können. Diese sind dann nahezu so kompakt und schnell wie Maschinensprache-Programme. FORTH ist inzwischen für zahlreiche Heimcomputer erhältlich.

**BASIC**
Leicht zu erlernen
Leicht zu merken
Leichte Fehlerbeseitigung
Langsam in der Ausführung
Verbraucht viel Speicherplatz
Unterstützt keine strukturierte Programmierung

**Assembler-Sprache**
Nicht leicht erlernbar
Nicht leicht zu merken
Schwierige Fehlerbeseitigung
Sehr schnell in der Ausführung
Ermöglicht totale Kontrolle über die CPU

**FORTH**
Nicht sehr leicht erlernbar. Einfacher für totale Anfänger, schwieriger für BASIC-Programmierer
Relativ leicht zu merken
Fehlerbeseitigung im Interpreter-Modus sehr einfach
Kann compiliert werden. Geschwindigkeit fast so hoch wie bei ASSEMBLER-Sprache
Bietet komplette Kontrolle über die CPU
Sehr sparsam im Speicherplatzbedarf
Leichter zu erlernen als ASSEMBLER-Sprache, jedoch weniger „intuitiv" als BASIC

**PASCAL**
Relativ leicht erlernbar
Relativ leicht zu merken
Fehlerbeseitigung schwieriger als in BASIC
Unterstützt bessere Programmiertechniken
Ausführungsgeschwindigkeit schneller als BASIC, jedoch langsamer als ASSEMBLER
Muß compiliert werden, was Zeit kostet. Nach korrekter Compilierung ist die Geschwindigkeit fast so schnell wie bei ASSEMBLER.
Bietet eine gute Kontrolle über die CPU, jedoch nicht so gut wie ASSEMBLER.
Das String-Handling ist nicht so leicht wie bei BASIC.

Obwohl es noch viele weitere Programmiersprachen gibt, neigen die meisten Hobby-Programmierer dazu, nach BASIC entweder die ASSEMBLER-Sprache, PASCAL oder FORTH zu erlernen. Die Vor- und Nachteile der einzelnen Sprachen sind in Kurzform aufgeführt.

## BASIC-Dialekte

**LYNX 96**

```
 1 REM *ERSTELLE DATEN-
   DATEI
 2 DIM N$ (30)
 3 LET N$="@ERST"
 4 DIM F$(15)
 5 LET F$="TEST"
 6 LET Z=2
 7 EXTBACK 1
 8 EXTSTORE 1,Z,N$,N$,N$
 9 EXTSTORE 1,F$,F$, F$, F$
10 INPUT "LEGE DATENCASSETTE
   EIN, DRUECKE RECORD UND
   'J'"; A$
11 SSAVE 1, "ADDBKDAT"
12 PRINT "STOPPE BAND, SPULE
   ZURÜCK"
13 END
```

ANMERKUNG: Dies ist das Initialisierungsprogramm für den 96K-Lynx.

**Hauptprogramm-Variablen**
Kopieren Sie die Spectrum-Liste mit den folgenden Änderungen für die numerischen Variablen:

Ersetze: GROSS durch Z
VMOD durch R
SRTD durch D
CURR durch C
WAHL durch H
BTM durch b
MD durch m
TP durch t

Außerdem müssen Sie die folgenden Ergänzungen und Änderungen durchführen:

```
1100 REM *CREARR* U/R
1110 DIM N$ (30) (50)
1120 DIM M$ (30) (50)
1130 DIM S$(30) (50)
1140 DIM T$(15) (50)
1150 DIM C$(15) (50)
1160 DIM R$(15) (50)
1170 DIM X$(15) (50)
1180 DIM Z$(30)
1210 LET Z=0
1220 LET R=0
1230 LET D=1
1240 LET C=0
1250 LET Z$="@ERST"
1260 LET Q$=""
1300 RETURN
.
1400 REM *LSINDT* U/R
1405 PRINT "DATENCASSETTE
     EINLEGEN UND PLAY
     DRUECKEN"
1410 GOSUB 3100
1420 SLOAD 1, "ADDBKDAT"
1430 PRINT "STOPPE BAND"
1440 GOSUB 3100
1450 EXTBACK 1
1460 EXTFETCH 1,Z
1470 FOR K=1 TO Z-1
1480 EXTFETCH 1, N$(K), M$(K),
     S$(K), T$(K), C$(K), R$(K),
     X$(K)
1490 NEXT K
1500 LET Q$=N$(1)
1510 RETURN
.
3120 IF KEYN <> 32 THEN LET L=0
.
3780 LET A$=KEY$
.
3810 LET H=VAL(A$)
3820 IF (H < 1) OR (H > 9) THEN
     LET L=0
.
4500 REM *MODNAM* U/R
4510 REM WANDLE IN GROSS-
     BUCHSTABEN UM
4520 LET D$=UPC$(N$(Z)) (lösche
     Zeilen 4530—4590)
.
4600 LET P$=""
4601 LET A$=""
4602 LET T=LEN(D$)
4603 LET S=0
.
4610 REM LOKALISIERE
     LETZTE LEERSTELLE
.
4630 IF MID$(D$,L,1)=" " THEN
     LET S=L
.
4670 IF MID$(D$,L,1)>"@" THEN
     LET P$=P$+MID$(D$,L, 1)
.
4710 IF MID$(D$, L, 1)> "@" THEN
     LET A$=A$+MID$(D$,L,1)
```

Die Zeilen 5410 bis 5460 müssen in einzelne Anweisungen umgewandelt werden. Z. B.:

```
5410 LET U$=N$(L):LET N$(L)=N$(T)
     :LET N$(T)=U$
```

wird zu

```
5410 LET U$=N$(L)
5411 LET N$(L)=N$(T)
5412 LET N$(T)=U$
```

Wandeln Sie in dieser Form alle Zeilen um.

```
.
5600 REM *SPEVER* U/R
5605 PRINT "LEGE DATENCASSETTE
     EIN UND DRUECKE RECORD"
```

```
5610 GOSUB 3100
5620 EXTBACK 1
5630 EXTSTORE 1,Z
5640 FOR K=1 TO Z-1
5650 EXTSTORE 1, N$(K), M$(K),
     S$(K), T$(K), C$(K), R$(K),
     X$(K)
5660 SSAVE 1, "ADDBKDAT"
5670 PRINT "STOPPE BAND"
5680 GOSUB 3100
5690 RETURN
.
5855 LET X=0
5860 LET m=INT((b+t)/2)
5870 IF M$(m)=U$ THEN LET X=1
5880 IF U$ > M$(m) THEN LET
     b=m+1
.
6080 LET A$=KEY$
.
6110 IF A$=" " THEN RETURN
6120 GOSUB 6200
6130 RETURN
.
6730 FOR I=1 TO 1
6735 LET I=0
6740 LET A$=KEY$
6750 IF (A$=E$) OR (A$=" ") THEN
     LET I=1
6760 NEXT I
```

Dieser Teil muß in den Zeilen 6880-6910, 6990-7030, 7110-7140, 7220-7250 und 7640-7670 reproduziert werden.

DRAGON 32

BBC MICRO

COMMODORE 64

VIC-20

**Initialisierungsprogramm**
Das Initialisierungsprogramm für den Dragon 32:

```
1 REM *ERSTELLE DATENDATEI
2 LET Z=2
3 LET N$="@ERST"
4 OPEN"0",#-1, "ADDBKDAT"
5 INPUT "DATENCASSETTE EIN-
  LEGEN, DRUECKE RECORD UND
  'J' ";A$
6 PRINT#-1,Z,N$,N$,N$,N$,N$,N$,N$
7 CLOSE #-1
8 PRINT "STOPPE BAND, SPULE
  ZURÜCK"
9 STOP
```

Beim Acorn B ersetzen Sie die Zeilen 4, 6 und 7 durch:

```
4 F1=OPENOUT("ADDBKDAT")
6 PRINT #F1,Z,N$,N$,N$,N$,N$,
  N$,N$
7 CLOSE#F1
```

Beim Commodore 64 und VC 20 ersetzen Sie die Zeilen 4, 6 und 7 durch:

```
4 OPEN  1,1,2,"ADBKDAT"
6 PRINT #1,Z:PRINT #1,N$:PRINT #1,
  N$:PRINT #1,N$:PRINT #1,
  N$:PRINT #1,N$:PRINT #1,
  N$:PRINT #1,N$
7 CLOSE 1
```

**Hauptprogramm-Variablen**
Beim Dragon, den Commodore-Computern und dem Acorn B kopieren Sie bitte die Spectrum-Liste und nehmen die folgenden Änderungen in Bezug auf die numerischen Variablen vor.

```
Ersetze: GROSS durch Z
         VMOD  durch R
         SRTD  durch D
         CURR  durch C
         WAHL  durch H
         BTM   durch BT
         MD    durch MD
         TP    durch TP
```

Außerdem sind die folgenden Änderungen und Ergänzungen auszuführen:

```
1100 REM *CREARR* U/R
1110 DIM N$(50)
1120 DIM M$(50)
1130 DIM S$(50)
1140 DIM T$(50)
1150 DIM C$(50)
1160 DIM R$(50)
1170 DIM X$(50)
```

Löschen Sie die Zeilen 1180 bis 1190.

```
1210 LET Z=0
1220 LET R=0
1230 LET D=1
1240 LET C=0
1250 LET Z$="@ERST"
1260 LET Q$=""
1300 RETURN
```

Dies ist die Dragon 32-Fassung der Unterroutine ab Zeile 1400:

```
1400 REM *LSINDT* U/R
1410 OPEN"I",#-1, "ADBKDAT"
1420 PRINT "DATENCASSETTE EIN-
     LEGEN UND PLAY DRUECKEN"
1430 GOSUB 3100
1440 INPUT#-1,Z
1450 FOR K=1 TO Z-1
1460 INPUT#-1,N$(K),M$(K),S$(K),
     T$(K),C$(K),R$(K),X$(K)
1470 NEXT K
1480 Q$=N$(1)
1490 CLOSE#-1
1500 PRINT "STOPPE BAND"
1510 GOSUB 3100
1520 RETURN
```

In der vorherigen Liste müssen Sie beim Acorn B Zeile 1410 wie folgt ersetzen:

```
1410 F1=OPENIN
     ("ADBKDAT")
```

Außerdem muß #-1 in den Zeilen 1440, 1460 und 1490 durch #F1 ersetzt werden.

Beim Commodore 64 und VC 20 muß Zeile 1410 wie folgt ersetzt werden:

```
1410 OPEN 1,1,0, "ADBKDAT"
```

Ersetzen Sie außerdem #-1 in den Zeilen 1440 und 1460 durch #1 sowie Zeile 1490 wie folgt:

```
1490 CLOSE 1
```

Beim Acorn B muß durchgehend INKEY$ durch INKEY$(0) ersetzt werden. Die Anweisungen INPUT "... Text ..."; A$ müssen durch INPUT " ...Text ..",A$ ersetzt werden. Bei den Commodore-Computern ersetzen Sie im gesamten Programm LET A$=INKEY$ durch GET A$. Außerdem muß IF INKEY$ ... durch GET GT$:IF GT$ .. ersetzt werden.

Beim Acorn B, dem Dragon und den Commodore-Computern müssen in der Unterroutine ab Zeile 4500 alle Referenzen zu D$(L) durch MID$(D$,L,1) ersetzt werden. Ersetzen Sie CODE .. durch ASC(...). Ersetzen Sie in Zeile 4610 ERST durch LAST.
Löschen Sie: LET L=T ab Ende der Zeile 4630.

Dies ist die Dragon-Version der Unterroutine ab Zeile 5600. Für den Acorn B und die Commodore-Computer sehen Sie die Anmerkungen zur Initialisierung (s.o.).

```
5600 REM *SPEVER* U/R
5610 OPEN "O"#-1, "ADBKDAT"
5620 PRINT "DATENCASSETTE EIN-
     LEGEN UND RECORD
     DRUECKEN"
5630 GOSUB 3100
5640 PRINT#-1,Z
5650 FOR K=1 TO Z-1
5660 PRINT#-1,Z,N$(K),M$(K),S$(K),
     T$(K),C$(K),R$(K),X$(K)
5670 NEXT K
5680 CLOSE #-1
5690 PRINT "STOPPE BAND"
5693 GOSUB 3100
5695 RETURN
```

In der Unterroutine ab Zeile 6200 fügen Sie beim Acorn B ein:

```
6205 VDU 2
6275 VDU 3
```

Außerdem müssen Sie LPRINT durch PRINT ersetzen.

Bei den Commodore-Computern fügen Sie ein:

```
6205 OPEN 4,4:CMD 4
6275 PRINT #4:CLOSE 4
```

Außerdem müssen Sie LPRINT durch PRINT ersetzen.

Beim Dragon ersetzen Sie LPRINT durch PRINT #-2.

# Spurensicherung

**Viele Heimcomputer verfügen zwar über hervorragende Grafik, aber die Eingabe einer Zeichnung vom Papier auf den Bildschirm des Rechners kann sehr schwierig und zeitraubend sein. Hier bieten Digital-Tracer erhebliche Erleichterung.**

Mit einem Digital-Tracer können Sie Zeichnungen, Fotos oder Entwürfe direkt auf den Bildschirm Ihres Rechners übertragen. Wie leicht oder schwer das ist, hängt weitgehend von der mitgelieferten Software ab. Hier geht es um drei Tracer-Modelle für den Acorn B und eins für den Sinclair Spectrum.

Alle vier Tracer arbeiten nach dem gleichen Prinzip: Die Zeichenspitze, die elektrische Signale an den Computer übermittelt, ist an einem Doppelgelenkarm befestigt. Die Signalspannungen hängen von der Position der Spitze ab; sie werden digitalisiert, und über den Rechner wird ein Punkt an der entsprechenden Stelle des Bildschirms erzeugt. Die dazugehörige Software bietet zusätzliche Möglichkeiten, etwa Linien in unterschiedlichen Farben zu zeichnen usw. Bei den Modellen für den Acorn sind verschiedene Darstellungsarten wählbar, wobei wegen der begrenzten Speicherkapazität größerer Farbumfang weniger Auflösung bedeutet.

