Einsteigen - Verstehen - Beherrschen

DM 3,80 öS 30 sfr 3,80

computer kurs

- Micro-Überwachung
- Bits aus der Büchse
- Action ins Spiel
- Rechner-Netzwerke
- Tips: „Sphärenklänge“
- Maschinensprache
- Neu von Oric: Atmos

Heft 17

Ein wöchentliches Sammelwerk

computer kurs

Heft 17

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

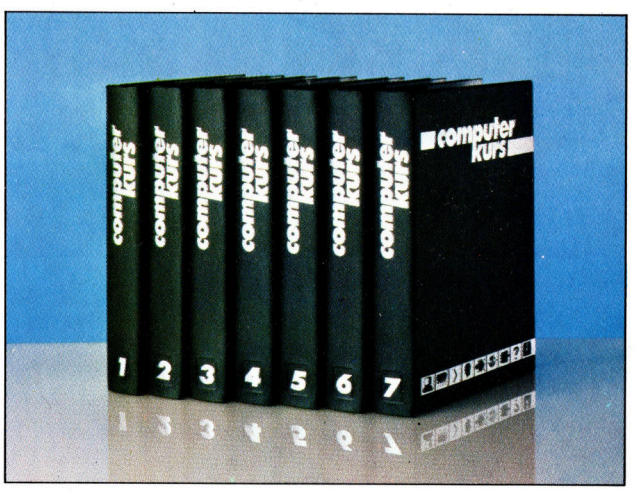
Schweiz: Der Sammelordner kostet sfr 15. Bitte wenden Sie sich an Ihren Kiosk; dort werden Sie jederzeit die gewünschten Exemplare erhalten.

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

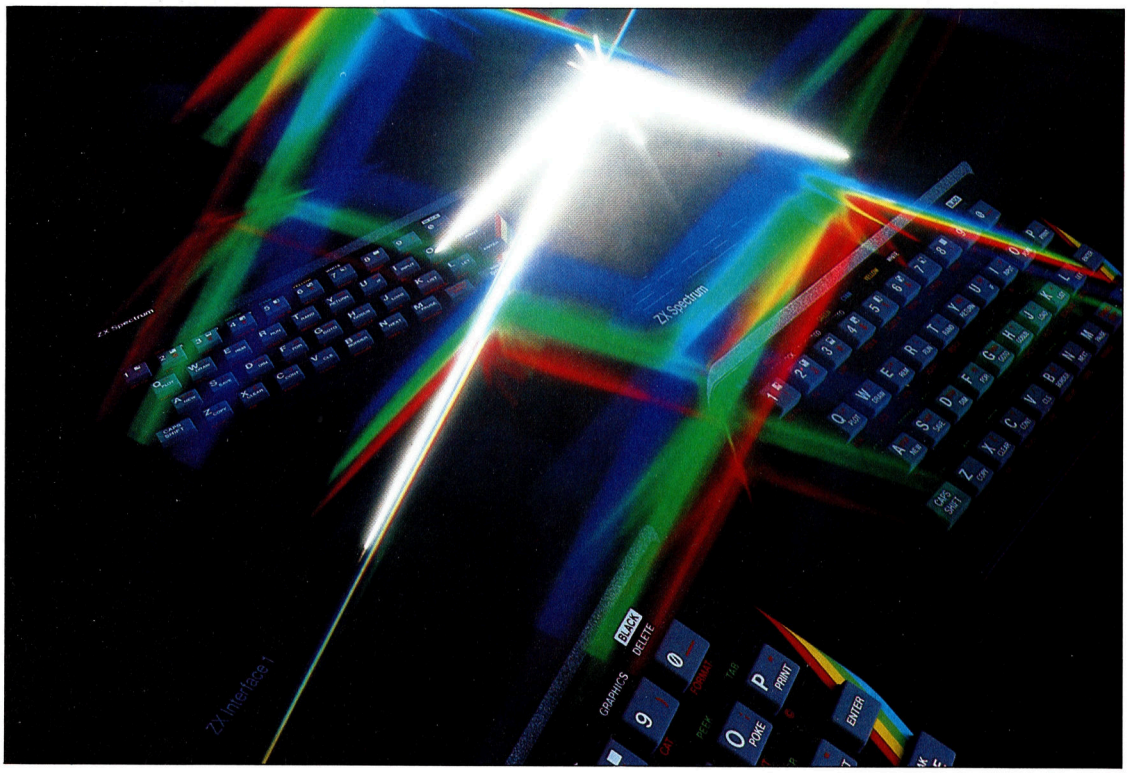
Vertrieb: Marshall Cavendish International Ltd., Heidenkampsweg 74, 2000 Hamburg 1, Tel.: 040/23 40 85



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

Netzwerk-Verbund

Computer-Netzwerke können sich über das ganze Land erstrecken, aber auch kleinere Bereiche abdecken, z. B. die Verbindung zwischen Reisebüros und Fluggesellschaften. Diese Systeme werden allerdings mit kapazitätsstarken, teuren Computern betrieben.



Netzwerke müssen nicht unbedingt mit kilometerlangen Kabeln und kostspieliger Ausrüstung ausgestattet sein. Spectrum-Rechner, die mit dem Interface 1 ausgerüstet sind, können zu minimalen Kosten miteinander kommunizieren und Microdrives oder Printer gemeinsam nutzen.

Unter dem Begriff Netzwerk versteht man ein System von Computern, die miteinander verbunden sind, um Daten und Peripherie gemeinsam zu nutzen. Aber jeder Computer hat sein eigenes Betriebssystem und seine spezifischen „Protokolle" (Prozeduren, Formatierungseigenarten etc.), um mit der Außenwelt zu kommunizieren. Wegen dieses Kompatibilitätsproblems müssen die einzelnen Netzwerk-Stationen Computer eines Typs sein.

Angenommen, eine Gruppe von Usern hat fünf Computer, die mit einem einzigen Drucker verbunden werden sollen. Diese Gruppe muß von jedem Terminal Informationen an den Drucker senden können. Was aber, wenn von zwei oder drei Terminals gleichzeitig Text ausgedruckt werden soll? Noch wichtiger ist, was geschieht, wenn an Terminal 3 während des Druckens weiter gearbeitet werden muß. Um diese Probleme zu lösen, muß ein sechster Computer, Print Server genannt, installiert werden. Dieser Rechner hat ausschließlich die Funktion, den Datenfluß zum Drucker zu kontrollieren, und kann deshalb für nichts anderes verwendet werden. Der Print Server speichert die Dokumente nach ihrer Priorität. Sobald ein Text vom Terminal zum Print Server geschickt worden ist, kann am Terminal weitergearbeitet werden.

Der Einsatz eines solchen speziellen Rechners, der als Server dient, ist Grundvoraussetzung für ein Netzwerk, denn allein durch ihn ist die Teilung von Informationen möglich. Ergänzend zum Print Server wären in anderen Netzwerken File Server erforderlich, um gemeinsame Diskettenstationen nutzen und den Informationsfluß von Terminal zu Terminal kontrollieren zu können.

Nächster Schritt ist die Verbindung der einzelnen Rechner, die von Computer zu Computer mittels Kabel – Doppelkabel oder Koaxialkabel – erfolgt. Obwohl es verschiedene Anordnungsmöglichkeiten für ein Netzwerk gibt, so etwa stern- oder ringförmig, ist das Grundkonzept identisch. Zur Herstellung einer Verbindung benötigt man ein spezielles Netzwerk-Interface für jedes Terminal. Das kann eine einfache Schnittstelle RS 232 sein oder eine entsprechende Erweiterungsplatine. Ferner ist für die Server-Station eine Speichereinheit erforderlich, die über ausreichende Kapazität verfügt, um alle Daten abzuarbeiten. Die

Server-Station selbst muß über genügend RAM verfügen, um das Netzwerk richtig verwalten zu können.

Von der in einem Computer verwendeten Software hängt die Güte der Funktion eines Rechners ab. Und dies gilt besonders für Netzwerke. Eine spezielle Layer-Software, die neben dem Betriebssystem des Rechners vorhanden ist, stellt die Verbindung zwischen Terminals und Netzwerk insgesamt her. Durch diese sogenannte Netzwerk-Software kontrolliert die Server-Station die Abläufe spezieller Operationen und den Datenfluß des Netzwerks. Ergänzend dazu informiert die Netzwerk-Software jeden Terminal-Computer darüber, daß er an ein Netzwerk angeschlossen ist, ein Server zur Verfügung steht und wieviele andere Terminals vorhanden sind. Schließlich erhalten die Terminals über die Netzwerk-Software ein Protokoll für die Kommunikation mit dem Gesamtsystem. Voraussetzung für die Funktion eines Netzwerks ist also dieser Layer.

Nach dessen Installierung müssen die einzelnen Terminals ein Programm oder eine Reihe von Programmen haben, die das Netzwerk erkennen und „wissen", wie damit zu kommunizieren ist. Diese Software wird entweder von Cassette bzw. Diskette direkt geladen oder durch den File Server übermittelt. Die Software ist nur so komplex wie die Operation. Läuft bei Terminal 1 des Netzwerks ein Textverarbeitungsprogramm, das die Ergebnisse unabhängig von den anderen Rechnern an den Drucker sendet, ist lediglich eine Modifikation beim Textverarbeitungsprogramm erforderlich, und zwar die des Netzwerk-Befehlssatzes. Komplizierter wird es, wenn die Terminals 2 und 4 dieselben Daten benutzen und die Ergebnisse gegenseitig eingesehen werden müssen. In diesem Fall haben Software (ob Textverarbeitung, Spreadsheet, Datenbank oder auch ein Spiel) wie Hardware mehrere Aufgaben gleichzeitig zu erfüllen. Die Zentraleinheit muß also mehrere Aufgaben erledigen und zugleich die Kommunikation zweier anderer Zentraleinheiten miteinander korrekt steuern können.

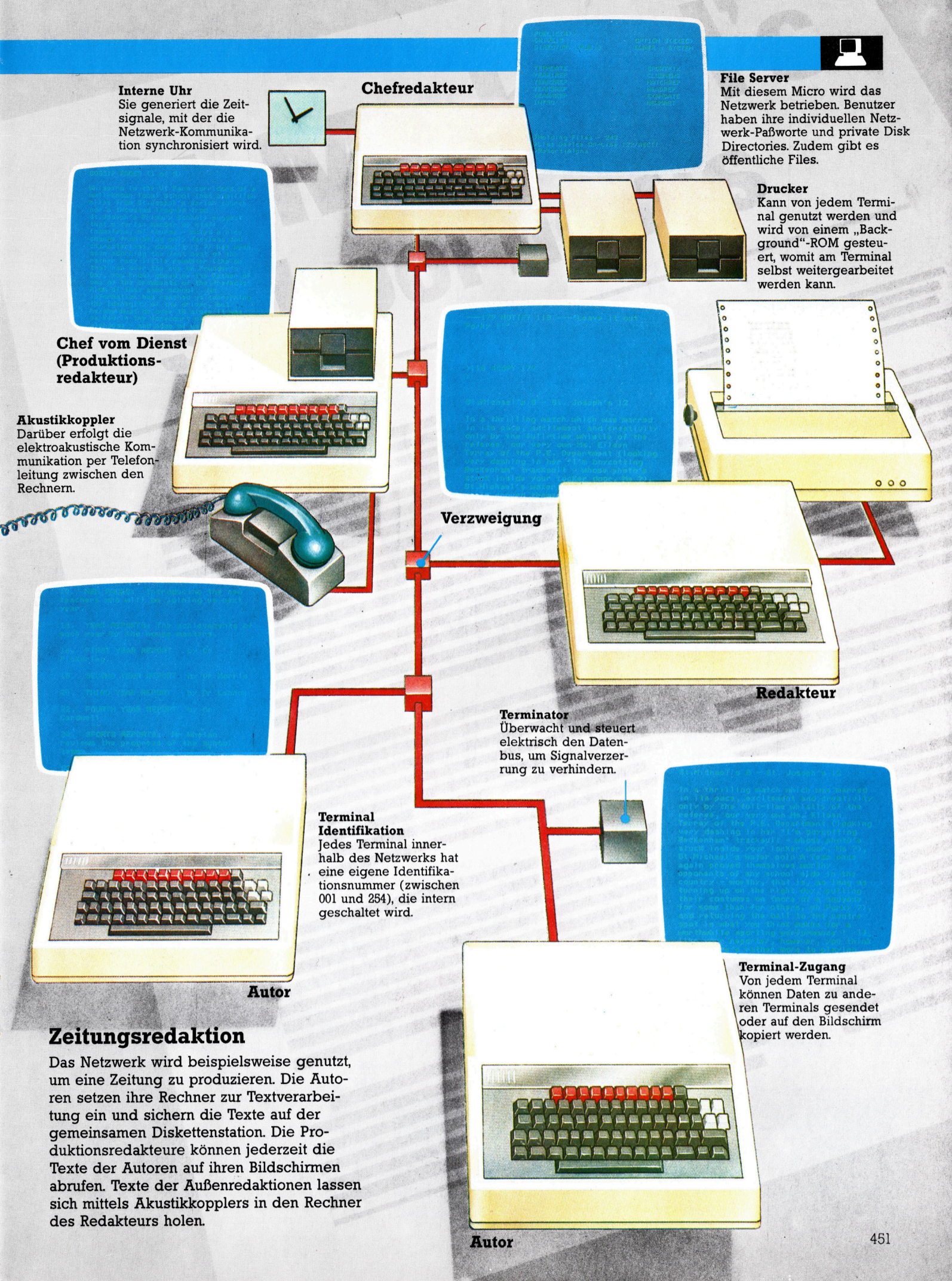
Geteilte Erfahrungen
Diese Schule wurde 1984 mit 16 Acorn B Computern, Farbmonitoren, einem Drucker, einem Doppel-Disk-Drive und Econet (Acorns Netzwerksystem für den Acorn B) zum Preis von 64 000 Mark ausgestattet. Dies war preiswerter, als für jeden Computer eine Diskettenstation und einen Drucker zu kaufen. Die Arbeitsgeschwindigkeit ist so groß, daß 30 Schüler gleichzeitig arbeiten können, ohne daß Verzögerungen auftreten. Die Kommunikation der Terminals untereinander ist bei der Nutzung des Netzwerks im Unterricht ein besonderer, nicht zu unterschätzender Vorteil.

Sinclair Research hat als erster Anbieter Netzwerke für den privaten Anwender möglich gemacht. In das Interface 1 für den Spectrum ist eine Netzwerk-Schnittstelle integriert. Damit können Microdrives in Verbindung mit dem Spectrum betrieben werden. Entsprechende Netzwerk-Software dürfte bald verfügbar sein.

Sinclairs neuester Rechner, der QL, verfügt standardmäßig über ein Netzwerk-Interface, kompatibel mit dem des Spectrum. Wenngleich dieses Interface auch einfach ist, wird die Produktion von Netzwerksoftware wegen der Popularität des Rechners für Anbieter attraktiv. Spielprogramme werden wahrscheinlich als erstes angeboten. Andere Applikationen für Heimcomputer-Nutzer gibt es im Moment kaum.

Zur Verwirklichung eines Netzwerks sind mehrere Elemente erforderlich. Zunächst einmal müssen natürlich mindestens zwei Micros miteinander verbunden werden. Diese sollten so dicht beieinander plaziert sein, daß eine Kabelverbindung möglich ist. Sie müssen sich also in einem Gebäude befinden. Schließlich, um das Netzwerk überhaupt sinnvoll zu machen, ist der Daten-Verkehr erforderlich. Dies bedeutet: Es muß Anwender geben, die mehrmals täglich Daten auszutauschen haben oder sich die hohen Kosten teilen.

Werden nur wenige Daten innerhalb des Netzwerks bewegt, ist es für die Anwender einfacher, die Datenträger (Cassetten oder Disketten) miteinander auszutauschen. Sind nur wenige Rechner an das Netzwerk angeschlossen, wäre es vielleicht billiger, jeden einzelnen mit einem Drucker und einer Diskettenstation auszurüsten. Für den praktischen Einsatz eines Netzwerks in kleinem Rahmen bieten sich lediglich Anwendungen in Schule und Kleinbetrieben an.



Zeitungsredaktion

Das Netzwerk wird beispielsweise genutzt, um eine Zeitung zu produzieren. Die Autoren setzen ihre Rechner zur Textverarbeitung ein und sichern die Texte auf der gemeinsamen Diskettenstation. Die Produktionsredakteure können jederzeit die Texte der Autoren auf ihren Bildschirmen abrufen. Texte der Außenredaktionen lassen sich mittels Akustikkopplers in den Rechner des Redakteurs holen.

Leonardo Torres

Seine Interessen reichten vom Luftschiffbau bis zu Drahtseilbahnen. Torres leistete aber auch wichtige Beiträge zur Computer-Entwicklung.

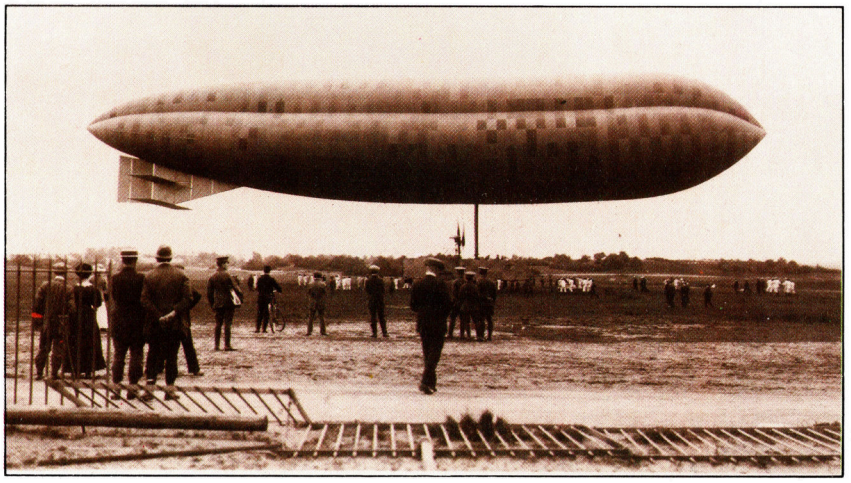
Leonardo Torres y Quevedo, Erfinder der Fließkomma-Arithmetik, wurde am 28. 12. 1852 im spanischen Santa Cruz geboren. Er ging in Bilbao aufs Gymnasium und besuchte später die Technikerschule von Madrid, der Beginn einer brillanten Ingenieurs-Karriere.