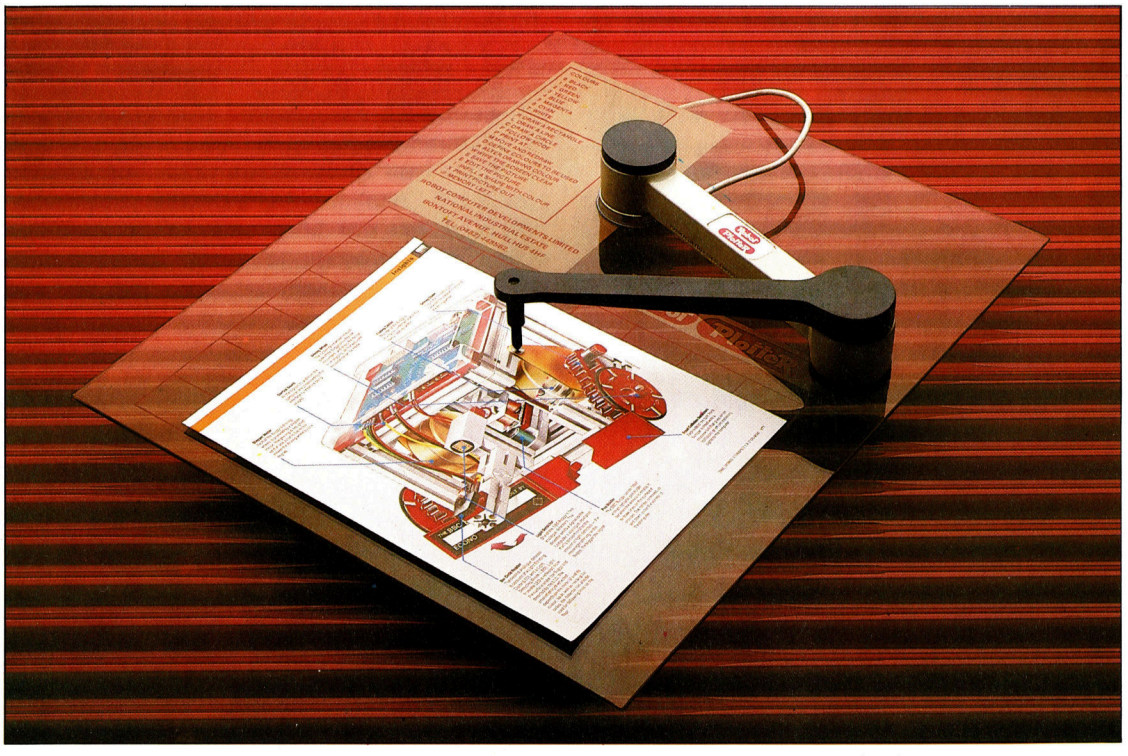
Von den vier untersuchten Geräten hat der „Robot Plotter" der Firma Robot Computer Development das beste Aussehen. Der Gelenkarm ist hier auf einer Plexiglas-Grundplatte verankert, die mit einem Strichraster versehen ist. Der Arm ist solide aus kräftigen Metallprofilen und Kunststoffteilen gebaut. Leider beeinträchtigt diese Konstruktion merklich die freie Sicht auf die Vorlage.

Der Robot Plotter wird mit Cassetten-Software ausgeliefert. Außer den Zeichen-Routinen sind darauf noch Programme zum Entwurf von Kreisen, Kästchen und Linien enthalten.

Das Programm speichert die Bilder, z. B. kartografische Skizzen, als eine Folge von Linienzügen. Dank dieses Prinzips können auch einzelne Linien unkompliziert korrigiert werden. Andererseits wird bei der Speicherung von detailreichen Bildern schnell die Kapazitätsgrenze des Acorn B erreicht.

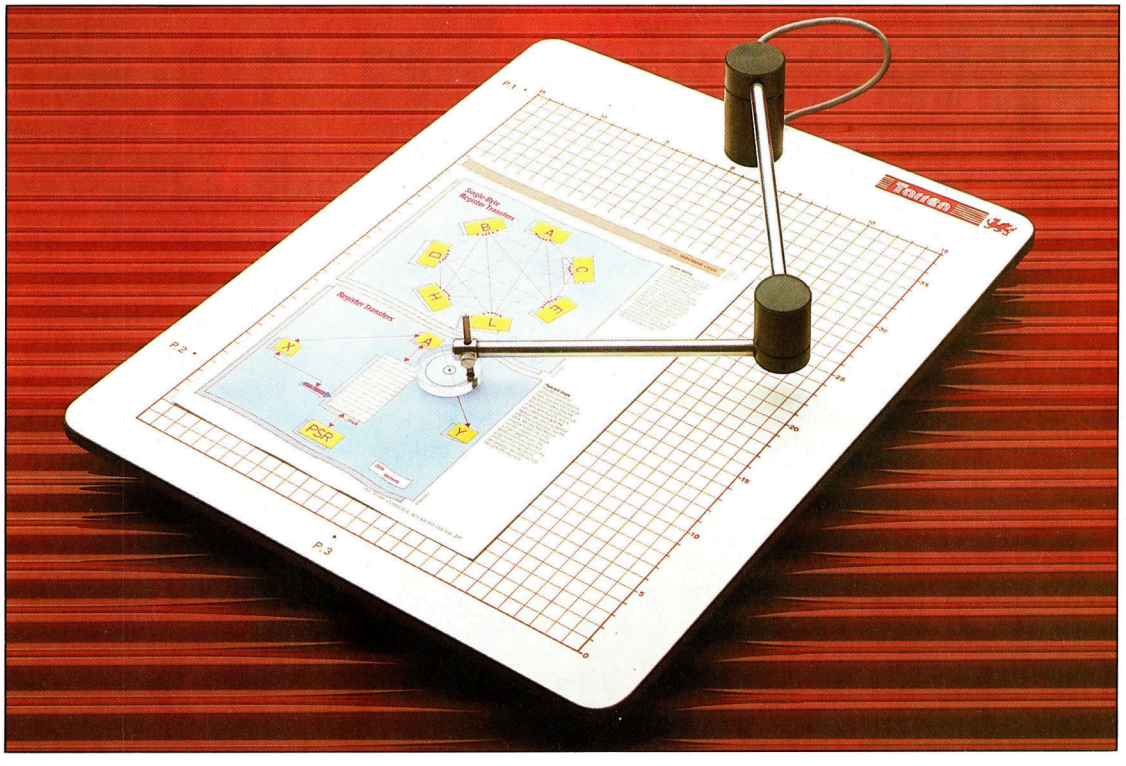
## Konstruktionsmerkmale

Auch der „Digigraph" ist ein stabiler Tracer. Der Gelenkarm aus Aluminiumrohr trägt eine Plexiglasscheibe mit einer Markierung. Der Arm läßt sich zügig über die Vorlage auf der Grundplatte führen – dies Gerät ist das handlichste von den hier gezeigten. Dafür bietet die Software weniger Möglichkeiten als die des



**Die grafischen Vorlagen werden zur Abtastung unter die Arbeitsplatte aus Plexiglas gelegt, die mit einem Gitterraster versehen ist. Der Arm ist aus kräftigen Metallprofilen und Kunststoffteilen zusammengesetzt und wirkt sehr stabil. Während die Stiftspitze über die Zeichnung geführt wird, werden die Drehwinkel der Gelenke über Meßwertaufnehmer zum Rechner übertragen. Die mitgelieferte Software unterstützt nicht nur das Übertragen der Zeichnungen, sondern auch das Einfügen von Kreisen und Vierecken.**

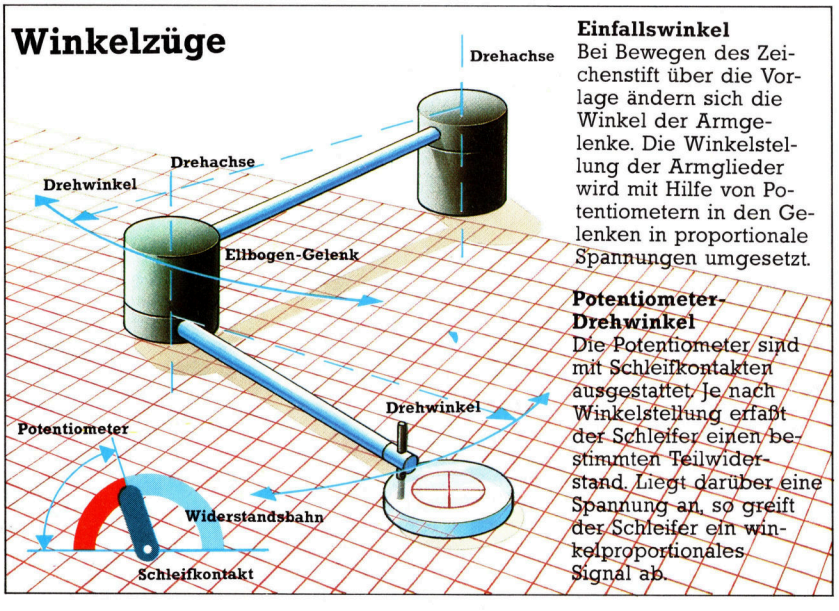
**Das Gerät ist mit einem soliden Gelenkarm aus Aluminiumrohr ausgestattet, das auf einer hölzernen Grundplatte mit einem Gitterraster befestigt ist.**



Robot-Plotters. Beim Digigraph werden nicht die einzelnen Linienzüge abgespeichert, sondern die Grafik wird unmittelbar auf den Bildschirm übertragen und anschließend als kompletter Schirminhalt gespeichert. Das erschwert zwar die Bildbearbeitung, aber Sie brauchen für ein komplexes Bild nicht mehr Speicherplatz als für eine einfache Strichzeichnung. Beim Digigraph erhält der Benutzer außerdem mehrere Übungsvorlagen, an denen er die optimale Handhabung des Geräts erlernen soll. Im Lieferumfang des Digigraph sind die Software (auf Cassette oder Diskette), ein ausgezeichnetes Handbuch und Übungsvorlagen enthalten.

Das dritte und das vierte Gerät sind unterschiedliche Versionen des gleichen Tracers von RD Laboratories für den Acorn B bzw. den Sinclair Spectrum. Dabei wird der Gelenkarm ohne Grundplatte geliefert und muß mit einem beigefügten doppelseitigen Klebestreifen auf einer geeigneten Zeichenunterlage befestigt werden. Der Kunststoffarm ist zwar biegsam, erlaubt aber trotzdem präzise Zeichnungen.

## Winkelzüge




**Einfallswinkel**
Bei Bewegen des Zeichenstift über die Vorlage ändern sich die Winkel der Armgelenke. Die Winkelstellung der Armglieder wird mit Hilfe von Potentiometern in den Gelenken in proportionale Spannungen umgesetzt.

**Potentiometer-Drehwinkel**
Die Potentiometer sind mit Schleifkontakten ausgestattet. Je nach Winkelstellung erfaßt der Schleifer einen bestimmten Teilwiderstand. Liegt darüber eine Spannung an, so greift der Schleifer ein winkelproportionales Signal ab.

## Qualitätsunterschiede

Die Übertragung der Vorlage mit Hilfe des Fadenkreuzes ist allerdings etwas mühsam.

Die Bilder werden ähnlich wie beim Robot Plotter abgelegt, so daß der Speicherplatz wieder knapp wird. Die mitgelieferte Software ist recht umfangreich und enthält Programme zum Zeichnen von Kreisen, Vierecken und Linien sowie zur Erstellung von Trickfilmszenen. Dabei wird von der Möglichkeit Gebrauch gemacht, die Farbskala des Acorn B neu zu definieren. Eine Demonstrationscassette zeigt alle Anwendungen dieses Tracers.

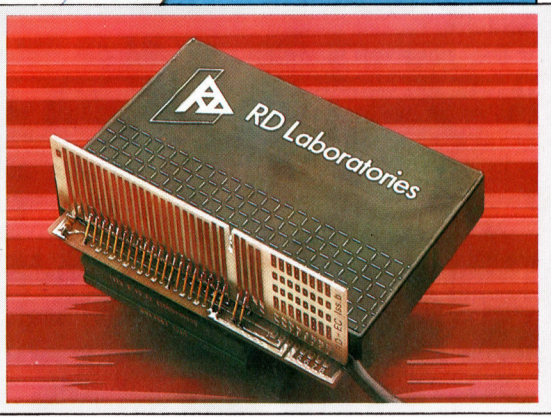
Bei der Spectrum-Version des RD-Tracers wird ein Analog-Digital-Wandler mitgeliefert, der bei der Acorn-Version bereits integriert ist. Kontinuierliches Zeichnen ist möglich, wird aber durch die träge Reaktion der Software erschwert: Eine glatte Kurve erscheint auf dem Bildschirm leicht als „eckiger" Ploygonzug. Die Software unterstützt auch hier das Zeichnen von Kreisen, Rechtecken usw. Wenn dabei die Linien über den Bildschirmrand hinausragen, erfolgt im Unterschied zu den üblichen Spectrum-Programmen keine Fehlermeldung.

Für alle diese Tracer lassen sich Programme zur direkten Ablesung der Position der Zei-


**Die hier gezeigte Version für den Acorn B ist eine biegsame Plastikkonstruktion, die ohne Grundplatte geliefert wird. Der Zeichenarm wird mit einem doppelseitigen Klebeband auf einer festen Unterlage befestigt.**

chenspitze schreiben. Beim Sinclair Spectrum ist dies allerdings nicht einfach, weil er – anders als der Acorn B – (in der Grundausstattung) keinen Joystick-Eingang hat und somit die zugehörigen BASIC-Befehle fehlen.

Die Acorn-Besitzer haben nun die Qual der Wahl: Der mechanisch wenig zuverlässige RD-Tracer bietet bei niedrigem Preis ein gutes Software-Paket; für den wesentlich solideren Robot Plotter gilt das ebenfalls, aber hier fällt die Abtastung nicht besonders leicht – und beim Digigraph ist die Software zwar weniger ausgefeilt, jedoch ist das Gerät stabil und besser zu handhaben.



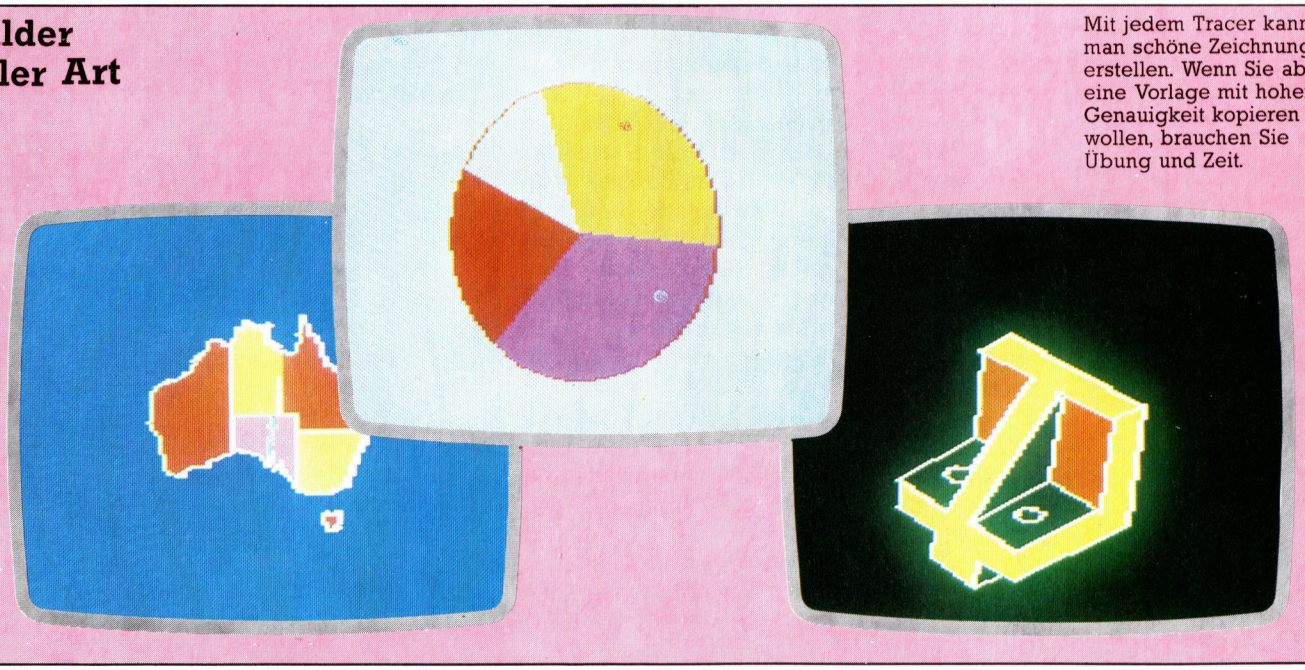

## Spectrum-Interface

für den RD-Tracer

Bei der Spectrum-Version des RD-Tracers wird genau der gleiche Abtastarm wie bei der oben gezeigten Ausführung für den Acorn B verwendet. Der Tracer wird über ein spezielles Interface mit Analog/Digital-Wandler angeschlossen.

Eine andere Möglichkeit, interessante Zeichnungen zu erstellen, bieten die Grafiktabletts mit der dazugehörigen Software. Einige Systeme erlauben sogar die Erstellung kompletter Trickfilme durch die Kombination mit Animationsprogrammen.

## Bilder aller Art



Mit jedem Tracer kann man schöne Zeichnungen erstellen. Wenn Sie aber eine Vorlage mit hoher Genauigkeit kopieren wollen, brauchen Sie Übung und Zeit.

# Bauteile

**Jedes elektronische Gerät besteht aus einer Vielzahl von Bauteilen. Bei Microprozessoren oder Videochips zum Beispiel müssen zahlreiche Funktionen auf kleinstem Raum zusammengefaßt werden.**

Auch das komplizierteste Bauelement besteht meist aus einer Vielzahl von einfachen Teilen. Diese Grundbausteine werden in der Elektrotechnik zwei verschiedenen Gruppen zugeordnet: Es gibt „aktive" und „passive" Bauteile.

Die passiven Bauelemente sind die einfacheren: Sie hemmen den Fluß elektrischer Signale, wobei unterschiedlichen Signalen ein jeweils anderer Widerstand entgegengesetzt wird. Ein Beispiel dafür sind Widerstände und Kondensatoren.

Aktive Komponenten können ein wenig mehr: Sie modifizieren das Ur-Signal. So zum Beispiel der Transistor – ihm wird ein schwaches Signal zugeführt, das er verstärkt wieder abgibt.

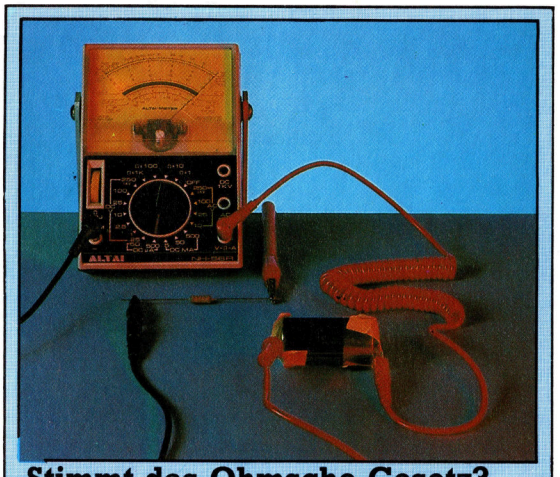
## Die Halbleiter

Passive Bauteile bestehen aus einfachen Werkstoffen. Abgesehen vom Isoliermaterial ist ein Widerstand etwa aus Kupfer, Stahl und Kohle aufgebaut – alles leitfähige Elemente. Dagegen bestehen die aktiven Bauteile fast alle aus Germanium oder Silizium. Diese Elemente haben eine besondere Eigenschaft: Anders als etwa ein Metall leiten sie nur unter ganz bestimmten Bedingungen. Sie werden daher als „Halbleiter" bezeichnet.

Wegen ihres Aufbaus aus halbleitendem Material haben die aktiven Bauteile eine etwas ungewohnte Funktion: Wird zum Beispiel eine bestimmte Spannung über einem Halbleiter angelegt, kann man daraus noch nicht den fließenden Strom berechnen. Die passiven Bauteile machen es unserem Verständnis da etwas leichter, weil sich ihr Verhalten als leitfähiges Material aus den Ohmschen Gesetzen ermitteln läßt.

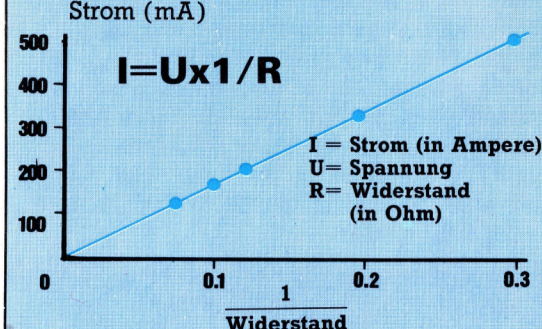
Für den Computer-Bereich ist der Transistor besonders wichtig, weil er sowohl für die Speicherung als auch für die Bearbeitung von Informationen gebraucht und deshalb entsprechend eingesetzt wird.

In der nächsten Folge wird gezeigt, wie man die Funktionen einfacher Logik-Gatter mit Hilfe von Transistoren und einer Steckleiste problemlos nachbauen kann.



### Stimmt das Ohmsche Gesetz?

Mit etwas Draht, ein paar Widerständen, einer Batterie und einem Meßgerät läßt sich das ganz leicht feststellen: Die Widerstände sollten Werte zwischen 3,3 und 15 Ohm aufweisen und mit 1 Watt belastbar sein. Stellen Sie das Meßgerät auf Strommessung bis ca. 400 mA ein. Der Stromkreis wird nun wie oben im Bild geschlossen. Setzen Sie nacheinander die verschiedenen Widerstände ein und lesen jeweils das Meßgerät ab. Wenn Sie Strom und Kehrwert des Widerstandes in das Koordinatensystem eintragen, ergibt sich eine gerade Linie, in der bei konstanter Spannung I genau proportional 1/R ist.




### Dioden


Dioden können Sie sich als „elektrisches Ventil" vorstellen. In der einen Durchgangsrichtung haben sie für den Strom nur einen ganz geringen Widerstand von wenigen Ohm. Mehrere Millionen Ohm setzt die Diode dem Stromfluß in der anderen Richtung entgegen. Dioden gehorchen dem Ohmschen Gesetz nicht – sie sind aus Halbleitermaterial aufgebaut. Der Ring auf dem Gehäuse gibt die Durchlaßrichtung an.

## Spectrum

```
 20 DIM b$(3,7)
 25 DIM c$(4,7)
 30 INPUT "input colour of bands"
 40 INPUT "band number 1",b$(1)
 50 IF b$(1)="gold" OR b$(1)="silver" T
    HEN  GO TO 30
 60 INPUT "band number 2",b$(2)
 70 INPUT "band number 3",b$(3)
 80 PRINT
 90 FOR i=1 TO 3
100  RESTORE 180
110 FOR j=0 TO 9
120 READ c$(1),c$(2),c$(3),c$(4)
130 IF c$(1)=b$(i) THEN  GO SUB 1000
140 NEXT j
150 NEXT i
160 PRINT "  ohms"
170 STOP
180 DATA "black","0","0",".0","brown","
    1","1","0","red","2","2","00","
    orange","3","3","000"
190 DATA "yellow","4","4","0000","green
    ","5","5","00000","blue","6","6","
    000000"
200 DATA "violet","7","7","0000000","gr
    ey","8","8",".0","white","9","9",".0"
1000 FOR k=1 TO 7
1020 IF c$((i+1),k)<>" " THEN  PRINT c$(
(i+1),k);
1025 NEXT k
1030 RETURN
```

## Acorn B

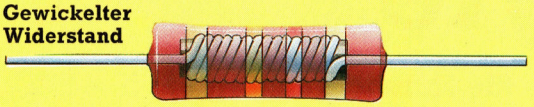
```
 10 REM resistor colour codes
 20 DIM B$(3),C$(3)
 30 PRINT "Input colour of bands:"
 40 INPUT "BAND NUMBER 1",B$(1)
 50 IF B$(1)="GOLD" OR B$(1)="
    SILVER" THEN GOTO 30
 60 INPUT "BAND NUMBER 2",B$(2)
 70 INPUT "BAND NUMBER 3",B$(3)
 80 PRINT
 90 FOR I=1 TO 3
100   RESTORE
110   FOR J=0 TO 9
120     READ C$(0),C$(1),C$(2),C$(3)
130     IF C$(0)=B$(I) THEN PRINT C$(I);
140     NEXT J
150   NEXT I
160 PRINT "  ohms"
170 END
180 DATA BLACK,0,0,, BROWN,1,1,0,
    RED,2,2,00, ORANGE,3,3,000
190 DATA YELLOW,4,4,0000, GREEN,5,5,
    00000, BLUE,6,6,000000
200 DATA VIOLET,7,7,0000000, GREY,8,8,,
    WHITE,9,9,,
```

Beim Commodore 64 ersetzen Sie die Zeilen 40, 60 und 70 durch:

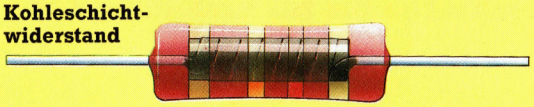
```
40 INPUT "BAND
   NUMMER 1 ";B$(1)
60 INPUT "BAND
   NUMMER 2 ";B$(2)
70 INPUT "BAND
   NUMMER 3 ";B$(3)
```

## Widerstände

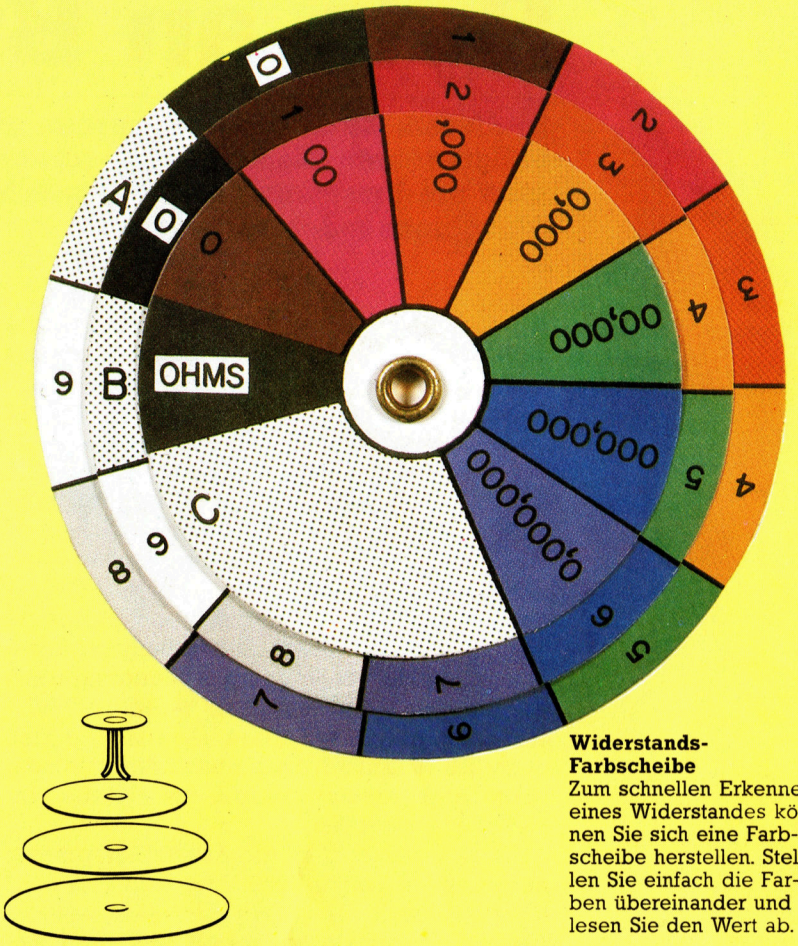
**Gewickelter Widerstand**



**Kohleschicht-widerstand**

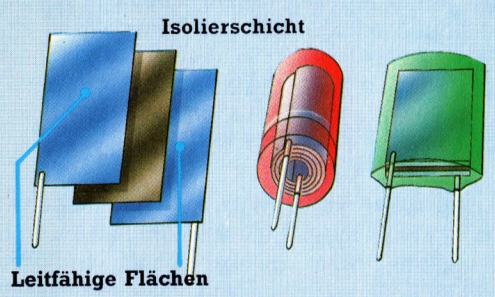


Widerstände sind einfachste elektronische Bauelemente. Bei gewickelten Widerständen ist ein Widerstandsdraht um einen Kern herum aufgerollt und mit einer Isolierschicht abgedeckt. Der Strom muß durch die gesamte Drahtlänge hindurchgehen und wird dabei „gebremst". Kohleschichtwiderstände arbeiten ähnlich.
Beide Widerstands-Typen werden durch farbige Ringe gekennzeichnet, von denen sich der jeweilige Wert des Bauteils ablesen läßt. Die zwei ersten Ringe bestimmen den Zahlenwert, der mit dem Wert des dritten Rings multipliziert den Widerstand angibt. Silberne und goldene Ringe definieren die Genauigkeit und zeigen, von welcher Seite her die Ringe gelesen werden – man liest immer auf die silbernen und goldenen Ringe zu. Damit Sie die Farben nicht auswendig lernen müssen, können Sie das nebenstehende Programm zur Bestimmung eines Widerstandes verwenden.

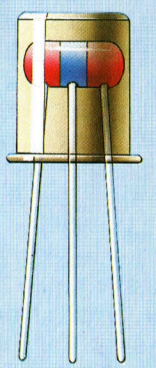


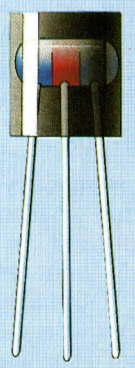

**Widerstands-Farbscheibe**
Zum schnellen Erkennen eines Widerstandes können Sie sich eine Farbscheibe herstellen. Stellen Sie einfach die Farben übereinander und lesen Sie den Wert ab.

## Kondensatoren




Kondensatoren sind Widerstände für Wechselspannungen. Wechselspannungen hoher Frequenz gehen leichter durch sie hindurch als solche mit niedriger Frequenz. Man verwendet sie daher häufig zum Ausfiltern bestimmter Frequenzen. Kondensatoren bestehen aus zwei leitfähigen Flächen, die durch eine Isolierschicht getrennt sind. Je nach der erforderlichen Spannungsfestigkeit wird die Isolierschicht aus speziellen Salzen oder aus Keramik hergestellt. Um die Kondensatoren möglichst kompakt zu halten, werden die drei Schichten meist aufgewickelt und mit Isoliermasse vergossen.





## Transistoren

Mit der Erfindung des Halbleiterelementes „Transistor" begann das Zeitalter der Elektronik – von allen vorgestellten Bauteilen ist der Transistor das komplizierteste. Transistoren können auf zweifache Weise eingesetzt werden: einmal als Verstärker, der einen geringen Eingangsstrom in einen hohen Ausgangsstrom umwandelt, zum anderen als Schalter, mit dem sich der Stromfluß elektronisch unterbrechen und fortsetzen läßt. Auf dieser Fähigkeit basiert die digitale Elektronik.
Transistoren bestehen wie Dioden aus halbleitendem Material. Sie haben jedoch insgesamt drei Anschlüsse, die als Basis, Kollektor und Emitter bezeichnet werden. Ob der Transistor als Schalter oder als Verstärker arbeitet, hängt von der Position seiner Anschlüsse in der Gesamtschaltung ab. Bei Transistoren können Sie nicht ohne weiteres erkennen, welcher Anschluß nun Basis, Emitter oder Kollektor ist – das variiert jeweils bei den unterschiedlichen Transistoren. Wer es genau wissen will, braucht eine Transistor-Vergleichstabelle.