Seine großen Erfindungen machte Torres erst in der zweiten Lebenshälfte. Dazu gehört die Niagara-Brücke und eine Drahtseilbahn an den berühmten Wasserfällen, aber auch das im Ersten Weltkrieg eingesetzte Starrluftschiff. Das besondere Interesse des Spaniers galt aber elektromechanischen Geräten. 1906 führte er dem spanischen König ein funkferngesteuertes Modellboot vor. 1911 baute er einen Schach-Automaten, der die Figuren mit verborgenen Elektromagneten auf dem Spielfeld bewegen und ein einfaches Spiel gegen einen Menschen gewinnen konnte.

Torres Interesse an Automaten entwickelte sich aus der Kenntnis der ersten Fließbänder, die Anfang des 20. Jahrhunderts in den Fabriken zum Einsatz kamen. Zeit seines Lebens beschäftigte ihn die Frage, welche Teile der menschlichen Arbeit sich besser durch Maschinen erledigen lassen.

Im Jahre 1914 veröffentlichte er eine Studie, in der er die Realisierbarkeit von Babbages Analytischer Maschine mittels elektromechanischer Techniken bewies. Und in dieser Studie war auch erstmals von der Fließkomma-Arithmetik für den Einsatz in zukünftigen Computern die Rede. Schon 1920 baute Torres einen elektromechanischen Rechner, der zur Datenein- und -ausgabe mit Telefondraht an eine Schreibmaschine gekoppelt war – eine frühe Version des zentralen Rechners mit angeschlossenen Terminals.

Torres wurde Ehrenmitglied der französischen Akademie der Wissenschaften, später sogar Vorsitzender der spanischen Wissenschafts-Akademie. Er starb am 18. Dezember 1936 in Madrid.



Leonardo Torres war ein Universalgenie. Seine Interessen reichten vom Entwurf mechanischer Geräte – wie dem hier abgebildeten Luftschiff – über das Gebiet elektromechanischer Rechenmaschinen bis zu der reinen Mathematik.

Fließkomma-Arithmetik

Eine Ladenkasse zeigt die Summe in Mark und Pfennig an (z. B. 12,25 Mark), es werden also nur zwei Stellen nach dem Dezimalkomma gebraucht. Anders in einem Computer: Hier muß mit größerer Genauigkeit gearbeitet werden. Die Anzahl der Stellen hinter dem Komma bzw. Punkt „fließt" je nach den gestellten Anforderungen – daher die Bezeichnung „Fließkomma-Arithmetik".

Jede Zahl kann auf unterschiedliche Weise dargestellt werden. So läßt sich für 0,8752 Meter ebenso gut 875,2 Millimeter, $0{,}8752 \times 1000$ mm bzw. einfach $0{,}8752 \times 10^3$ mm schreiben. Die letztere Schreibweise ist für den Computer am wirtschaftlichsten. Kann der Rechner nur sechs Ziffern verarbeiten, läßt sich unsere Zahl als 875203 speichern: Die letzten beiden Ziffern rechts nennt man „Index", sie bedeuten „mal 10 hoch x" (in diesem Fall 3). Die vier ersten Ziffern sind die „Mantisse". Zweites Beispiel: Die Zahl 418302 im Rechner heißt ausgeschrieben $0{,}4183 \times 10^2$ oder 41,83.

Üblicherweise werden Zahlen vom Rechner zuerst in die Normalform gebracht, führende Nullen werden aus der Mantisse entfernt. Die Zahl 41,83 ließe sich zwar als 004104 ausdrücken, enthält aber in der Normalform 418302 eine höhere Genauigkeit in der Darstellung der Mantisse.

Vorteil der Mantisse/Index-Darstellung einer Zahl ist der große Wertebereich, der mit nur sechs Ziffern abgedeckt werden kann. Für den oben vorgestellten Rechner, der mit einem zweiziffrigen Index arbeitet, reichen die Werte von $0{,}999 \times 10^{99}$ bis zu einer weitaus kleineren Zahl, bei der erst nach 98 Stellen hinter dem Dezimalpunkt eine richtige Ziffer auftaucht.

Natürlich bleibt die Rechengenauigkeit bei diesem Verfahren durch die Anzahl der Ziffern in der Mantisse beschränkt. Einige Zahlen lassen sich nur annähernd darstellen, so daß bei der Programmierung von Rechenvorgängen große Vorsicht und Sachkenntnis nötig ist, um Fehler zu vermeiden. Diese Schwierigkeit kennt jeder, der schon einmal mit Computern eine einfache Berechnung wie $(1/3) \times 3$ ausgeführt hat: Meist erscheint 0,9999999 anstelle des richtigen Ergebnisses „1".

Oric Atmos

Der Oric-1, 1983 in England auf den Markt gebracht, konnte an die Erfolge seiner Konkurrenz aufgrund von technischen Mängeln und fehlender Softwareunterstützung nie anknüpfen. Beim Oric Atmos wurden die Fehler des Vorgängers bereinigt.

Der Oric-1 schien mit seinem dem Microsoft ähnlichen BASIC, eingebauter Centronics-Schittstelle und der Anschlußmöglichkeit für einen Farbmonitor keine schlechte Geldanlage zu sein. Leider war für die Maschine nur wenig gute Software erhältlich. Da außerdem das ROM etliche Fehler aufwies, konnte sich dieser Rechner nicht durchsetzen.

Oric Products International hat die Fehler im ROM inzwischen beseitigt und den Computer in der verbesserten Version des Oric Atmos auf den Markt gebracht. Die ursprünglichen „Taschenrechnertasten" sind durch eine professionelle Schreibmaschinentastatur ersetzt worden. Das Gehäuse präsentiert sich in stilvollem Rot und Schwarz. Entgegen der ersten Serie des Atmos, die noch mit der englischen QWERTY-Tastatur ausgestattet war, verfügt das Gerät nun über eine Tastenbelegung nach dem QWERTZ-Format (das heißt, Umlaute und ß sind vorhanden).

Der Atmos arbeitet mit dem 6502-Prozessor und bietet 37 KByte RAM für die Programmierung. Für die Grafik stehen acht Farben und eine maximale Auflösung von 240 × 200 Pixeln zur Verfügung. Der Zeichensatz liegt im RAM und gibt damit dem Anwender die Möglichkeit, jedes Zeichen neu zu definieren, wobei in einem zweiten, zusätzlichen Zeichensatz teletexähnliche Grafikelemente definiert sind.

Sounds für Spiele

Im ROM des Atmos sind vier vorgefertigte Klänge vorhanden – ZAP, PING, SHOOT und EXPLODE –, die die entsprechenden Geräusche für Spiele produzieren. Mit den Befehlen MUSIC, PLAY und SOUND läßt sich aber mit einer Vielzahl von Parameterangaben auch der hochentwickelte Chip für die Klangerzeugung voll ausnutzen. Dabei kann die Lautstärke von sehr leise bis äußerst laut eingestellt werden, während drei Tongeneratoren und ein Generator für Rauschen immerhin einen Tonumfang

Der Oric Atmos ist ein preisgünstiger Heimcomputer mit einem Arbeitsspeicher von 48 K, Farbgrafik und Tongenerator. Es gibt zwei Zusatzgeräte in den gleichen Gehäusefarben: ein Diskettenlaufwerk als Alternative zu einem Cassettengerät und einen Drucker/Plotter, der Grafik und Text in Farbe darstellen kann.

Oric-1
Der Atmos ist eine verbesserte Version des Oric-1. Er besitzt die gleiche Grundplatine, hat jedoch ein neues BASIC-ROM. Die Unzulänglichkeiten, die der Oric-1 aufwies, wurden bei der Produktion des Atmos korrigiert. In diesem Zusammenhang soll auch nicht unerwähnt bleiben, daß viele Oric-1-Programme nicht auf dem Atmos laufen.

von sieben Oktaven erzeugen.