# Byte an Byte

**In diesem Kursabschnitt werden wir die Entwicklung eines vollständigen Maschinenprogramms von der Definition der Aufgabe über die Umsetzung in die Assemblersprache bis zur Erstellung des Maschinencodes verfolgen.**

In der letzten Folge wurde dargestellt, wie BASIC-Programmzeilen bei der Eingabe durch Token und ASCII-Codes ersetzt werden. Das zeigte, in welcher Form sich die „Hochsprache" BASIC am Maschinencode orientiert: Sie besteht aus Folgen einzelner Befehle (die durch den Austausch gegen Token nur eine Ebene über der Maschinensprache liegen) und Befehlsdaten. Da die Schlüsselworte und Daten (Variablen, Zahlen oder Strings) der natürlichen Sprache ähnlich sind und einzelne Befehle deutlich durch Zeilennummern bzw. Doppelpunkte voneinander getrennt werden, erscheint uns BASIC weitaus mehr als Hochsprache als der Interpreter. Daraus folgt, daß auch der Maschinencode nur etwas „kosmetische" Aufbereitung braucht, um für uns verständlich zu werden.

Diese „Kosmetik" ist die Assemblersprache, in der mnemotische Kürzel wie LDA und ADC für die Ein-Byte-Op-Codes eingesetzt werden, die der Microprozessor verarbeiten kann. Außerdem können dabei alphanumerische Symbole wie LABEL1 und TFLAG statt Speicheradressen und numerischer Bezeichnungen verwendet werden. Da der Prozessor die Assemblersprache nicht verstehen kann, muß das Programm vor der Ausführung übersetzt werden. Diese Aufgabe übernimmt entweder ein Programm – der Assembler – oder der Programmierer selbst. Der Vorteil der Assemblersprache liegt darin, daß sie eine direkte Übersetzung des Maschinencodes ist. Durch einfache Umsetzung der mnemotischen Kürzel in Op-codes und der Symbole in Zahlen läßt sich lauffähiger Code herstellen. Da die Assemblersprache leichter zu verstehen ist als der Maschinencode, ist sie bei der Programmentwicklung eine große Hilfe. Normalerweise werden Programme in der Assemblersprache entwickelt und nur im letzten Stadium in den Maschinencode übersetzt. Wir werden beide Sprachen behandeln, um die Grundlagen der Programmierung aufzuzeigen.

Ein Microprozessor kann zwar viele unterschiedliche Vorgänge ausführen, bearbeitet im Grunde genommen aber immer nur die Daten seines Speichers. Dabei greift er entweder direkt auf die RAM- oder ROM-Speicher des Computers zu oder setzt seine eigenen internen Speicher ein, die „Register". Register sind Speicherbytes, die sich direkt im Prozessorchip befinden. Sie haben bestimmte Spezialfunktionen, unterscheiden sich sonst aber nicht von anderen Speicherbytes.

## Akkumulator-Register

Das wichtigste Register des Microprozessors ist der Akkumulator. Um ihn einsetzen zu können, müssen wir darin Informationen ablegen – ein Vorgang, den wir „Laden des Akkumulators" nennen. In der Assemblersprache des 6502 führt der Befehl LDA diesen Ladevorgang aus, bei dem Z80 der Befehl LD A. Das Lesen von Informationen aus dem Akkumulator ist ebenso wichtig wie das Laden. Der Assembler des 6502 führt diesen Vorgang mit dem Befehl STA (STore the Accumulator contents) aus. Der Z80 betrachtet Laden und Speichern als einzelne Elemente des gleichen Vorgangs. Das Lesen von Informationen aus dem Akkumulatorregister führt daher ebenfalls der Befehl LD A aus, jedoch mit anderem Format.

Angenommen, Sie möchten ein Assemblerprogramm schreiben, das den Inhalt eines Speicherbytes in das nächste Speicherbyte kopiert: von Byte$09FF in Byte$0A00. In Assembler sieht das folgendermaßen aus:

| **6502** | **Z80** |
|---|---|
| LDA $09FF | LD A,($09FF) |
| STA $0A00 | LD ($0A00),A |

Beachten Sie dabei, daß wir den Inhalt von Byte$09FF in Byte$0A00 kopiert haben, ohne zu wissen, was dieser Inhalt bedeutet. Byte$09FF kann irgendeine Zahl von $00 bis $FF enthalten, unser Programm lädt diese Zahl nur in den Akkumulator und speichert sie vom Akkumulator aus in Byte$0A00. Aus der Assemblerversion des 6502 geht nicht hervor, daß sich LDA auf den Inhalt von Byte$09FF bezieht. Es wird dabei aber eindeutig zwischen dem Ladevorgang (LDA) und dem Speichervorgang (STA) unterschieden. Die Z80-Version hat folgendes Befehlsformat:

OPCODE BESTIMMUNG, (QUELLE)

Dabei sind Speicheradressen in Klammern eingeschlossen, wenn „der Inhalt von" gemeint ist. Mit dieser Methode wird eine äußerst wichtige Unterscheidung zwischen der Adresse eines Bytes und seinem Inhalt vorgenommen.

Da das Assemblerprogramm als Unterrou-

tine eingesetzt werden soll, müssen wir eine Art RETURN integrieren, um die Steuerung an das Ursprungsprogramm zurückzugeben. Der Op-code dieses Befehls für den 6502 ist RTS und der des Z80 RET.

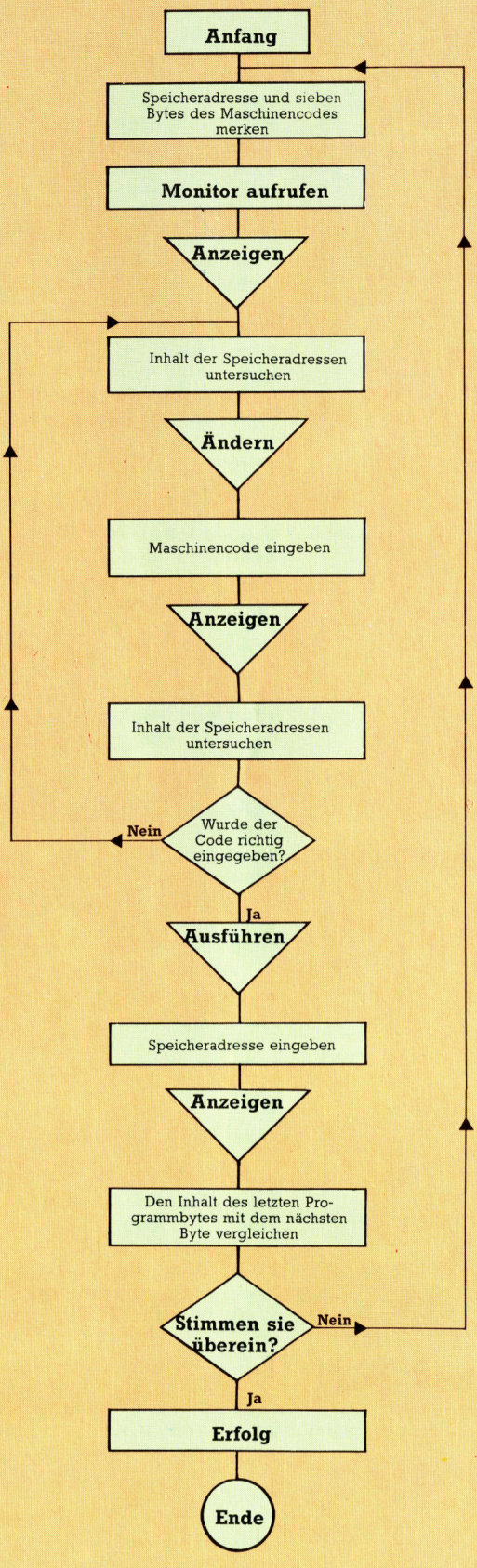
Vor der Ausführung muß die Unterroutine in den Maschinencode übersetzt werden, dann den Code im Speicher unterbringen und schließlich den Microprozessor mit Hilfe des Monitorprogramms veranlassen, das Programm auszuführen. Zuerst müssen wir den Code jedoch übersetzen und entscheiden, an welcher Stelle des Speichers er abgelegt werden soll. Für BASIC-Programmierer ist dies eine ungewöhnliche Überlegung, da sie sich darüber nie Gedanken machen mußten, wo ein BASIC-Programm untergebracht wird – die Entscheidung über den Speicherplatz trafen bereits die Konstrukteure des Systems.

Grundsätzlich kann ein Maschinencodeprogramm an jeder Stelle des Speichers untergebracht und ausgeführt werden. Einige Bereiche eignen sich dafür jedoch besser als andere. Diese Bereiche sind in jeder Maschine verschieden, und daher ergeben sich die folgenden Programmversionen:

| 6502 | | |
|---|---|---|
| **Speicher-adresse** | **Maschinen-code** | **Assembler-sprache** |
| **Commodore 64** | | |
| $0350 | AD 56 03 | LDA $0356 |
| $0353 | 8D 57 03 | STA $0357 |
| $0356 | 60 | RTS |
| **Acorn B** | | |
| $0070 | AD 76 00 | LDA $0076 |
| $0073 | 8D 77 00 | STA $0077 |
| $0076 | 60 | RTS |
| **Z80** | | |
| **16-K-Spectrum** | | |
| $7FA0 | 3A A6 7F | LD A,($7FA6) |
| $7FA3 | 32 A7 7F | LD ($7FA7),A |
| $7FA6 | C9 | RET |
| **48-K-Spectrum** | | |
| $FFA0 | 3A A6 FF | LD A,($FFA6) |
| $FFA3 | 32 A7 FF | LD ($FFA7),A |
| $FFA6 | C9 | RET |

In jeder Version des Programms wird das letzte Byte des Programms in das darauffolgende Speicherbyte kopiert. So kopiert das Programm z. B. auf dem Spectrum mit 48 K den Inhalt von $FFA6 nach $FFA7. Beachten Sie dabei, daß die Adressen der Assemblersprache das hi-lo Format haben, im Maschinencode jedoch im lo-hi Format erscheinen. Interessant ist auch, daß das mnemotische Kürzel des Z80-Befehls im ersten und zweiten Befehl LD lautet, die Op-codes jedoch verschieden sind: 3A lädt Daten in den Akkumulator, während 32 Daten aus dem Akkumulator liest.

## Einsatz des Monitorprogramms




Mit dem auf der nächsten Seite aufgeführten Programm können Sie Speicherinhalte ändern, anzeigen und ausführen lassen. Nach Auswahl einer dieser Möglichkeiten geben Sie die Anfangsadresse im Hexadezimalformat ein, von der an:
**1)** die Bytes verändert werden sollen;
**2)** die Byteinhalte angezeigt werden sollen;
**3)** der Microprozessor das Programm ausführen soll.
Die Eingabe von „X" anstelle einer Hexadezimalzahl läßt das Programm auf die Wahlebene zurückspringen. „Q" beendet das Programm.
Wenn Sie im Änderungsmodus die Adresse angegeben haben, deren Inhalt Sie verändern wollen, wird diese Adresse angezeigt, gefolgt von einem Fragezeichen. Geben Sie jetzt den neuen Inhalt ein.
Nach dem Drücken der RETURN-Taste wird das nächste Speicherbyte in der gleichen Weise angezeigt und kann ebenso geändert werden. Wenn Sie X anstelle einer Adresse eingeben, kehrt das Programm zur Befehlsebene zurück.

**In den Speicher laden**
Sehen Sie sich das Programm für Ihren Computer an, merken Sie sich die erste Speicheradresse und die sieben Bytes Maschinencode. Mit dem Monitorprogramm können Sie diese sieben Hexzahlen in die sieben Speicherbytes laden, die von der angegebenen Speicheradresse aufwärts liegen:
**1)** Rufen Sie das Monitorprogramm mit RUN auf. Lassen Sie sich den Inhalt der ersten Adresse ANZEIGEN.
**2)** Wählen Sie den Befehl ÄNDERN und geben Sie die Speicheradresse und die sieben Bytes des Maschinenprogrammes ein.
**3)** Wählen Sie wieder ANZEIGEN und überprüfen Sie, ob Sie die Adressen und Bytes des Programms richtig eingegeben haben.
**4)** Wählen Sie den Befehl AUSFÜHREN und geben Sie die Anfangsadresse an – es scheint nichts zu passieren.
**5)** Wählen Sie ANZEIGEN und untersuchen Sie die Speicheradressen. Sie werden feststellen, daß der Inhalt des letzten Programmbytes in das darauffolgende Byte kopiert wurde.

# Monitorprogramm für den Spectrum

```
48 REM++++++++++++++++++++++
49 REM+                     +
50 REM+  HCAC MONITOR 1     +
51 REM+ ----SPECTRUM------  +
52 REM+ SAVE THIS PROGRAM   +
53 REM+ BEFORE YOU RUN IT   +
54 REM+                     +
55 REM++++++++++++++++++++++
100 GOSUB 1000  :REM *INIT*
200 CLS
300 PRINT " ** HCAC MONITOR 1 CO
MMANDS **"
400 FOR P=1 TO LT:PRINT O$(P):NE
XT P
500 FOR Z=0 TO 1 STEP 0
550 GOSUB 2000  :REM *INPUT*
600 GOSUB (4500+CM*500)
650 NEXT Z
700 STOP
750 REM++++END MAIN PROG++++
1000 REM*****INIT  S/R******
1050 LET LT=4:DIM C$(LT):DIM O$(
LT,24):DIM X$(16)
1100 LET X$="0123456789ABCDEF"
1150 LET C$="ADGQ":LET C1=-48:LE
T C2=10-CODE(C$(1))
1200 LET O$(1)="       A - ALTER
  MEMORY"
1220 LET O$(2)="       D - DISPLA
Y MEMORY"
1240 LET O$(3)="       G - EXECUT
E M/CODE"
1260 LET O$(4)="       Q - EXIT
 PROGRAM"
1300 RETURN
2000 REM*****INPUT S/R******
2100 FOR P=0 TO 1 STEP 0
2150 PRINT:PRINT"COMMAND ??"
2190 IF INKEY$<>"" THEN GO TO 21
90
2200 LET A$=INKEY$:IF A$="" THEN
 GO TO 2200
2250 FOR J=1 TO LT
2300 IF A$=C$(J) THEN LET CM=J:L
ET J=LT:LET P=2
2350 NEXT J:NEXT P:IF A$="Q" THE
N RETURN
2400 PRINT O$(CM)
2450 FOR P=0 TO 1 STEP 0
2500 INPUT"HEX ADDRESS (X=QUIT)"
;A$
2550 GOSUB 5200 :REM CHK&ADJ
2600 NEXT P:IF A$="X" THEN LET C
M=0
2650 RETURN
3000 REM****HEX BYTE S/R****
3010 LET HB=INT(N/16):LET LB=N-H
B*16
3020 LET B$=X$(HB+1)+X$(LB+1)
3030 RETURN
3100 REM*****D-H   S/R******
3110 IF NM<256 THEN LET N=NM:GOS
UB 3000:LET H$="00"+B$:RETURN
3120 LET HI=INT(NM/256):LET LO=N
M-256*HI
3130 LET N=HI:GOSUB 3000:LET H$=
B$
3140 LET N=LO:GOSUB 3000:LET H$=
H$+B$
3150 RETURN
4000 REM*****H-D   S/R******
4050 LET RX=1:LET DN=0:LET HL=LE
N(H$):IF (HL<1) OR (HL>4) THEN L
ET DN=-1:RETURN
4100 FOR H=HL TO 1 STEP -1
4150 LET D$=H$(H)
4200 LET V=CODE(D$)+C1*(D$>="0"
AND D$<="9") + C2*(D$>="A" AND D
$<="F")
4250 IF V>15 THEN LET DN=-1:LET
H=1:NEXT H:RETURN
4300 LET DN=DN+V*RX:LET RX=RX*16
4350 NEXT H:RETURN
4500 REM*****DUMMY S/R*****
4550 RETURN
5000 REM*****ALTER S/R*****
5020 FOR P=0 TO 1 STEP 0
5040 PRINT A$;:INPUT"NEW HEX VAL
UE (X=EXIT) ?";V$
5050 PRINT V$
5060 GOSUB 5340 :REM CHK&OBY
5080 NEXT P:RETURN
5200 REM*CHECK&ADJUST S/R**
5220 IF A$="X" THEN LET P=2:RETU
RN
5240 LET LL=LEN(A$):IF LL>4 THEN
 RETURN
5260 LET H$=A$:GOSUB 4000
5280 IF DN>=0 THEN LET P=2:LET N
M=DN
5300 LET A$=A$+"  ":IF LL<4 THEN
 LET A$="0000"(TO 4-LL)+A$
5320 RETURN
5340 REM**CHECK & OBEY S/R*
5360 IF V$="X" THEN LET P=2:RETU
RN
5380 LET H$=V$:GOSUB 4000
5400 IF (DN<0) OR (DN>255) THEN
RETURN
5420 POKE NM,DN
5440 LET NM=NM+1:IF NM>65535 THE
N LET P=2:RETURN
5460 GOSUB 3100     :REM D-H S/R
5480 LET A$=H$+"  ":RETURN
5500 REM****DISPLAY S/R*******
5520 FOR P=0 TO 1 STEP 0
5540 INPUT"DEC.NO.OF BYTES(X=QUI
T)";N$:IF N$="X" THEN LET P=2:NE
XT P:RETURN
5560 LET BN=VAL(N$):IF (BN>0) AN
D (BN+NM<65536) THEN LET P=2
5580 NEXT P
5600 FOR B=NM TO (NM+BN-1) STEP
4
5620 LET L$="":LET NM=B:GOSUB 31
00
5640 PRINT H$;TAB(6);
5660 FOR C=0 TO 3
5680 LET N=PEEK(B+C):LET K$="."
5700 GOSUB 3000 :REM D-H S/R
5720 PRINT TAB(6+4*C);B$;
5740 IF N=0 THEN LET K$="■"
5760 IF (N>31) AND (N<128) THEN
LET K$=CHR$(N)
5780 LET L$=L$+K$
5800 NEXT C
5820 PRINT TAB(26);L$
5840 NEXT B:RETURN
6000 REM*****EXECUTE S/R****
6050 RANDOMIZE USR(NM):RETURN
6500 REM*****EXIT  S/R******
6550 PRINT TAB(5);"■■■■END OF PR
OGRAM■■■■"
6600 LET Z=2:RETURN
```

## Acorn B

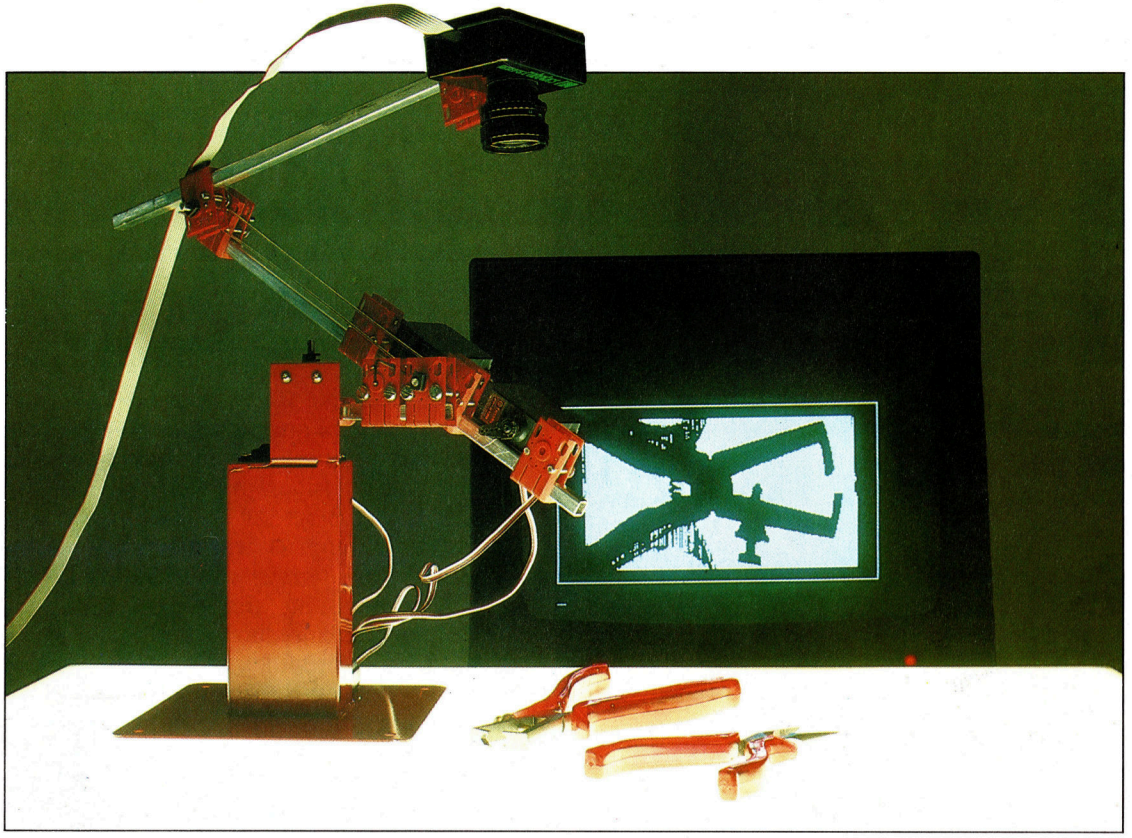
```
39 REM************************
40 REM*    HCAC MONITOR 1     *
41 REM* ---------BBC--------- *
42 REM*   CHANGE THE SPECTRUM  *
43 REM*   VERSION AS FOLLOWS:  *
44 REM*                       *
45 REM*  REPLACE CODE( BY ASC( *
47 REM*                       *
50 REM*  ADD,CHANGE,OR DELETE  *
51 REM*       AS DIRECTED:    *
52 REM*                       *
53 REM************************
60 *TV 255
70 MODE 7
200 PRINT CHR$(147);CHR$(142)
600 ON CM GOSUB 5000,5500,6000,6500
1050 LT=4:DIM C$(LT),O$(LT)
1150 C$(1)="A":C$(2)="D":C$(3)="G":C$(3)
="Q":C1=48:C2=ASC(C$(1))-10
2190 ----------DELETE----------
3020 B$=MID$(X$,HB+1,1)+MID$(X$,LB+1,1)
4150 D$=MID$(H$,H,1)
4500 ----------DELETE----------
4550 ----------DELETE----------
5050 ----------DELETE----------
5300 A$=A$+"  ":IF LL<4 THEN A$=LEFT$("0
000",4-LL)+A$
5420 ?(NM)=DN
5680 N=?(B+C):K$="."
5740 IF N=13 THEN K$=CHR$(255)
6050 CALL NM:RETURN
6600 Z=1:RETURN
```

## Commodore 64

```
49 REM****************************
50 REM*    HCAC MONITOR 1        *
51 REM* ---------CBM--------- *
52 REM*   CHANGE THE SPECTRUM  *
53 REM*   VERSION AS FOLLOWS:  *
54 REM*                       *
55 REM*  REPLACE ALL INSTANCES *
56 REM*  OF:"LET P=2" BY "P=1" *
57 REM*                       *
58 REM*  REPLACE CODE( BY ASC( *
59 REM*                       *
60 REM*  AND CHANGE OR DELETE  *
61 REM*       AS DIRECTED:    *
62 REM*                       *
63 REM****************************
200 PRINT CHR$(147);CHR$(142)
600 ON CM GOSUB 5000,5500,6000,6500
1050 LT=4:DIM C$(LT),O$(LT)
1150 C$(1)="A":C$(2)="D":C$(3)="G":C$(3)
="Q":C1=48:C2=ASC(C$(1))-10
2190 ----------DELETE----------
3020 B$=MID$(X$,HB+1,1)+MID$(X$,LB+1,1)
4150 D$=MID$(H$,H,1)
4500 ----------DELETE----------
4550 ----------DELETE----------
5050 ----------DELETE----------
5300 A$=A$+"  ":IF LL<4 THEN A$=LEFT$("0
000",4-LL)+A$
5740 IF N=0 THEN K$=CHR$(122)
6050 SYS(NM):RETURN
6600 Z=1:RETURN
```

**Mit diesem Programm können Sie sich den Inhalt des Speichers anzeigen lassen, Speicherbytes ändern und ein gespeichertes Maschinencodeprogramm ausführen lassen.**

# Hindernisstrecke



**Dank der fortschreitenden Prozessortechnik werden bewegliche Roboter bald in der Lage sein, einen ganzen Katalog von Grundformen zu erlernen, der in Verbindung mit Umrißerkennung und musterberechnenden Algorithmen zur Anwendung kommt. Der hier gezeigte Roboterarm „Beasty" ist mit einer „Snap"-Kamera ausgestattet, die ein digitales Bild erzeugt, unterstützt durch spezielle Software, zu der ein Objekterkennungs-Modul gehört.**

**Nachdem in den anderen Folgen bereits erläutert wurde, wie sich ein „unintelligentes" berädertes Gefährt bewegen und steuern läßt, befassen wir uns mit dem Konstruktionsprinzipien eines Roboters, der sich wirklich „intelligent" bewegt.**

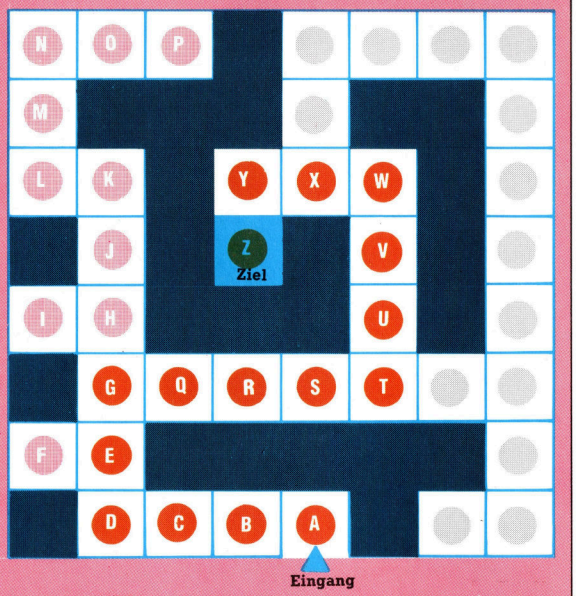
Grundbedingung ist, daß dieser Roboter nicht durch einen menschlichen Operator und auch nicht mit Hilfe gespeicherter Anweisungssequenzen kontrolliert bzw. gesteuert wird. Das Ergebnis wäre sonst lediglich eine Art verbesserter Automat, ein Gerät, das nichts weiter täte, als die vorgegebene Sequenz exakt auszuführen. Trotzdem gibt es auch Einsatzmöglichkeiten für einen derart gebauten Roboter: Roboterarme etwa werden oft als „intelligent" bezeichnet, obwohl sie lediglich eine Abfolge vorprogrammierter Anweisungen ausführen.

Unsere ursprüngliche Idee für die Definition eines „intelligenten" Roboters war die, ein Wesen zu schaffen, das den Frühstückstee ans Bett bringt. Dieser Vorgang kann nicht von einem Menschen kontrolliert werden, da die Aufgabe auszuführen ist, bevor der Mensch erwacht. Wäre dieses Tee bringende Wesen mit einer bestimmten Anweisungssequenz programmiert, ergäben sich nunmehr Probleme, wenn man die Position des Bettes verändern oder ein Kleidungsstück auf den Boden legen würde.

Für die Erforschung künstlicher Intelligenz und das Studium komplexer Probleme bedient man sich Spiele unterschiedlicher Art. Ähnlich wie Schachprogramme beachtliche Einblicke in andere Bereiche künstlicher Intelligenz gegeben haben, kann auch der durch ein Labyrinth laufende Roboter bei der Definition echter intelligenter Bewegung helfen. Ende der siebziger Jahre startete in den USA der erste „Mikromaus"-Wettbewerb. Die Idee war einfach: Man baute ein etwa drei Quadratmeter großes Labyrinth. Die von den Teilnehmern gebauten Roboter-„Mäuse" mußten ohne Hilfe den Weg in die Labyrinthmitte finden. Dieser Irrgarten bestand aus mehreren gleich großen Quadraten, deren Seiten zuweilen offen waren und so einen möglichen Weg zeigten oder in einer Sackgasse endeten. Die Maus, die das Zentrum in kürzester Zeit erreichte, siegte.

Beim ersten britischen Mikromaus-Wettbewerb machten lediglich fünf Teilnehmer mit. Die startenden Mäuse waren recht merkwürdige Konstruktionen. Eine konnte nicht einmal geradeaus laufen, und andere waren völlig durcheinander, nachdem sie um ein paar Ekken gelaufen waren. Im selben Jahr führte man

**Einfach wie das Abc**
Mit einfachen Algorithmen ist das Durchqueren eines Labyrinths möglich. Der Roboter bewegt sich so lange im leeren Raum, bis er in eine Sackgasse gerät, z. B. in die Quadrate F, I oder P. Er kehrt dann auf den zuletzt betretenen Knotenpunkt zurück – in diesem Fall „G" – und löscht alle dazwischenliegenden Quadrate, da diese für sein Weiterkommen nutzlos sind, aus seinem Gedächtnis. Hat er alle von diesem Knotenpunkt ausgehenden Wege bereits ausprobiert, geht er auf den davorliegenden Punkt zurück und so weiter.