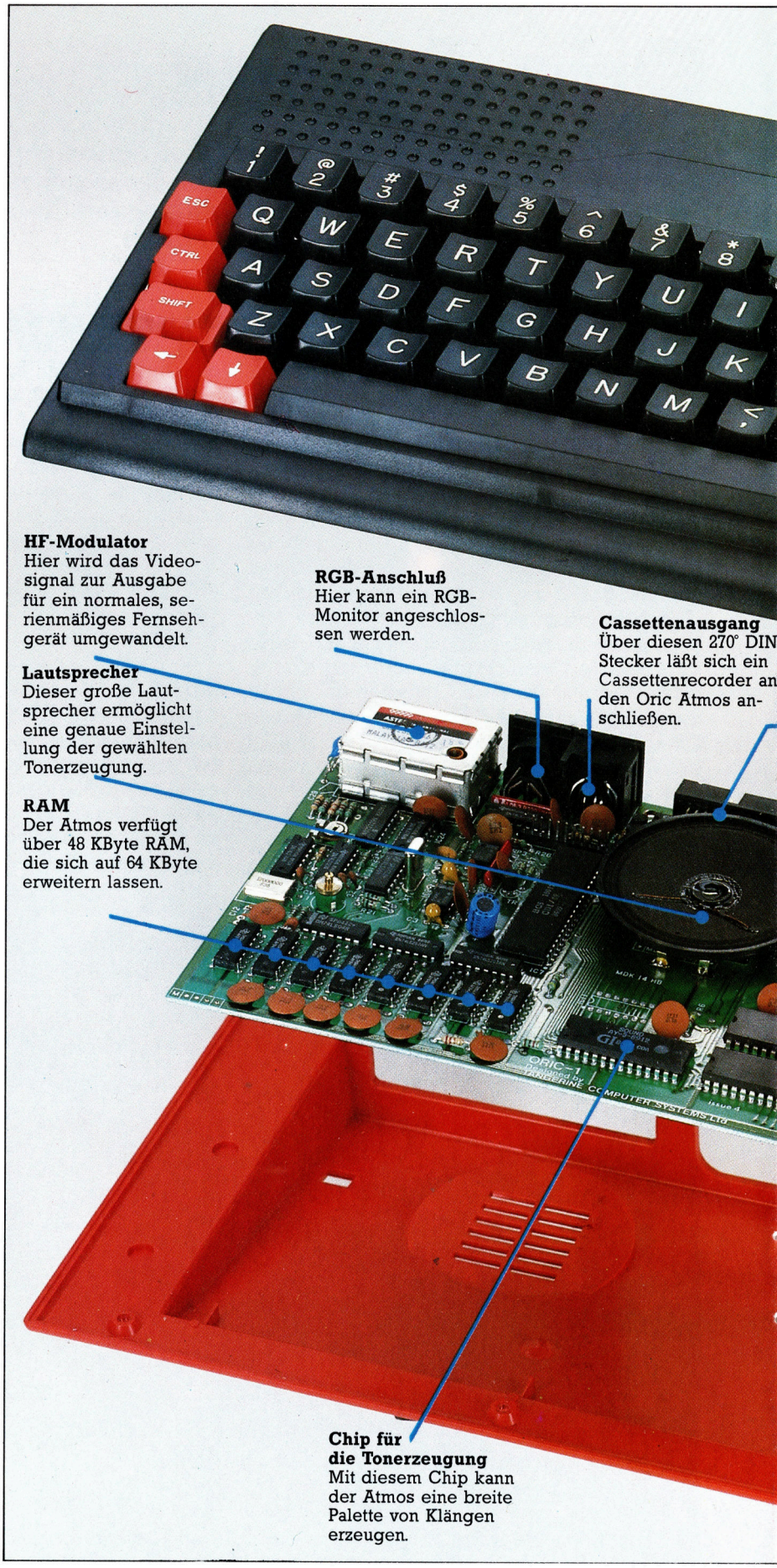
Das ursprüngliche Oric-BASIC enthielt einige gravierende Fehler. So arbeitete der TAB-Befehl nicht richtig, und die Bildschirmanzeige war oft mit ungewollten Klangeffekten durchsetzt. Auch erzeugte der Oric-1 bei der Ausführung der STR$-Funktionen fehlerhafte Steuercodes und gab bei der Verwendung von LEN und VAL falsche Werte an. In dem neuen ROM sind diese Schwächen beseitigt. Unglücklicherweise funktionieren aufgrund dieser Verbesserungen die Maschinenprogramme des Oric-1 nicht auf dem Atmos, da etliche Adressen im ROM umgesetzt wurden.

Das BASIC ist eine erweiterte Version des Microsoft Dialektes und wurde von der Firma Tansoft aus dem ursprünglichen Tangerine-BASIC entwickelt. Es unterstützt die volle IF...THEN...ELSE-Struktur und macht auch die REPEAT...UNTIL-Schleife verfügbar. Eine ungewöhnliche Möglichkeit, aus GOSUB und REPEAT...UNTIL-Routinen ohne Fehlermeldung herauszuspringen, bieten die Befehle POP und PULL.

Gleichzeitig mit dem Atmos brachte Oric auch den Drucker/Plotter neu heraus. Die vier verkürzten Kugelschreiberminen, mit denen das Gerät ausgerüstet ist (schwarz, rot, grün und blau), sind auf einem drehbaren Druckkopf montiert. Die einzelnen Farben können dabei über die Software angewählt werden. Der Drucker/Plotter produziert zwar nur 12 Zeichen pro Sekunde, druckt aber auf Normalpapier.

Auch das langerwartete Microdiskettenlaufwerk präsentiert sich in den Farben des Atmos. Oric setzt dabei auf die Hitachi 3" Disketten, die von einem stabilen Plastikgehäuse umschlossen sind. An den Atmos können dabei bis zu vier Diskettenlaufwerke angeschlossen werden.

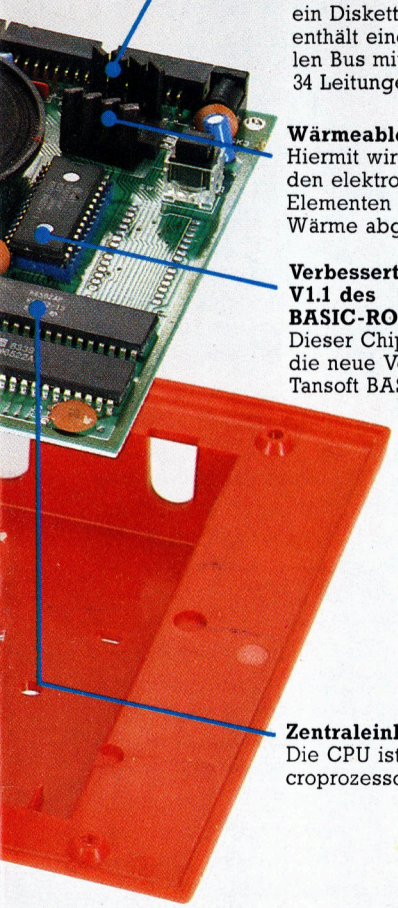
Zum Lieferumfang der Diskettenstation gehört ein seperates Netzteil, das zwei Lauf-



Funktionstaste
Diese Taste ist für Sonderfunktionen vorgesehen.

Druckerschnittstelle
Parallele Centronics-Schnittstelle

Erweiterungsschnittstelle
Diese Schnittstelle für ein Diskettenlaufwerk enthält einen parallelen Bus mit insgesamt 34 Leitungen.



Wärmeableitung
Hiermit wird die von den elektronischen Elementen erzeugte Wärme abgeleitet.

Verbesserte Version V1.1 des BASIC-ROMs
Dieser Chip enthält die neue Version des Tansoft BASIC.

Zentraleinheit
Die CPU ist der Microprozessor 6502.

Diskettenstation
Der Oric arbeitet mit 3" Disketten, die durch ein hartes Plastikgehäuse geschützt sind. Die Disketten haben pro Seite eine Kapazität von 160 KByte. Die Diskettenlaufwerke können nur sequentielle Daten und keine Dateien mit direktem Zugriff (Random Access) verarbeiten.

werke und den Computer selbst mit Strom versorgen kann. Bei frühen Versionen des Betriebssystems traten Fehler auf, wenn Drucker und Laufwerk gleichzeitig eingeschaltet waren. Jeder Versuch, eine Programmzeile zu korrigieren, hatte die Wirkung, daß diese Zeile, wie auch alle anderen numerierten Programmzeilen, gelöscht wurden. Laut Oric soll dieser Fehler in den neueren Versionen des Betriebssystems eliminiert worden sein.

Obwohl die frühen Versionen des Betriebssystems noch Probleme mit Drucker und Cassettenlaufwerk hatten, so scheint Oric auf die Konstruktion des Atmos und den Anschluß von Peripheriegeräten doch große Sorgfalt verwandt zu haben. Die Konstrukteure haben viele kritische Punkte des Oric-1 berücksichtigt und die meisten Fehler bereinigt. Da sich nur wenige Software-Produzenten des Oric-1 angenommen hatten, wurde die Firma Tansoft beauftragt, eine Anzahl Programme herzustellen, die mit dem Diskettenlaufwerk arbeiten. Je mehr Software für den Atmos verfügbar wird, desto größer wird der Marktanteil sein, den Oric sich mit der Maschine erobern kann.

Drucker/Plotter
Ein ausgezeichnetes Zusatzgerät des Atmos ist dieser Drucker/Plotter. Farbige Textausgabe ist ebenso möglich wie die Erstellung von Grafiken. Die Buchstabengröße läßt sich von Millimetern bis auf eine Größe von mehreren Zentimetern einstellen. Nachteilig sind die Papierbreite mit nur etwa 11 cm und die geringe Druckgeschwindigkeit.

Oric Atmos

PREIS
600 DM

GRÖSSE
278 × 178 × 50 mm

CPU
6502

MASCHINENSPEICHER
48 K RAM, 16 K ROM

BILDSCHIRMDARSTELLUNG
28 Zeilen mit je 40 Zeichen im Textmodus, bis zu 200 × 240 Pixel im 8 Farben-Grafikmodus

SCHNITTSTELLEN
Centronics Druckerschnittstelle, Cassettenbuchse und RGB-Monitoranschluß

TASTATUR
57 Schreibmaschinentasten, plus zusätzliche Funktionstaste

DOKUMENTATION
Das Handbuch ist auf Anfänger ausgerichtet, die BASIC lernen wollen. Für Fortgeschrittene gibt es Kapitel über Maschinencode und höhere Ein- und Ausgabetechniken, wie auch eine Anzahl von Anhängen, die über die technische Ausstattung berichten.

STÄRKEN
Der Atmos verfügt über eine breite Palette von Programmiermöglichkeiten. Das BASIC ist gut und verfügt über eine Anzahl Befehle, die das Programmieren einfach machen.