## Im Labyrinth

```
49 REM*****CBM 64*********
50 REM*  MAZE SOLVER      *
51 REM*****CBM 64*********
100 GOSUB 2000            :REM INIT
150 GOSUB 9000            :REM PR.MAZE
200 FOR L=1 TO 1
250 GOSUB 7000            :REM LOOKROUND
300 GOSUB 8000            :REM MOVE
400 NEXT L
450 RO=16:CO=12:GOSUB 9500
500 IF MZ<>HM THEN PRINT"BLIND"
550 IF MZ=HM THEN PRINT"HOME"
900 PRINT:PRINT:INPUT A$:RUN
1999 REM**********************
2000 REM*    INIT            *
2001 REM**********************
2100 SZ=10:S2=SZ*SZ:GX=SZ/2:GY=SZ+2
2120 DIM MZ(SZ,SZ),R$(4),LX(4),LY(4)
2140 X=RND(-TI)
2150 DEFFNR(N)=INT(RND(1)*N+1)
2160 HM=-1:G0=-2:WL= 42:W$=CHR$(WL)
2180 CL$=CHR$(147):H$=CHR$(19)
2200 X=GX-1:Y=GY-2:DR=3
2220 D$=CHR$(17):P$=D$
2240 FORK=1TO5:P$=P$+P$:NEXT K:P$=H$+P$
2400 DATA 0,1,"0",-1,0,".",0,-1,"",1,0,""
2420 FOR K=1 TO 4:READ LX(K),LY(K),R$(K)
2490 NEXT K:RETURN
6999 REM**********************
7000 REM*  LOOK AROUND      *
7001 REM**********************
7110 RO=1:CO=1:GOSUB9500
7120 L=0:ND=0:FOR S=1 TO 4
7140 NX=X+LX(S):NY=Y+LY(S)
7200 IF NX<1 OR NX>SZ THEN NEXTS:RETURN
7220 IF NY<1 OR NY>SZ THEN NEXTS:RETURN
7230 MZ=MZ(NX,NY):IF MZ=0 THEN ND=S
7260 IF MZ=HM THEN ND=S:S=4:L=1
7280 NEXT S:RETURN
7999 REM**********************
8000 REM*    MOVE            *
8001 REM**********************
8100 MZ(X,Y)=DR:FL=1
8120 RO=Y+1:CO=X+1:GOSUB 9500
8140 IF ND=0 THEN ND=DR-2-4*(DR<3):FL=2
8160 PRINT R$(FL):X=X+LX(ND):Y=Y+LY(ND)
8180 DR=ND:IF Y>SZ THEN L=1:RETURN
8200 IF FL=2 THEN DR=MZ(X,Y)
8490 RETURN
8999 REM**********************
9000 REM* PRINT THE MAZE    *
9001 REM**********************
9040 PRINT CL$;:RO=1:CO=1
9050 WL=42:W$=CHR$(WL)
9060 S$="                    "
9070 FOR K=1 TO SZ+2:T$=T$+W$:NEXT K
9080 E$=W$+LEFT$(S$,SZ)+W$
9100 PRINT T$:FOR J=2 TO SZ+1
9120 PRINT E$:NEXT J:PRINT T$
9140 FOR K=1 TO S2/2
9150 WX=FNR(SZ):WY=FNR(SZ)
9160 MZ(WX,WY)=WL:RO=WY+1:CO=WX+1
9200 GOSUB 9500:PRINT W$:NEXT K
9250 CO=1+FNR(SZ):RO=1+FNR(SZ):GOSUB 9500
9300 PRINT"H":MZ(CO-1,RO-1)=HM
9350 RO=GY:CO=GX:GOSUB9500:PRINT"^"
9490 RETURN
9499 REM**********************
9500 REM* POSITION THE CRSR *
9501 REM**********************
9600 PRINT LEFT$(P$,RO)TAB(CO-1);:RETURN
```

## BASIC-Dialekte

Fügen Sie folgende Änderungen ein:

### Acorn B

```
49 REM*****BBC***********
50 REM*  MAZE SOLVER      *
51 REM*******BBC**********
9040 CLS:RO=1:CO=1
9600 PRINT TAB(CO-1,RO-1);:RETURN
```

### Spectrum

```
49 REM*****SPECTRUM********
50 REM*  MAZE SOLVER      *
51 REM*****SPECTRUM********
2120 DIM MZ(SZ,SZ):DIM R$(4)
2130 DIM LX(4):DIM LY(4)
2140 RANDOMIZE
2150 DEFFNR(N)=INT(RND*N+1)
9040 CLS:RO=1:CO=1
9600 PRINT AT (RO-1,CO-1);:RETURN
```

das Europäische Finale durch, zu dem sich Teilnehmer und Mäuse aus Finnland, Deutschland und der Schweiz einfanden. Überraschend schaffte es eine Maus auch tatsächlich, ins Zentrum des Labyrinths zu gelangen. Es war Nick Smiths „Stirling Mouse", die mit einfachen mechanischen Sensoren ausgestattet und von einem sehr simplen Schrittmotor angetrieben an den Wänden entlanglief. Seitdem ist das Interesse an dieser Art von Wettbewerben beachtlich gewachsen. Beim Euromouse Wettbewerb 1984 in Madrid schaffte die schnellste Maus in 31,4 Sekunden den Weg ins Zentrum.

## Orientierung im Labyrinth

Wie findet sich nun die Robotmaus in einem Labyrinth zurecht? Prinzipiell muß der Roboter so konstruiert sein, daß er sich exakt bewegen kann und seine Position jederzeit kennt. Dies geschieht, indem man den Roboter mit Rädern ausstattet und durch Schrittmotoren antreibt. Dabei finden die bereits vorgestellten „Shaft Encoder" Anwendung, die eine Roboter-interne Positionsrückmeldung erlauben. Zudem sind Sensoren erforderlich, mit denen der Roboter die Präsenz oder Nichtpräsenz von Wänden feststellen und so eine „Karte" der Hindernisse erstellen kann.

Obwohl die Präzisionsmessungsmethoden der Roboter sehr unterschiedlich sind, haben sie doch eines gemeinsam: An der Stirnseite befindet sich ein einfacher Tastsensor. Steht der Roboter genau in der Mitte eines Quadrates, kann er feststellen, ob sich direkt vor ihm eine Wand befindet. Danach dreht er sich um 90 Grad im Uhrzeigersinn, und der Vorgang wird wiederholt. Bald „weiß" er, wo in jedem Quadrat des Labyrinths die Wände sind. Diese Information kann als einfache Vier-Bit-Binärzahl gespeichert werden. Die Binärzahl 1111 könnte ein Quadrat beschreiben, das vier Wände hat (in der Praxis ist das natürlich unmöglich, da der Roboter erst gar nicht in ein geschlossenes Quadrat hineingelangen könnte). 0000 stellt folglich ein Quadrat ohne Wände dar. 0111 bedeutet ein Quadrat mit drei Wänden und einer Öffnung.

Diese Information läßt sich in BASIC in einem zweidimensionalen Datenfeld unterbringen. DIM A (17,17) zum Beispiel repräsentiert ein Labyrinth mit 17 „Zellen". Der Roboter muß nun einen Weg errechnen, der ihn nach A(8,8) – zum Labyrinthzentrum – führt. Roboter sind häufig mit einem Computerprogramm ausgestattet, das für jede Strecke durch das Labyrinth eine „Baumstruktur" errechnet. Einige Zweige dieses Baumes führen in Sackgassen oder bringen den Roboter an einen bereits besuchten Punkt zurück. In solchen Fällen werden die Zweige „gekappt" und bleiben unberücksichtigt. Das Programm sucht dann in den verbleibenden Zweigen nach der Strecke

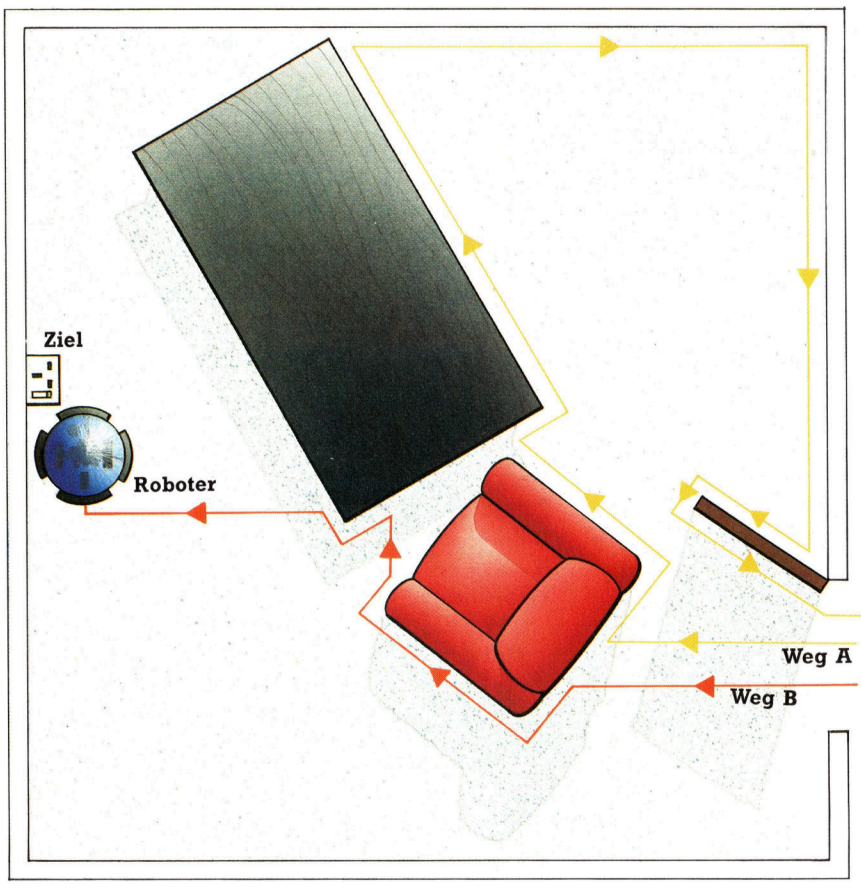
mit den wenigsten Quadraten. Diese wird dann als Weg zum Zentrum ausgewählt.

Mit Hilfe dieser Methode lassen sich wirkungsvolle Strategien ausarbeiten. Entscheidend für die Orientierung des Roboters sind seine Sensoren. Ein einfacher mechanischer Berührungssensor bewirkt, daß der Roboter gegen jede Labyrinthwand laufen müßte. Ein Näherungssensor kann das Vorhandensein einer Wand feststellen, ohne sie berühren zu müssen, wogegen der Abstandssensor selbst eine weit entfernte Wand innerhalb des Labyrinths registrieren kann.

Soll sich der Roboter in einer Wohnung orientieren können, kommen wiederum Sensoren zum Einsatz, mit denen er die exakte Position aller Gegenstände in einem Raum feststellen kann. Daraufhin errechnet er den Weg, der ihn um die Gegenstände herum zu seinem Zielort führt. Zusätzliche Probleme entstehen dadurch, daß ein Raum vielfältiger als ein Labyrinth ausgestattet ist. Ein typisches Zimmer ist nicht in Quadrate aufgeteilt. Ebensowenig bleiben die in ihm befindlichen Gegenstände stets an ein und derselben Stelle. Unser Roboter kann zwar die Position der einzelnen Gegenstände registrieren und somit lernen, doch sobald ein Stuhl bewegt wird oder etwa eine Katze auf dem Boden sitzt, muß der errechnete Weg modifiziert werden.

Dieses Problem ist nur dadurch zu lösen, daß der Roboter seine Sensoren ständig benutzt, um die interne Karte stets zu aktualisieren. Zur Lösung des Problems „Katze" ist mehr Nachdenken erforderlich. Da Roboter nichts über Katzen wissen (ebensowenig natürlich über Menschen), ist es für ihn schwer, bei der ersten Begegnung mit Wesen der Katzenart eine Entscheidung zu treffen. (Umgekehrt wird es der Katze bei ihrer ersten Begegnung mit einem Roboter ähnlich gehen!) Die beste Lösung wäre, den Roboter mit einem Bewegungssensor auszustatten. Hierbei handelt es sich um einen Abstandssensor, der auf verschiedene Entfernungsmessungen reagiert und so bewegte Objekte registriert. Ist der Gegenstand als solcher erkannt, wäre die richtige Schlußfolgerung für den Roboter, so lange stehenzubleiben, bis der Gegenstand sich nicht mehr bewegt oder fort ist. Das mag nicht sehr intelligent klingen und ist sicherlich weniger freundlich, als zur Katze zu gehen und sie zu streicheln, doch diese Handlungsweise ist der Reaktion ähnlich, die viele Tiere zeigen, wenn sie bewegte Objekte sehen, sie „erstarren", verhalten sich regungslos.

Der ganze Komplex intelligenter Bewegung ist also tatsächlich mit dem Einsatz von Sensoren, gekoppelt mit einem entsprechenden Computerprogramm, verbunden. Ein Roboter ohne Sensoren kann sich nicht intelligent bewegen. Mit je mehr Sensoren ein Roboter ausgestattet ist, desto besser wird seine Kenntnis über die Umgebung sein, was ihn wiederum befähigt, Ansätze von „künstlicher" Intelligenz zu zeigen.




**Leicht ist es nie, den Weg um unbekannte Objekte zu finden. Weg A zeigt die Strecke, die ein einfacher Heimroboter gehen muß, um an die Steckdose zu gelangen. Sein Objekt-Ausweich-Algorithmus weist ihn an, den Flächen der Gegenstände zu folgen, auch um Ekken herum, wobei seine tastenden Sensoren nach dem Gegenstand suchen. Diese einfache Methode ist bei unkomplizierten Strecken wirkungsvoll, erweist sich aber als nachteilig bei einer Fülle von Hindernissen.**

**Weg B zeigt, wie sich ein ähnlicher Roboter mit geringfügig modifiziertem Algorithmus verhält: Sobald er einen Gegenstand erreicht hat, dreht er sich im kleinstmöglichen Winkel, da er auf diese Weise weniger häufig „zurückgehen" muß. Diese Verbesserung macht ihn weniger störanfällig. Seine Tast-Sensoren können seine Bewegungen wirkungsvoller steuern.**

# Wer war der Mörder?

**In diesem Teil des LOGO-Kurses wird die Listenverarbeitung anhand eines weiteren Beispiels dargestellt – die Untersuchung in einem Mordfall, in dem es den Täter zu finden gilt.**

Das kleine Städtchen am Fuße des Ozark-Gebirges wurde durch einen schrecklichen Mord erschüttert. Irgend jemand hatte Zacharias hinterrücks mit einer Axt angegriffen und tödlich verwundet. Es ist bekannt, daß Mattias und Josua Äxte besitzen, Johann und Hannes verfügen über Gewehre, und Cousine Diane hat ein Messer. Bei der polizeilichen Vernehmung stellt sich heraus, daß Mattias und Johann Blut an ihren Händen haben.

Die Informationen für unsere Krimi-Datei werden in Faktenlisten zusammengefaßt – jede besteht aus einer Relation in Verbindung mit mindestens einem Substantiv. In der LOGO-Schreibweise sieht das etwa so aus: [BESITZT MATTIAS AXT] oder zu gut deutsch: Mattias besitzt eine Axt und [BLUTIGER MATTIAS] um festzuhalten, daß Mattias Blut an seinen Händen hat. Die Ermittlung beginnt mit einer leeren Datei:

```
TO ANFANG
  MAKE "DATEI []
END
```

Anschließend werden die neuen Sachverhalte der Datei zugeführt. Zum Beispiel mit der Eingabe ZUF [BESITZT DIANE MESSER].

```
TO ZUF :FAKTEN
  IF NOT MEMBER? :FAKTEN :DATEI
    THEN MAKE "DATEI FPUT :FAKTEN
    :DATEI
END
```

Mögliche Eingaben wären:

```
[[BLUTIGER MATTIAS] [BLUTIGER JOHANN]
[TOETETE ZACHARIAS AXT] [BESITZT
MATTIAS AXT] [BESITZT JOSUA AXT]
[BESITZT JOHANN GEWEHR] [BESITZT
HANNES GEWEHR] [BESITZT DIANE
MESSER]]
```

Die Prozedur ZEIGEN stellt den Inhalt der Datei dar. In Verbindung mit "ALL werden alle Daten aufgelistet und bei Angabe der Relation nur die mit diesem Begriff in Verbindung stehenden Fakten.

```
TO ZEIGEN :S
  IF :S = "ALL THEN LIST.ALL :DATEI
  LIST:REL :S :DATEI
END
TO LIST.ALL :LIST
  IF EMPTY? :LIST THEN STOP
  PRINT FIRST :LIST
  LIST.ALL BUTFIRST :LIST
END
TO LIST.REL :S :LIST
  IF EMPTY? :LIST THEN STOP
  IF :S = FIRST FIRST :LIST THEN PRINT
    FIRST :LIST
  LIST.REL :S BUTFIRST :LIST
END
```

Nun muß eine Möglichkeit gefunden werden, wie die Informationen in der Datei abzufragen und gleichzeitig auf den Wahrheitsgehalt zu überprüfen sind. Die Prozedur RICHTIG gibt zum Beispiel bei RICHTIG [BESITZT DIANE MESSER] als Antwort JA aus.

```
TO RICHTIG :FAKTEN
  IF MEMBER? :FAKTEN :DATEI PRINT
    "JA ELSE PRINT "NEIN
END
```

Besser wäre es natürlich, wenn man konkrete Fragen wie „Wer besitzt eine Axt?" stellen könnte. Das läßt sich mit dem Einsatz von Variablen erreichen, die durch ein Fragezeichen gekennzeichnet werden, z. B.:

```
WER [BESITZT ?JEMAND AXT]
```

Die Antwort ist eine Liste der Werte, die der Variablen ?JEMAND zugeordnet sind.

```
[?JEMAND MATTIAS]
[?JEMAND JOSUA]
KEINE (WEITEREN) ANTWORTEN
```

Man kann mehrere Variablen einsetzen. Zum Beispiel würde

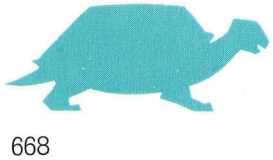
```
WER [TOETETE ?PERSON ?WAFFE]
```

als Antwort

```
[?PERSON ZACHARIAS] [?WAFFE AXT]
KEINE (WEITEREN) ANTWORTEN
```

ergeben. Wir brauchen jedoch Prozeduren, mit denen alle Daten abgefragt werden. WER ruft FINDEN auf und befragt die Routine DATEI, in der sich die Sachverhalte befinden.

```
TO WER :ZWEIFEL
  FINDEN :ZWEIFEL :DATEI
```



## Mörder-Matrix

Der alte Herr Rose wurde tot hinter einem Lieferwagen aufgefunden – ermordet durch 30 Stiche mit einem spitzen Gegenstand. Die Polizei identifizierte vier Verdächtige: einen Maurer, einen Metzger, einen Gärtner und einen Grafiker. Jeder von ihnen hat ständigen Zugang zu scharfen Werkzeugen wie Meißel, Gartenschere, Fleischermesser, Gartenschere und Federmesser. Die Alibis der Personen: Einer wurde auf der Straße gesehen, der zweite arbeitete während der Tatzeit im Garten, und ein anderer gab vor, im Bett gelegen zu haben. Derjenige, der mit dem Lieferwagen in Verbindung gebracht werden kann, muß also der Mörder sein. Die polizeilichen Ermittlungen ergaben folgende Fakten: 1. Niemand gibt zu, im Besitz der Mordwaffe zu sein. 2. Man entdeckte den Metzger beim Öffnen eines Briefes – mit einem Federmesser. 3. Ein Augenzeuge bestätigte, den Maurer auf der Straße gesehen zu haben. Und zwar genau an der Stelle, wo der Gärtner später seine Gartenschere wiederfand. 4. Der Gärtner benutzte ein Fleischermesser, um einen Braten zuzubereiten.

```
    PRINT [KEINE (WEITEREN) ANTWORTEN]
  END
```

In der Prozedur FINDEN werden die globalen Variablen VARS und ANT verwendet. VARS enthält den Inhalt der jeweils befragten Variablen, die in der Liste ANT zugesammengefaßt sind.

```
  TO FINDEN :FRAGEN :DATEN
    MAKE "VARS []
    MAKE "ANT []
    VERGLEICHE :ZWEIFEL :DATEN
    PRINTL :ANT
  END
```

Die Prozedur VERGLEICHE ruft alle Daten auf und prüft, ob Übereinstimmungen vorhanden sind. Ist die Abfrage positiv, werden die entsprechenden Daten aus VARS der Liste ANT zugefügt.

```
  TO VERGLEICHE :ZWEIFEL :DATEN
    IF EMPTY? :DATEN THEN STOP
    IF GLEICH? :ZWEIFEL FIRST :DATEN
      THEN MAKE "ANT FPUT :VARS :ANT
    MAKE "VARS[]
    VERGLEICHE :ZWEIFEL BUTFIRST
      :DATEN
  END
```

Bei den Eingaben [BESITZT ?JEMAND AXT] und [BESITZT JOSUA AXT] zum Beispiel ist das Ergebnis des Vergleiches positiv, so daß GLEICH? als Ausgabe TRUE produziert, und [?JEMAND JOSUA] wird VARS zugewiesen. Dagegen ergibt GLEICH? bei [BESITZT ?JEMAND AXT] und [TOETETE ZACHARIAS AXT] als Resultat FALSE.

In der nächsten Prozedur wird der Befehl TEST zum Überprüfen von Bedingungen eingesetzt. Ist das Ergebnis positiv, werden die Anweisungen nach IFTRUE ausgeführt, andernfalls ruft das Programm die Befehle nach IFFALSE auf.

```
  TO GLEICH? :ZWEIFEL :FAKTEN
    IF ALLOF EMPTY? :ZWEIFEL EMPTY?
      :FAKTEN THEN OUTPUT "TRUE
    TEST FIRST FIRST :ZWEIFEL = "?
    IFTRUE IF NOT WERT? FIRST :ZWEIFEL
      FIRST :FAKTEN :VARS THEN OUTPUT
      "FALSE
    IFFALSE IF NOT (FIRST :ZWEIFEL =
      FIRST :FAKTEN) THEN OUTPUT
      "FALSE
    OUTPUT GLEICH? BUTFIRST :ZWEIFEL
      BUTFIRST :FAKTEN
  END
```

Die Arbeitsweise von WERT? läßt sich am besten anhand eines Beispiels demonstrieren. Bei den Eingaben ?WAFFE, AXT und [?PERSON ZACHARIAS] versucht WERT? zu ermitteln, ob die Variable ?WAFFE den Wert AXT enthält. Hier gibt es drei Möglichkeiten: ?WAFFE wurde bereits ein Wert zugewiesen, der jedoch nicht AXT lautet, worauf ?WERT das Ergebnis FALSE ausgibt; ?WAFFE enthält den Wert AXT, das Ergebnis ist demnach TRUE; ?WAFFE trägt noch keinen Wert, so daß sie jetzt mit AXT definiert und die Information VARS zugeführt wird. Das Ergebnis der Abfrage lautet dann TRUE.

```
TO WERT? :NAME :WERT :VLIST
  IF EMPTY? :VLIST THEN MAKE "VARS
    LPUT LIST :NAME :WERT :VARS
    OUTPUT "TRUE
  TEST :NAME = FIRST FIRST :VLIST
  IFTRUE IF :WERT = LAST FIRST :VLIST
    THEN OUTPUT "TRUE ELSE OUTPUT
    "FALSE
  OUTPUT WERT? :NAME :WERT
    BUTFIRST :VLIST
END
```

Die Prozedur PRINTL veranlaßt, daß die Informationen aus ANT untereinander ausgegeben werden.

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

Mit Hilfe dieser Prozeduren kann die Polizei bei ihren Ermittlungen jedoch keine Erfolge erzielen. Es sollten Fragen gestellt werden können wie: „Durch welche Waffe wurde Zacharias getötet, und wer besitzt eine solche Waffe?" In LOGO sieht das so aus:

```
WER [[TOETETE ZACHARIAS ?WAFFE]
  [BESITZT ?VERDACHT ?WAFFE]]
```

WER überprüft die einzelnen Listen, und die gefundenen Werte können nun als Tatsachen betrachtet werden. Die folgende Frageform gestattet die Abfrage von einzelnen Faktoren:

```
WER [[BESITZT ?EIN MESSER]]
```

Die Prozeduren müssen jetzt noch geringfügig modifiziert werden:

```
TO WER :FRAGEN
  FINDEN :FRAGEN :DATEI
  PRINT [KEINE (WEITEREN)
    ANTWORTEN]
END

TO FINDEN :FRAGEN :DATEN
  MAKE "VARS[]
  MAKE "ANT []
  VERGLEICHE :FRAGEN :DATEN
  PRINTL :ANT
END
```

Die Aufgabe der Prozedur VERGLEICHE ist relativ umfangreich. Bei den Eingaben [[TOETETE ZACHARIAS ?WAFFE] [BESITZT ?VERDACHT ?WAFFE]] zum Beispiel überprüft VERGLEICHE das erste Element mit sämtlichen Informationen auf eine eventuelle Übereinstimmung. Dabei wird festgestellt, daß ?WAFFE und AXT zusammengehören. Anschließend beschäftigt sich die Vergleichs-Prozedur, wiederum indem die gesamten Daten verglichen werden, mit der Eingabe ([BESITZT ?VERDACHT ?WAFFE]). Da auch bei der zweiten Bedingung eine Übereinstimmung gefunden wurde, enthält die Variable ?WAFFE jetzt den Wert AXT, und die Variable ?VERDACHT beinhaltet MATTIAS.

Mit einer zusätzlichen Prozedur kann herausgefunden werden, wie der momentane, durch VERGLEICHE ermittelte Wert der Variablen lautet, bevor an GLEICH? (GLEICH? ändert unter bestimmten Voraussetzungen die Variablenwerte) übergeben wird.

```
TO VERGLEICHE :FRAGEN :DATEN
  IF EMPTY? :FRAGEN THEN MAKE "ANT
    FPUT :VARS :ANT STOP
  IF EMPTY? :DATEN THEN STOP
  LEGEN :VARS
  TEST GLEICH? FIRST :FRAGEN FIRST
    :DATEN
  IFTRUE VERGLEICHE BUTFIRST
    :FRAGEN :DATEI
  NEHMEN "VARS
  VERGLEICHE :FRAGEN BUTFIRST
    :DATEN
END
```

Bei VERGLEICHE wird ein Stack (in diesem Fall ist ein Speicher für bestimmte Variablenwerte gemeint) für die Variable VARS benutzt, um ihren gegenwärtigen Wert ermitteln zu können. Gleichzeitig verhindert der Stack das Überschreiben des ursprünglichen Wertes, wenn VERGLEICHE erneut aufgerufen wird.

Der Befehl LEGEN sorgt dafür, daß der Variablenwert auf den Stack geschoben wird.

```
TO LEGEN :DATEN
  IF NOT THING? :STACK THEN MAKE
    "STACK []
  MAKE "STACK FPUT :DATEN :STACK
END
```

Die Anweisung NEHMEN holt ein Element vom Stack und weist es der Variablen zu.

```
TO NEHMEN :NAME
  MAKE "NAME FIRST :STACK
  MAKE "STACK BUTFIRST :STACK
END
```

Auf diesen Grundlagen läßt sich ein komplexes Programm entwickeln, bei dem Informationen in einer Datei abgelegt und die Such- bzw. Abfragekriterien, mit denen einzelne Elemente aufgezeigt werden sollen, selbst zu definieren sind.

## LOGO-Dialekte

In einigen LOGO-Versionen gibt es die Befehle EMPTY? und MEMBER? nicht. Die jeweiligen Erklärungen wurden bereits dargelegt. Ersetzen Sie gegebenenfalls EMPTY? durch EMPTYP und MEMBER? durch MEMBERP. Ferner gibt es bei einigen Versionen den Befehl EQUALP, der überprüft, ob seine beiden Eingaben identisch sind. Man verwendet ihn statt =, um Listen und Worte zu vergleichen. Im folgenden Beispiel wird die variierte Schreibweise des IF-Befehls gezeigt:

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

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

Beim LCSI-LOGO können die Befehle TEST, IFTRUE und IF-FALSE für Bedingungsabfragen eingesetzt werden.

# Software-Verkauf

**Imagine Software war einer jener neuen Produzenten, die aufgrund der Entwicklung im Heimcomputerbereich enorme Erfolge verzeichnen konnten. In der Blütezeit arbeiteten weit über hundert Leute für das Unternehmen.**

Wie alle erfolgreichen Gesellschaften erlangte Imagine deshalb eine Spitzenposition, weil talentierte Leute aus verschiedensten Bereichen gemeinsam miteinander arbeiteten. Die Gründer, David Lawson und Mark Butler, beide aus Liverpool stammend, repräsentierten die zwei grundlegenden Elemente, die für das Gelingen jedes Unternehmens entscheidend sind: technische Erfahrung und ausgeprägten Geschäftssinn.

Imagine war auf die Herstellung von Programmen für den Commodore 64 spezialisiert. So erklärte sich auch, warum gleich vier verschiedene Programme, nämlich „Betwiched", „Catcha Snatcha", „Wacky Waits" und der Longseller „Arcadia" gleichzeitig in den Top Ten der Software-Hitparade standen. Aufgrund von Imagines Commodore-Engagement lud der Hardware-Anbieter das Software-Haus nach Norriston, Pennsylvania, ein, damit das Unternehmen sich vorab mit der Technik der Typen 264 und V364 vertraut machen konnte. Das war 1983. Die damals präsentierten Maschinen wurden speziell für Computerspielfreunde entwickelt.

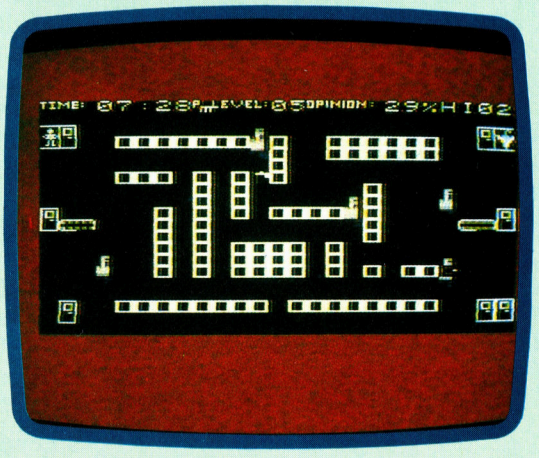
Commodore erwarb daraufhin die Rechte an zwei von Imagine entwickelten Spielen, die unter dem eigenen Markennamen vertrieben werden. Eine besondere Auszeichnung für ein so junges Unternehmen wie Imagine.

Um auch Programmierer für andere Prozessoren wie den 68000 zu finden, startete Imagine eine Anzeigenkampagne. Das Ergebnis indes war enttäuschend. Obwohl die Rücklaufquote geradezu gigantisch war, gab es nur wenige Interessenten, die mehr als ein paar Monate Erfahrung mit dem Prozessor hatten. Viele von ihnen waren nicht einmal imstande, die Maschinenprogrammierung eines einfachen Spieles fertigzustellen. Kaum eine Basis, um völlig neue Projekte erfolgreich anzugehen!

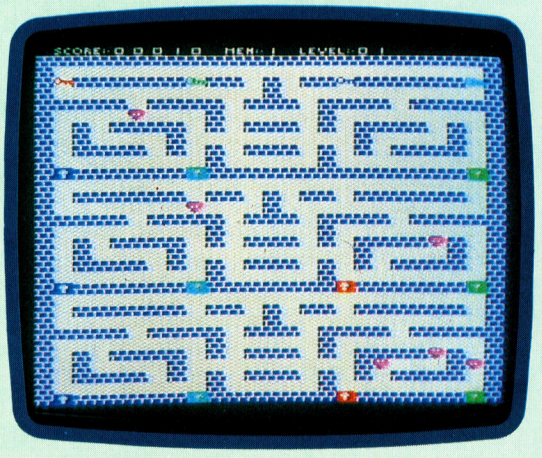
Danach schien Imagine seine Aktivitäten in eine andere, lukrativere Richtung auszuweiten, nämlich in den Geschäftssoftware-Bereich. Geplant war das im Grunde nicht. Die Initiative dazu ging vielmehr von Apple aus. Imagine erkannte bald, an welcher Software Apple interessiert war, nämlich an Programmen mit anwenderfreundlicher Bildschirmführung und Bildelementen wie etwa „Windows" (also „Fenstern") und Symbolen (oder „Piktogrammen"), die über die Maus gesteuert werden können.