SCHWÄCHEN
In der hochauflösenden Grafik läßt sich der Atmos nur umständlich handhaben. Die Diskettenstationen sind nur für sequentielle Verarbeitung geeignet.

Programmierhilfen

„Tool Kits" sind Programmierhilfen, die den BASIC-Bereich erweitern und zusätzliche Funktionen zur Fehlersuche bieten.

Ältere Heimcomputer, wie der Apple II und der Commodore PET, sind im wesentlichen für die Verarbeitung von Zahlen und Text ausgelegt. Ihre BASIC-Versionen beschränken sich daher auf Befehle und Routinen, die auf diesen Zweck abgestimmt sind. Um diesem Mangel entgegenzuwirken, entstanden die sogenannten „Tool Kits" (zu deutsch: Werkzeugsätze) als Programmierhilfen. Sie arbeiten als externe BASIC-Erweiterungen und sind gewöhnlich in Maschinensprache gehalten. Tool Kits liefern direkte Zusatzbefehle, mit denen Programmentwicklung und Fehlersuche erleichtert werden.

Das explosionsartig angewachsene Interesse an Computerspielen hat dazu geführt, daß neue Heimcomputer durch immer ausgeklügeltere Grafik- und Tonfähigkeiten überraschen. Die BASIC-Version dieser Heimcomputer wurde jedoch, von ein oder zwei Fällen abgesehen, nicht oder nur geringfügig gegenüber den früheren Versionen verbessert. Der Anwender muß, um die neuen Fähigkeiten mit den verfügbaren Befehlen handhaben zu können, eigene Routinen ausarbeiten. Dies wird durch die Verwendung von PEEKs und POKEs sehr mühselig. Hier helfen nun die „Tool Kits". Es gibt sie heute bereits als Hilfs- und Zusatzprogramme zur BASIC-Version für die meisten Heimcomputer. Diese Programme ermöglichen im allgemeinen einen leichteren Zugriff auf bestehende Funktionen (z. B. Sprite- oder Ton-Editor), erweitern die Software-Eigenschaften (z. B. Sprite-Generatoren) oder liefern einfache Hilfen zur Programmierung in BASIC.

Programmierhilfen dieser Art können in den RAM- oder internen ROM-Speicher geladen oder auch von einem ROM-Modul gelesen werden. Ein ROM-Modul ist auf jeden Fall vorzuziehen, weil dadurch der im Computer frei verfügbare Speicherbereich voll erhalten bleibt und ein unbeabsichtigtes Löschen vermieden wird. Ein mit Programmierhilfen erstelltes Programm läuft in der Regel nur auf gleich ausgerüsteten Computern. Es gibt jedoch auch Programmierhilfen, die „freistehende" Programme erzeugen, die auch auf anderen Computern funktionieren. Dies trifft für einige Grafik- und Sprite-Editoren sowie für Ton-Editoren zu.

Nützlich sind spezielle Grafikbefehle, wie PAINT, DRAW, PLOT, CIRCLE und Tonbefehle wie SOUND, PLAY, MUSIC, ENVELOPE oder auch Anweisungen, die einen bestimmten Geräuscheffekt beschreiben, wie BANG oder ZAP. Von Nutzen sind auch Hilfen zum strukturierten Programmieren, wie REPEAT . . .UNTIL und IF . . .THEN . . .ELSE. Solche Anweisungen machen es möglich, Programme zu schreiben, die ohne allzuviele GOTOs auskommen.

Dies sind einige Programmierhilfen und BASIC-Erweiterungspakete, die es für die populärsten Heimcomputer gibt. Software-Pakete zur Erzeugung von Sprites werden immer beliebter.

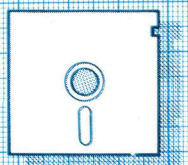
Programmierhilfen	
SUPER TOOL KIT	Von Nectarine für den 16K- und 48K-Spectrum
SPECTRUM EXTENDED BASIC	Von CP Software für den 48K-Spectrum
SPECTRUM KEYDEFINE	Von Scientific Software für den 48K-Spectrum
PROGRAMMER'S AID	Von Commodore für den VC 20
BUTI	Von Audiogenic für den VC 20
TOOL BOX	Von BBC Software für den Acorn B
SPRITE MAGIC	Von Merlin Micro Systems für den Dragon 32
SPRITE GRAPHICS	Von B Sides Software für den 48K-Spectrum
SPRITE MASTER	Von Micro Dealer UK für den Acorn B

Simon's BASIC

Eine der umfangreichsten BASIC-Erweiterungen ist das „Simon's BASIC", verfügbar als ROM-Modul für den Commodore 64. Das Standard-BASIC des Commodore 64 bietet kaum mehr als ein Minimum an Befehlen und nichts für strukturiertes Programmieren. Die recht fortschrittlichen Hardware-Eigenschaften dieses Computers (wie der umfassende Ton-Synthesizer, die hochauflösende Grafik und die Sprites) werden lediglich über PEEK und POKE gesteuert. Mit den folgenden Hilfen

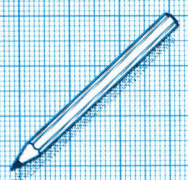
Tool Kits

Die hier gezeigten Befehle sind repräsentativ für gute BASIC-Erweiterungsprogramme:



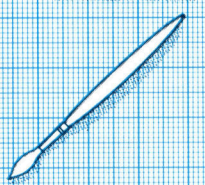
DISK

Verschiedene Befehle zum Formatieren neuer Disketten, Kopieren und Löschen einzelner Dateien und Herstellen von Sicherungskopien.



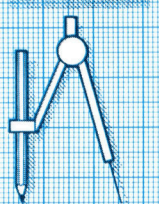
DRAW

Neben den Befehlen zum Erstellen gerader Linien in hochauflösenden Grafiken können hier auch Befehle für Punkt-zu-Punkt-Zeichnungen (DRAW) enthalten sein; damit lassen sich Bilder leichter aufbauen.



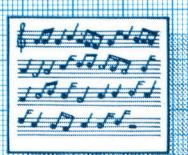
PAINT

Die Anweisung PAINT ermöglicht das Ausfüllen bestimmter Bildschirmbereiche mit Farbe; der Computer füllt die Fläche aus, indem er von der Mitte aus zur Randlinie arbeitet.



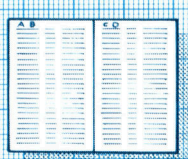
CIRCLE

Die Funktion von Grafikbefehlen ist offensichtlich. Einige erlauben das Zeichnen von Bögen und Ellipsen.



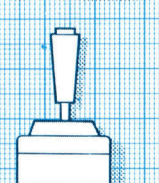
MUSIC

Die Tonfunktionen sind zwar von Gerät zu Gerät verschieden, jedoch umfaßt der MUSIC-Befehl gewöhnlich die Fähigkeit, eine Notenfolge zu spielen, die vorher in einem String definiert wurde.



DIR

Das Inhaltsverzeichnis einer Diskette enthält die aktuelle Liste der Filenamen. Mit Hilfe des DIR-Befehls ist es im allgemeinen möglich, das Inhaltsverzeichnis auf den Bildschirm zu holen, ohne das im RAM befindliche Programm zu verlieren.



JOY

Der Zugriff zum Joystick mit Hilfe von PEEK und POKE ist bei manchen Computern schwierig; Joystick-Befehle legen die Position des Sticks direkt in einer Variablen ab.

stellt Simon's BASIC eine beachtliche Erweiterung zum Commodore-BASIC dar:

(1) umfassende Programmierhilfen, einschließlich der Funktionen, die eine zusätzliche Kontrolle über Programmauflistung, Fehlersuche und Programmsicherung gegen unberechtigtes Kopieren ermöglichen,
(2) zusätzliche String- und Textbefehle,
(3) zusätzliche Arithmetik-Operatoren und Befehle zur String-Verwaltung,
(4) vereinfachte Befehle zur Laufwerksteuerung,
(5) Befehle für hochauflösende Grafik zum Einfügen von Text in Grafiken und zur Farbbestimmung,
(6) Grafik- und Bildschirmbefehle zum Duplizieren bestimmter Grafikbereiche und Verschieben des Bildschirmbereichs sowie zum Abspeichern des Bildschirminhaltes auf Diskette oder Cassette sowie zur Ausgabe über den Drucker,
(7) leicht zu steuernde Erzeugung und Änderung von Sprites,
(8) Ablaufbefehle zum strukturierten Programmieren (wie PROC, CALL und EXEC) sowie Schleifen und Status-Testroutinen (wie REPEAT . . .UNTIL, LOOP . . .EXIT, IF . . .END LOOP und IF . . .THEN . . .ELSE),
(9) Tonerzeugungsroutinen, die unter Verwendung einfacher Befehle zur Tonabstimmung den Zugriff auf den vollen Tonumfang des Commodore 64 erlauben, und
(10) einfache Lichtgriffel-, Paddle- und Joystick-Befehle.

Ein derart umfangreicher Satz zusätzlicher Routinen ist für eine BASIC-Erweiterung unüblich. Die meisten Erweiterungspakete enthalten nur Hilfen und Befehle für einen bestimmten Programmierbereich. Das Super-Expander-Modul für den VC 20 beispielsweise bietet lediglich einen einfachen Befehlsbereich für hochauflösende Grafik und Musik. Am geläufigsten sind Programmierhilfen zum Programmaufbau. Sie bieten gewöhnlich einfache Eingabebefehle und verschiedene automatische Routinen zur Vereinfachung der Zeilennumerierung, des Editierens und der Fehlersuche im Direktverfahren.

Allgemein kann gesagt werden, daß Programmierhilfen einen leichteren Zugriff auf die im Computer eingebauten Fähigkeiten ermöglichen. Seltener zu finden sind Routinen, die die Leistung eines Heimcomputers beträchtlich erweitern. Nach und nach kommen jedoch auch hochentwickeltere Programm-Pakete auf den Markt. So haben zum Beispiel die vielen Vorteile der Sprites für schnelle Action-Spiele einige Hersteller dazu animiert, Programmierhilfen zur Erzeugung von Sprites für Computer zu schreiben, die über diese Fähigkeit von Haus aus nicht verfügen.

Die hier genannten Programmierhilfen stellen nur einen kleinen Teil der heute erhältlichen Verbesserungen auf diesem Gebiet dar.

Ortswechsel

Nachdem bereits die Aufnahme neuer Verzeichnisse erklärt wurde, soll jetzt ein Weg ermittelt werden, die Daten erneut aufzurufen.

Der letzte Teil des BASIC-Kurses endete damit, daß Sie als Übung ein Datenverwaltungsprogramm schreiben sollten, das eine Dateneingabe ermöglicht. Voraussetzung ist, daß die folgenden Felder und entsprechenden Bereiche vorhanden sind:

FELD	BEREICH
1 NAME (Feld)	NAMFLD$
2 MODIFIZIERTER NAME (Feld)	MODFLD$
3 STRASSE (Feld)	STRFLD$
4 STADT (Feld)	STDFLD$
5 STAAT (Feld)	STAFLD$
6 TELEFONNUMMER (Feld)	TELFLD$
7 INDEX (Feld)	INDFLD$

Die Bedeutung der meisten Felder dürfte mit Ausnahme der Felder 2 und 7 klar sein. Betrachten Sie zuerst das MODIFIZIERTER NAME-Feld. Als wir das erste Mal das Problem des Datenformates für den Namen untersuchten, standen zwei Alternativen zur Wahl: ein fest vorgegebenes Format zur Namenseingabe oder eine beliebige Eingabeform. Da es eine Vielzahl an Möglichkeiten gibt, sind bei einer beliebigen Eingabe die Such- und Sortier-Routinen relativ kompliziert. Um dieses Problem zu lösen, werden die eingegebenen Namen in ein einheitliches Format umgewandelt: Alle Buchstaben werden zu Großbuchstaben, alle nichtalphabetischen Zeichen (z. B. Leerzeichen, Punkte, Apostrophe, usw.) werden entfernt und zwischen Vornamen und Familiennamen (falls vorhanden) befindet sich nur eine Leerstelle.

Feld-Umwandlung

Diese Standardisierung ist notwendig, da die Such- und Sortier-Routinen einen Vergleich durchführen müssen. Andererseits wollten wir beim Abrufen eines Namens oder einer Adresse die Daten in der eingegebenen Form erhalten. Zur Lösung dieses Problems gibt es zwei Möglichkeiten: Entweder wird jedes Namens-Feld nur beim Sortieren und Suchen in ein einheitliches Format umgewandelt, oder das Namens-Feld wird sofort in ein einheitliches Format gebracht und als separates Feld gespeichert.

Bei beiden Möglichkeiten gibt es Vor- und Nachteile. Die temporäre Umwandlung zum Sortieren und Suchen spart Speicherplatz, da weniger Daten in der Datei gespeichert werden müssen. Andererseits ist diese Methode sehr zeitintensiv. Reserviert man ein separates Feld für den standardisierten Namen, wird die Umwandlung für jedes Verzeichnis nur einmal vorgenommen.

Das andere Feld, das Verwirrung stiften kann, ist das INDEX-Feld. Es wurde als zusätzliches Feld integriert, um zukünftige Erweiterungen oder Modifikationen des Datensatzes zu erleichtern, ohne daß das ganze Programm umgeschrieben werden muß. Die Integration von Feldern heißt ‚binding' (Binden) – ein Ausdruck, der die Zusammenhänge zwischen den Daten umfaßt. Alle Felder oder Elemente in jedem der Verzeichnisse sind ‚zusammengebunden', da sie denselben Index haben und alle Felder eines Verzeichnisses gemeinsam in einer Datei gespeichert werden.

Dateistruktur festlegen

Bevor Sie versuchen, ein neues Verzeichnis in den Datensatz aufzunehmen, sollten Sie sich einige Gedanken über die Struktur von Dateien machen. Als erstes legen wir die Anzahl der Verzeichnisse auf 50 fest. Außerdem wird angenommen, daß alle Daten bereits in vorgegebene Bereiche übertragen worden sind (als Aufgabe der INITIALISIERUNGS-Routine).

Wenn ein neues Verzeichnis hinzugefügt wird, ist es am einfachsten, es am Ende der Datei abzulegen (das wäre das erste leere Element in jedem Bereich). Als erstes muß nun herausgefunden werden, wie groß der Bereich ist. Da das eine Information ist, die man in vielen Teilen des Programms gebrauchen kann, führt man dies sinnvollerweise in der INITIALISIERUNGS-Routine aus. Für diesen Zweck wird eine globale Variable eingesetzt (das ist eine Variable, die in jedem Programmteil verwendet werden kann). Wir nennen sie GROSS.

Da beim ersten Programmlauf noch kein Verzeichnis existiert, kann man der Variablen GEGEN erst dann einen Wert zuweisen, wenn das Programm etwas mit den Daten macht. Trotzdem kann man GEGEN in der INITIALISIERUNGS-Routine den Wert 0 zuweisen. Dieser Vorgang ist in BASIC nicht unbedingt notwendig, da er automatisch vollzogen wird. Es ist aber eine gute Angewohnheit und sollte für lokale Variablen immer vorgenommen werden. So kann unerwünschten Fehlfunktionen durch Mehrfachbenutzung derselben Variablen in verschiedenen Programmteilen vorgebeugt werden.

Beim ersten Start des Programms werden zahlreiche Initialisierungen durchgeführt und

Daten von Diskette oder Cassette geladen und in String-Variablen übertragen. Danach wird das AUSWAHL-Menü dargestellt. Wählt der Anwender nun Option 6 (Hinzufügen eines Verzeichnisses), enthält die Variable WAHL den Wert 6, worauf das Unterprogramm NEUADR aufgerufen wird.

Das Hinzufügen eines neuen Verzeichnisses bedeutet auch, daß die Datei sich nicht mehr in der richtigen Reihenfolge befindet. Da ein Sortiervorgang einige Zeit benötigt, ist es nicht unbedingt notwendig, alle Verzeichnisse nach jeder neuen Eingabe zu sortieren. Stattdessen wird ein Flag (Merker) gesetzt, der kennzeichnet, daß ein neues Verzeichnis eingegeben wurde.

Es folgt untenstehend eine komplette Liste der im Programm gebrauchten Arrays, Variablen und Flags:

ARRAYS (BEREICHE)

NAMFLD$	(Namens-Feld)
MODFLD$	(modifiziertes Namens-Feld)
STRFLD$	(Straßen-Feld)
STDFLD$	(Stadt-Feld)
STAFLD$	(Staat-Feld)
TELFLD$	(Telefon-Feld)
INDFLD$	(Index-Feld)

VARIABLEN

GROSS	(momentane Größe der Datei)
GEGEN	(Index des momentanen Verzeichnisses)

FLAGS

VNEU	(neues Verzeichnis einfügen)
SORT	(sortiert ab Verzeichnis Modifikation)
SICHERN	(gespeichert ab Verzeichnis Modifikation)
VMOD	(Modifikation durchgeführt seit letztem Sichern)

Es ist klar, daß im Verlauf der Programmentwicklung weitere Bereiche gebraucht werden. Wahrscheinlich sind auch weitere Variablen nötig. Eine neue Sicherung oder Sortierung der Dateien ist solange nicht notwendig, bis eine Modifikation stattgefunden hat. Somit ist VMOD wahrscheinlich das einzige Flag, das wirklich gebraucht wird. Als weitere Übung in der Top Down-Programmierung sehen Sie nun, wie einfach die *NEUADR*-Routine zu programmieren ist.

I 4 (AUSFÜHRUNG) 6 (NEUADR)

```
BEGIN (STARTE)
   Lokalisiere momentane Dateigröße
   Frage nach Eingaben
   Ordne Eingaben Ende des Bereiches zu
   Setze VMOD Flag
END (ENDE)
```

II 4 (AUSFÜHRUNG) 6 (NEUADR)

```
BEGIN (STARTE)
   (Größe der Datei ist GROSS)
   (Frage nach Eingaben)
   Lösche Bildschirm
   Drucke Meldung für erstes Element
      (GROSS)
   Eingabe der Daten in Element (GROSS)
      (Fragen und Eingaben für alle Ele-
      mente)
   Setze VMOD auf 1
END (ENDE)
```

All dies kann geradeaus programmiert werden und erfordert keine Schleifen oder komplizierte Strukturen. Der nächste Schritt kann bereits die Umsetzung in BASIC sein.