Imagine war mit Apples Acht-Bit-Technologie bestens vertraut, da der Apple IIe in diesem Unternehmen bereits zur Programmerstellung verwendet wurde. Daraus ergaben sich natürlich Vorteile für diese Verbindung. Imagine hatte zur Verbesserung des Equipments mehrere Sage-IV-Rechner erworben, um damit die Leistungsfähigkeit, aber auch die Verar-

**Catcha Snatcha**
Dieses Programm wurde für den VC 20 geschrieben. Ein Labyrinthspiel, in dem der Spieler in die Rolle eines Warenhausdetektives schlüpft, der Diebe aufspüren muß, verlorene Gegenstände zu finden hat und Kindern hilft, zu ihren Eltern zurückzukommen.



**Bewitched**
Auch hier handelt es sich um ein Labyrinthspiel. Es unterscheidet sich von herkömmlichen Versionen dieser Art dadurch, daß Teile des Labyrinthes erst nach dem Öffnen von Türen zugänglich sind. Die Sache wird kompliziert, weil sich nicht alle Türen öffnen lassen.



**Der Erfolg eines jeden Software-Hauses hängt vom Einfallsreichtum seiner Programmierer ab. Das Foto zeigt die Personen (von links nach rechts), die die Spiele „Psyclapse" und „Bandersnatch" entwickelt haben: Ian Weatherburn, Mike Glover, John Gibson und Eugene Evans.**

beitungsgeschwindigkeit der Rechner zu erhöhen. Dave Lawson schrieb Software, die zur Erstellung von Spiel-Programmen verwendet wird, auf den Sage-IV-Rechnern, die dann wiederum für den jeweiligen Computer kompiliert werden konnte. Was bedeutet: Die Programme werden in einem Rechner für die Anwendung in einem anderen Rechner modifiziert. Der Sage IV verfügt über eine Diskettenstation mit großer Speicherkapazität, die beim Programmschreiben in Maschinensprache von unschätzbarem Wert ist. Es ist wirtschaftlicher für ein Softwareunternehmen, in einen Rechner wie den Sage IV zu investieren und somit den Zeitaufwand fürs Programmieren zu reduzieren.

## Auch für Universitäten

Auch die Universität von Kalifornien in San Diego verfügt über ein Sage-System. Dies wiederum arbeitet mit dem sogenannten p-System, einer PASCAL-Implementierung, mit der Apples Lisa- und Macintosh-Techniken simuliert werden können. Die Bezeichnungen „Background" und „Foreground" (frei: „Hintergrund" bzw. „Vordergrund") beziehen sich auf Maschinen, die mehrere Programme gleichzeitig verarbeiten können. Dabei hat das „Vordergrund"- oder „Hauptprogramm" immer Priorität.

Wie so viele andere Software-Häuser beschäftigte sich auch Imagine mit einer der vielseitigsten Programmiersprachen, die unter der Bezeichnung „C"-Code bekannt ist. Aufgrund ihrer modularen Struktur ist sie ideal für die Entwicklung von System-Software.

Wodurch wird überhaupt Software zu einem Erfolg, einem Hit? Das berühmte „richtige Gefühl" hat damit eine Menge zu tun! Zu Beginn ihrer selbständigen Arbeit waren sich Mark Butler und Dave Lawson sehr wohl dessen bewußt, was den Marketing-Erfolg ihres ehemaligen Arbeitgebers Bug-Byte ausgemacht hatte. Sie fanden sich zu „Brainstormings" zusammen, versuchten sich in die Situation begeisterter Spieler zu versetzen. Überlegungen dieses Entwicklungsprozesses ließen sich wie folgt formulieren: „Wenn das mein erster Computer wäre, was würde ich spielen wollen?" Und: „Wie lange würde ich damit spielen?" Oder: „Würden mir mehr die verschiedenen Soundeffekte zusagen oder ein grafischer Höhepunkt in diesem oder jenem Spielstadium?" Der Spiel-Programmierer muß sich also in den potentiellen Spieler hineinversetzen und dabei dessen Alter berücksichtigen.

## Schnelle Ausweitung der Firma

Die Entwicklung neuer Ideen zog nach sich, daß bei Imagine bald acht Grafiker mit der Animation und der Drehbuchgestaltung neuer Programme befaßt waren. Deren Arbeiten wurden von einem hausinternen Test-Team geprüft, wobei es sich nicht um Programmierer handelte, da es in diesem Stadium noch nicht um technische Perfektion oder Details ging.

Zum Nachteil der Software-Häuser haben zu viele Anwender eine Technik erlernt, nämlich die Programme illegal zu vervielfältigen. Man schätzt, daß auf eine verkaufte Programmcassette mindestens sieben Raubkopien kommen. Eine Reihe von Versuchen zur Unterbindung des Kopierens (durch Kopierschutz etc.) erwiesen sich als uneffektiv, nicht zuletzt unter dem Gesichtspunkt, daß die daraus entstehenden Kosten an den Käufer weitergegeben werden müssen und so dem Kopieren noch weiter Vorschub geleistet wird. Auf diese Weise entstand auch bei Imagine ein beachtliches Umsatzminus.

Es ist damit zu rechnen, daß Cartridge und Diskette die Cassette als Datenträger bald völlig ersetzen werden. Dazu kommen neue Möglichkeiten des Software-Vertriebs, so zum Beispiel, Programme über einen zentralen Großrechner direkt in den Heimcomputer zu überspielen, etwa durch einen Akustikkoppler oder über Leitungen, die auch beim Kabelfernsehen genutzt werden. Ein Software-Haus, das zukunftsorientiert arbeiten will, muß sich dieser neuen Technologien bedienen und auch Entwicklungen der Spielqualität selbst berücksichtigen, so z. B. die Integration der Stimmerzeugung oder das Einbeziehen der Laserbildplatte.

Inzwischen ist das Kapitel „Imagine" beendet. Die erfolgreiche Firma mußte Konkurs anmelden. Software-Philosophie und Qualität, die mit dem Namen verbunden waren, erlaubten jedoch einen Neubeginn. „Ocean-Software" erwarb 1985 alle Rechte am Namen „Imagine" und setzt nun mit überzeugenden Programmen die leider nur kurze Tradition des berühmten Namens fort.

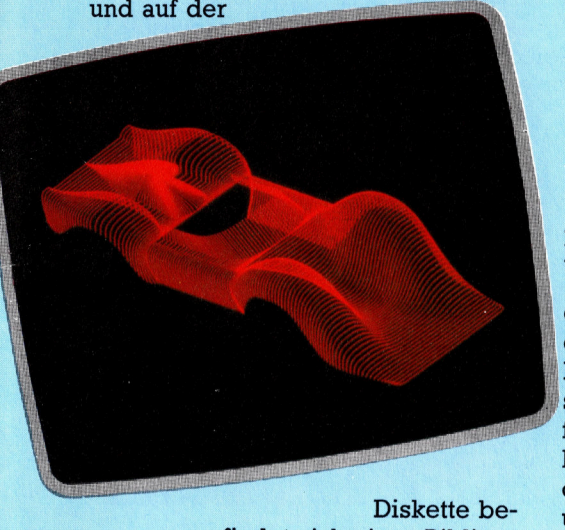
# Fachwörter von A bis Z

**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.**

## CAD/CAM = CAD/CAM

Die Begriffe CAD und CAM (Computer Aided Design/Manufacturing) für rechnergestützten Entwurf bzw. rechnergestützte Fertigung charakterisieren gemeinsam ein modernes Industriekonzept. „Entwurf" heißt in diesem Zusammenhang: ingenieurmäßige und nicht künstlerische Gestaltung – obwohl Computer jetzt auch in diesem Bereich erfolgreich eingesetzt werden.

Der Rechnereinsatz in der Konstruktion bietet wesentliche Vorteile – zunächst einmal bei der bildlichen Darstellung des Entwicklungsobjekts. CAD-Systeme arbeiten meist mit einem hochauflösenden Monitor sowie einem farbfähigen Plotter. Die zugehörige Software ist praktisch das grafische Gegenstück zu einem Textverarbeitungspaket: Bildelemente können korrigiert, gedreht oder gelöscht werden, und auf der



Diskette befindet sich eine „Bibliothek" für Standardbauteile. Bei aufwendigeren Systemen ist auch die räumliche Darstellung des Entwurfs möglich, der aus verschiedenen Blickwinkeln betrachtet werden kann.

Weiterhin kann der Computer mit Hintergrund-Programmen den Entwickler von langwierigen, aber unentbehrlichen Berechnungen entlasten, wie zum Beispiel die Bestimmung der aus Festigkeitsgründen erforderlichen Materialquerschnitte oder die Erstellung von Formeln und Funktionen für die in der Zeichnung benötigten Kurven.

Schon im Vorfeld der Konstruktion kann der Rechner die Suche nach der optimalen Lösung unter gegebenen Randbedingungen (Gewicht, Größe, Stabilität und Kosten) wirksam unterstützen, indem alle Möglichkeiten durchkalkuliert werden und so das optimale Ergebnis ermittelt wird.

Von großem Vorteil ist ferner, daß mit der Vollendung des Entwurfs alle Konstruktionsdaten gespeichert vorliegen und unmittelbar an CAM-Anlagen weitergegeben werden können.

Der Begriff CAM (rechnergesteuerte Fertigung) umfaßt eigentlich den gesamten Bereich der Robotics. In der Industrie geht der Trend von starren Fertigungsstraßen weg zu flexiblen Produktionsanlagen (Flexible Manufacturing Systems = FMS), die leicht umzuprogrammieren sind und eine bessere Anpassung der Fertigung an die Marktsituation ermöglichen. Dabei werden an den Fließbändern CNC-Werkzeugmaschinen (Computer Numerically Controlled = rechnergesteuert) eingesetzt, die bei Produktionsänderungen softwaremäßig „umgerüstet" werden können.

## CAI/CAL = CAI/CAL

Hier geht es um Computer und Lernprozesse – ein umstrittenes Thema. Selbst unter den Befürwortern des Rechners in Schulen gibt es noch Debatten darüber, ob der Computer besser als „Lehrhilfe" (CAI) oder als „Lernhilfe" (CAL) zu gebrauchen sei.

CAI (Computer Aided Instruction) ist die rechnergestützte „Unterweisung" im Stil des traditionellen Lehrens. Der Computer ist zugleich elektronisches Lehrbuch und rechnergesteuerte Lehrkraft. Im Rahmen eines Lehrprogramms wird der Stoff abschnittweise anhand von Grafiken, Trickfilmszenen und Sound angeboten. Das ist viel interessanter als das klassische Lehrbuch oder die Tafelmethode. Nach jedem Abschnitt versucht der Rechner (üblicherweise in Form von Multiple-Choice-Fragen), den Lernerfolg festzustellen. Das Programm ermöglicht dem Schüler, die individuelle Lerngeschwindigkeit selbst zu bestimmen.

Beim CAL (Computer Aided Learning) steht dagegen das „Lernen aus eigenem Antrieb" im Vordergrund, wovon man in der Vergangenheit aufgrund fehlender Voraussetzungen kaum gezielt Gebrauch machen konnte. Der Computer simuliert eine Umgebung, die zum „spielerischen" Lernen anregt.

Zweifellos wird CAL weit eher als CAI der kindlichen Auffassungsgabe gerecht. Das CAL-Konzept erfordert allerdings weit mehr Phantasie und auch raffiniertere Programme.

## CALL = CALL

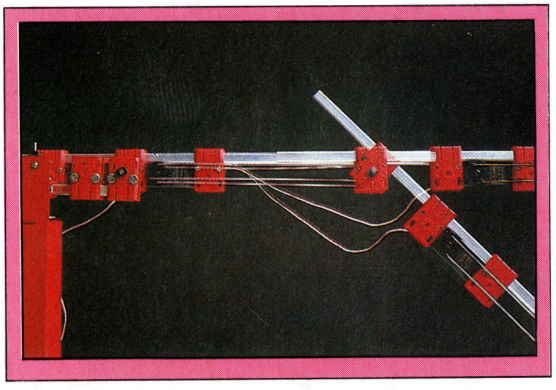
Der CALL-Befehl veranlaßt den Rechner, ein Unterprogramm aufzurufen. Wenn Sie diese Anweisung nicht in der Befehlsliste Ihres Rechners finden, gibt es dafür bestimmt etwas Gleichwertiges, beispielsweise GOSUB, PROC, SYS oder USR. Oft hat der CALL-Befehl auch die Bedeutung, daß der folgende Programmteil in der Maschinensprache geschrieben ist. Das entspricht ebenfalls dem Aufruf eines Unterprogramms (im Maschinencode).


**Bildnachweise**

645, 646, 647: Chris Stevens
648, 649, 652, 661, 667: Kevin Jones
650: Janos Marffy
651, 665: Ian McKinnell
660: Liz Heaney
666: Brian Morris
670: David Higham
671: Soft
672: Imagine
U3: Integraph

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

# computer kurs Heft 25



## Roboter-Arme

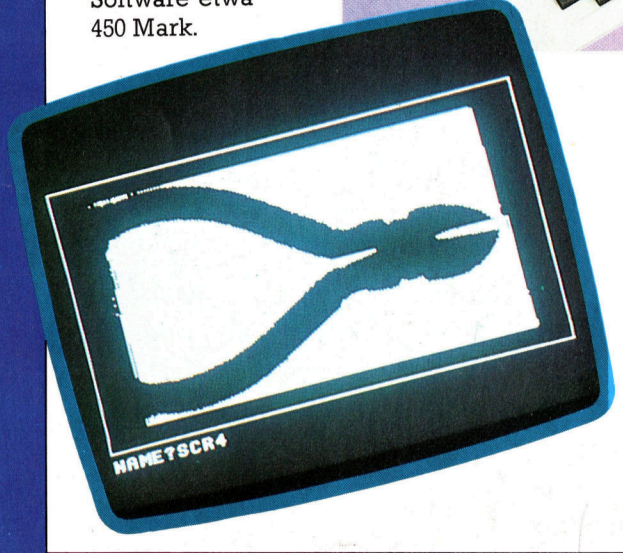
können auch per Heimcomputer gesteuert werden. Das Modell „Beasty“ gibt es als Bausatz, die Software wird als Cassette geliefert.


## Handheld-Nachbauten

Tandy Modell 100, NEC PC8201A und Olivetti M10 sind drei Versionen eines einzigen Grundmodells. Ein Leistungsvergleich aller Geräte.

## Camera obscura

Die elektronische Kamera EV1 bringt Bilder auf den Monitor. Das Gerät kostet inklusive Software etwa 450 Mark.




+++ Laservisionen +++ Register und Speicher +++ Daten-Ablage +++ Verkabelte Klänge +++ BASIC und LOGO +++ Fachwörter +++ Transistor-Logik +++ Audio-Spiel mit Listing +++