III 4 (AUSFÜHRUNG) 6 (NEUADR)
BASIC CODE

```
CLS: REM ODER VERWENDE PRINT
   CHR$(24) ETC ZUM LOESCHEN DES
   BILDSCHIRMS
INPUT "GIB NAMEN EIN";NAMFLD$
   (GROSS)
INPUT "GIB STRASSE EIN";STRFLD$
   (GROSS)
INPUT "GIB STADT EIN";STDFLD$(GROSS)
INPUT "GIB STAAT EIN";STAFLD$(GROSS)
INPUT "GIB TELEFON EIN";
   TELFLD$(GROSS)
LET VMOD=1
LET INDFLD$=STR$(GROSS)
GOSUB *MODNAME*
RETURN
```

In der drittletzten Zeile wird das INDFLD$-Feld auf den Wert von GROSS gesetzt (durch Umwandlung in einen String mit STR$), so daß es zu einem späteren Zeitpunkt als Index verwendet werden kann. Das Ergebnis der Routine *MODNAME* ist ein Element (GROSS) in einem Feld mit dem Namen MODFLD$. Alle Such- und Sortiervorgänge können nun mit diesem Feld durchgeführt werden. Da das Feld denselben Index wie die anderen Felder des Verzeichnisses hat, ist es einfach, den Namen und die Adresse in der ursprünglichen Form auszugeben.

Die Unterprogramme MODVER und LOESCHVER (zum Modifizieren und Löschen von Verzeichnissen) sind NEUADR sehr ähnlich, mit der Ausnahme, daß vor ihrer Ausführung das zu ändernde Verzeichnis erst herausgesucht werden muß. Aus diesem Grund starten beide Unterprogramme durch Aufruf von FINDVER.

Es gibt zwei Wege, über die ein Suchvorgang vorgenommen werden kann. Der eine ist, einen ungeordneten Haufen zu durchsuchen. Dies macht die Suche sehr langsam. Im

schlimmsten Fall muß die Routine alle Daten durchsehen, bevor sie den gesuchten Datensatz findet. Der Vorteil dieser Methode dagegen ist, daß eine Aktivierung der Suchroutine nicht immer nötig ist, wenn ein Verzeichnis hinzugefügt, gelöscht oder geändert wurde.

Wenn die Daten in einer bestimmten Form geordnet sind – numerisch oder alphabetisch beispielsweise – muß das Programm nur einen kleinen Teil der Daten durchsehen. Je größer die Datenmenge, desto effizienter wird das binäre Suchen. Wenn genug Daten in der Datei enthalten sind, kann der Sortiervorgang beschleunigt werden. Hierzu ist ein Weg, das erste und letzte Auftreten der Initialen des Vornamens zu lokalisieren.

Ein anderer Weg zur Beschleunigung einer Sortier-Routine wäre, eine Tabelle einzurichten, in der das erste Auftreten jedes Buchstabens des Alphabetes aufgeführt wird. Bei Änderungen muß diese Tabelle dann jeweils wieder angepaßt werden.

Die Aufgabe des Suchens und Sortierens ist eine der komplexesten in der Programmierung. Es gibt ganze Bücher zu diesem Thema. Wir werden nicht versuchen, die optimale Lösung für unser Programm zu finden, da dies von zahlreichen Faktoren abhängt, einschließlich der Menge der Verzeichnisse in der Datei und ob z. B. Diskettenlaufwerke angeschlossen sind oder nicht.

Im folgenden sehen Sie ein Programm in der Pseudo-Sprache zum Durchsuchen der Elemente im MODFLD$ Bereich. Die String-Variable KEY$ ist der Schlüssel für die Suche. Der Ausdruck „Key" bedeutet hier eine vorgegebene Gruppe von Zeichen, die zum Finden des Verzeichnisses bzw. der Verzeichnisse benötigt wird.

```
Frage nach zu suchendem Namen
LET KEY$=Name (der gesucht werden soll)
LET BTM=1
LET SUCHEN=0
LET TOP=GROSS
LOOP (SCHLEIFE) während (BTM < =
  TOP) AND (SUCHEN=0)
  LET MID=INT((BTM+TOP)/2)
  IF KEY$=MODFLD$(MID)
    THEN (DANN)
      PRINT NAMFLD$(MID)
      PRINT STRFLD$(MID)
      PRINT STDFLD$(MID)
      PRINT STAFLD$(MID)
      PRINT TELFLD$(MID)
      LET SUCHEN=1
    ELSE (SONST)
      IF KEY$ > MODFLD$(MID)
        THEN LET BTM=MID+1
        ELSE LET TOP=MID—1
      ENDIF
  ENDIF
ENDLOOP (ENDE DER SCHLEIFE)
IF SUCHEN=0 THEN PRINT "VERZEICHNIS
    NICHT GEFUNDEN"
END (ENDE)
```

Beachten Sie die Meldung des Programms, wenn es den gesuchten Datensatz nicht finden konnte. Diese Anweisung wird nur ausgeführt, wenn die Suche erfolglos blieb.

Die Suche mit Hilfe eines exakten Vergleichs ist allerdings nicht sehr praktisch, selbst wenn die gesuchten Daten vorhanden sind. Der Vergleich zwischen KEY$ und MODFLD$ ist völlig unflexibel. Selbst die geringste Abweichung führt dazu, daß die Daten nicht gefunden werden.

Doch es gibt Wege, um dies zu verhindern, auch wenn dadurch ein größerer Programmieraufwand entsteht und der ganze Vorgang etwas länger dauert. Die erste Verbesserung wäre, erst einmal die Familiennamen zu überprüfen. Jetzt macht es auch Sinn, daß im MODFLD$-Bereich der Name in der Form Familienname (Leerstelle) Vorname steht. Eine entsprechende Routine zum Vertauschen der Namen haben wir bereits an früherer Stelle in diesem Kurs beschrieben.

Ist das erste Auftreten des gesuchten Familiennamens erfolgreich, sollte die Routine nun die Vornamen durchsuchen und vergleichen. Gibt es den gesuchten Vornamen, ist alles in Ordnung – das gesuchte Verzeichnis ist gefunden. Gibt es den gesuchten Vornamen in der eingegebenen Form nicht, wird die Sache kompliziert und die Strategie zur Lösung muß sehr vorsichtig geplant werden. Wir könnten zum Beispiel alle Vornamen durchsuchen und, wenn ein exakter Vergleich erfolglos bliebe, nach einem annähernden Vergleich suchen.

Anstelle der Meldung „VERZEICHNIS NICHT GEFUNDEN" wie im oben gezeigten Programm wäre es besser, eine Meldung „EXAKTER VERGLEICH NICHT GEFUNDEN. VERSUCHEN SIE EINE ANNAEHERUNG? (J/N)?" auszugeben. Was bedeutet das Wort „Annäherung"? Ist Bobby eine Annäherung an Robert? Was ist mit Robrt? Beide Namen könnten ja tatsächlich eingegeben worden sein. Lassen Sie uns versuchen, die Bedeutung einer Annäherung zu definieren und dann ein entsprechendes Programm in BASIC zu entwickeln.

ROB oder RBRT?

Nehmen wir einmal an, der String im Speicher wäre ROBERT. Welche der folgenden beiden Alternativen ist eine größere Annäherung: ROB oder RBRT? Beim zweiten Wort haben wir vier richtige Buchstaben von insgesamt sechs, wobei beim ersten nur drei Buchstaben übereinstimmen. Andererseits befinden sich beim ersten Wort die drei Buchstaben in der richtigen Reihenfolge, wogegen beim zweiten nur zwei Buchstaben, diese aber immerhin in der richtigen Reihenfolge stehen.

Die Auswahl ist willkürlich. Wir wollen hier

der richtigen Reihenfolge von möglichst vielen Buchstaben mit dem Vergleichsstring den Vorzug geben. Wenn kein exakter Vergleich mit einem Teilstring möglich ist, versucht das Programm, die größtmögliche Anzahl übereinstimmender Buchstaben herauszufinden.

Das folgende Programm in der an BASIC angelehnten Pseudo-Sprache durchsucht die Strings eines Bereiches und beachtet dabei jeweils die ersten ‚n' Buchstaben, wobei ‚n' die Anzahl der Buchstaben im Schlüssel (KEY$) ist. Erweist sich der Vergleich als erfolglos, wird demnach eine entsprechende Meldung ausgegeben.

```
DIM BEREICH$(4)
FOR L=1 TO 4
READ BEREICH$(L)
NEXT L
DATA "ROBERT", "RICHARD", "ROBIANA",
  "ROBERTA"
LET KEY$="RON"
LET LKEY=LEN(KEY$)
LET SUCHEN=0
LOOP FOR INDEX=1 TO 4
  IF KEY$=LEFT$(BEREICH$(INDEX),LKEY)
    THEN PRINT "VERGLEICH IST"
      ;BEREICH$(INDEX)
      LET SUCHEN=INDEX
  ENDIF
ENDLOOP (ENDE DER SCHLEIFE)
IF SUCHEN=0
  THEN PRINT KEY$; "IST KEIN EXAKTER
      VERGLEICH VON IRGENDWEL-
      CHEN DER"
    PRINT "ERSTEN ";LKEY;" ZEICHEN"
```

Anschließend könnte das Programm nach LKEY-langen Gruppen von Zeichen suchen, startend mit dem jeweils zweiten Buchstaben in jeder Zeichenkette. Trifft wieder kein Vergleich zu, könnte der Vorgang mit dem jeweils dritten Zeichen usw. fortgesetzt werden. Wenn hierbei kein Erfolg zu verbuchen ist, könnte das Programm versuchen herauszufinden, welcher String die größtmögliche Anzahl an gemeinsamen Buchstaben mit KEY$ hat. Versuchen Sie als Übung, das Programm in dieser Form zu schreiben.

Die meisten Datenverwaltungsprogramme bieten die Möglichkeit, nach den ersten Zeichen in einem Feld zu suchen, ähnlich der hier beschriebenen Methode. Andere zeigen ein Verzeichnis an, wenn die gesuchte Zeichenkette irgendwo innerhalb des Feldes oder eines Verzeichnisses auftaucht. Die Möglichkeit, ein Zeichen nicht genau bestimmen zu müssen, ist besonders nützlich: K?R würde KIRSTEN oder KARL finden, nicht aber KATRIN. Die fortgeschrittenste Form des Suchens arbeitet phonetisch, so daß nach Eingabe des Namens Sibille bei der anschließenden Datenabfrage auch das ähnlich lautende Wort Sybille gefunden werden könnte.

BASIC-Dialekte

ZX81 SPECTRUM

Dies ist das Listing des Programms, das die Reihenfolge des Vor- und Familiennamens miteinander vertauscht:

```
 100 CLS
 200 PRINT "GEBEN SIE DEN
     NAMEN IN DER FORM"
 300 PRINT "VORNAME
     FAMILIENNAME EIN"
 400 PRINT "Z.B. OSWIN HEIDER"
 500 INPUT "EINGABE DES
     NAMENS";N$
 600 GOSUB 9500
 700 PRINT "NAME IN
     STANDARDFORM IST"
 800 PRINT N$
1000 STOP
9500 REM UNTERROUTINE ZUM
     AUSTAUSCH DER NAMEN
9520 GOSUB 9600
9540 IF P=0 THEN RETURN
9560 LET N$=S$+", "+F$
9580 RETURN
9600 REM UNTERROUTINE ZUM
     EINFUEGEN EINER
     LEERSTELLE IN N$
9620 LET N=LEN(N$)
9630 LET P=0
9640 FOR K=1 TO N
9650 IF N$(K)=" " THEN LET P=K
9655 IF N$(K)=" " THEN LET K=N
9660 NEXT K
9670 IF P=0 THEN RETURN
9680 LET F$=N$( TO P-1)
9700 LET S$=N$(P+1 TO )
9720 RETURN
```

LEFT$

RIGHT$

MID$

Beim Commodore 64, VC 20, Oric und Lynx ersetzen Sie die Zeilen 9600 bis 9720 des Spectrum Listings wie folgt:

```
9600 REM
9620 LET N=LEN(N$)
9630 LET P=0
9640 FOR K=1 TO N
9650 IF MID$(N$,K,1)=" "
     THEN LET P=K:LET K=N
9660 NEXT K
9670 IF P=0 THEN RETURN
9680 LET F$=LEFT$(N$,P-1)
9700 LET S$=RIGHT$(N$,N-P)
9720 RETURN
```

Beim Dragon 32 und dem Acorn B ersetzen Sie die Zeilen 9600 bis 9720 des Spectrum Listings durch die folgenden Zeilen:

INSTR

```
9600 REM
9620 LET N=LEN(N$)
9640 LET P=INSTR(N$," ")
9670 IF P=0 THEN RETURN
9680 LET F$=LEFT$(N$,P-1)
9700 LET S$=RIGHT$(N$,N-P)
9720 RETURN
```

Wie wir bereits zuvor angemerkt haben, ist INSTR eine nützliche Funktion, gerade dann, wenn man mit Datensätzen arbeitet. Wenn Ihr Computer diese Funktion kennt, können Sie sicher eine bessere Suchroutine entwickeln.

INPUT

Beim Acorn B ersetzen Sie Zeile 500 des Spectrum Listings wie folgt:

```
500 INPUT "NAMEN
    EINGEBEN",N$
```

Bits aus der Büchse

Winchester-Laufwerke bieten dem Computer-Besitzer weit mehr Speicherplatz und außerdem sehr kurze Zugriffszeiten.

Obwohl Heimcomputer während der letzten Jahre in vielen Bereichen an die Leistung kleinerer kommerzieller Rechner heranreichen, liegen sie in der Plattenspeicher-Kapazität immer noch weit zurück. Bei einem Heimcomputer ist eine Disketten-Kapazität von 100 KByte zufriedenstellend. Aber ein Bürorechner benötigt ein Vielfaches an Speicherplatz. Ein Stapel Disketten, auf denen sämtliche Betriebsdaten nicht systematisch verteilt sind, ist in diesem Fall keine Lösung. Um den schnellen Zugriff auf größere Datenmengen zu gewährleisten, werden große, starre Magnetplatten mit speziellen Laufwerken eingesetzt. IBM baute vor rund 20 Jahren die ersten Laufwerke dieser Art. Da die Kapazität 30 Megabyte je Plattenseite betrug, wurden sie kurz "30-30" genannt – genau wie das berühmte Winchestergewehr. Später entwickelte sich daraus der Begriff „Winchesterplatte".

Hohe Spurendichte

Bei Winchesterlaufwerken werden unflexible Platten (hard disks) verwendet: Das erlaubt die Erhöhung der Spurendichte auf mehrere Hundert tpi (tracks per inch = Spuren je Zoll). Zum Vergleich: Eine normale Floppy weist 96 tpi auf. Die Technologie ist inzwischen so weit fortgeschritten, daß selbst 50 Megabyte Speicherkapazität in einem Gehäuse von der Größe eines 5¼-Zoll-Floppylaufwerks nicht ungewöhnlich sind.

Der Trend zu den Festplatten hat auch für den Heimcomputer-Benutzer Vorteile, und zwar in Form der „halbstarren" Microdisketten mit 3½ Zoll oder 3 Zoll Durchmesser von Sony bzw. Hitachi mit erhöhter Kapazität.

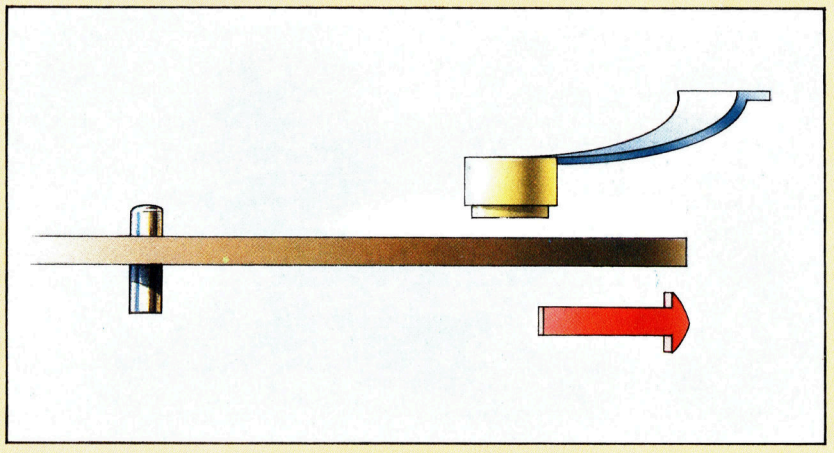
Die Zunahme der Aufzeichnungsdichte erzeugte jedoch neue Probleme. Die Ansprüche an Genauigkeit und Geschwindigkeit der Kopfpositionierung erforderten eine neue Antriebslösung für die Magnetköpfe. Bei der „Voice Coil"-Winchester benutzt man statt eines Schrittmotors mit Spiralkurvenschaltung und Hebelübertragung einen Linearantrieb.

„Fliegende" Köpfe

Die Magnetköpfe werden ähnlich wie die Membran eines Lautsprechers unmittelbar durch die elektromagnetische Kraftwirkung einer Schwingspule (voice coil) über die Plattenoberfläche bewegt.

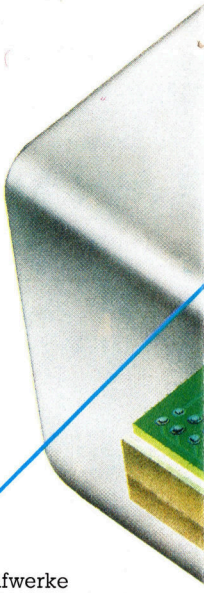
Die Köpfe „fliegen" auf einem Luftkissen über die Platte, ohne sie zu berühren. Daher kann die Drehzahl gegenüber einer Diskette auf das Zehnfache gesteigert werden (über 3000 Upm). Voraussetzung für dieses Verfahren ist eine luftdichte Versiegelung des Gerätes, um Schäden durch Fremdkörper zu vermeiden. Daher verwendete man bislang nicht auswechselbare Festplatten, neuerdings aber auch Wechselplatten in verschlossenen Cartridges. Diese geben den Zugang für den Magnetkopf erst frei, wenn sie in das Laufwerk eingesetzt sind. Um Staubeintritt zu vermeiden, werden die Cartridges unter leichtem Überdruck gehalten, und die Luftzufuhr dafür erfolgt über Microfilter.

Aus Festplatten kann man Stapel bilden und so z. B. mit zwei 5 MByte-Platten ein 10 MByte Winchester-Laufwerk aufbauen. Die Organisation des Speicherplatzes wird durch Aufteilung in kleinere Sektionen erreicht. Um nun die Kompatibilität mit diskettenorientierter Soft-



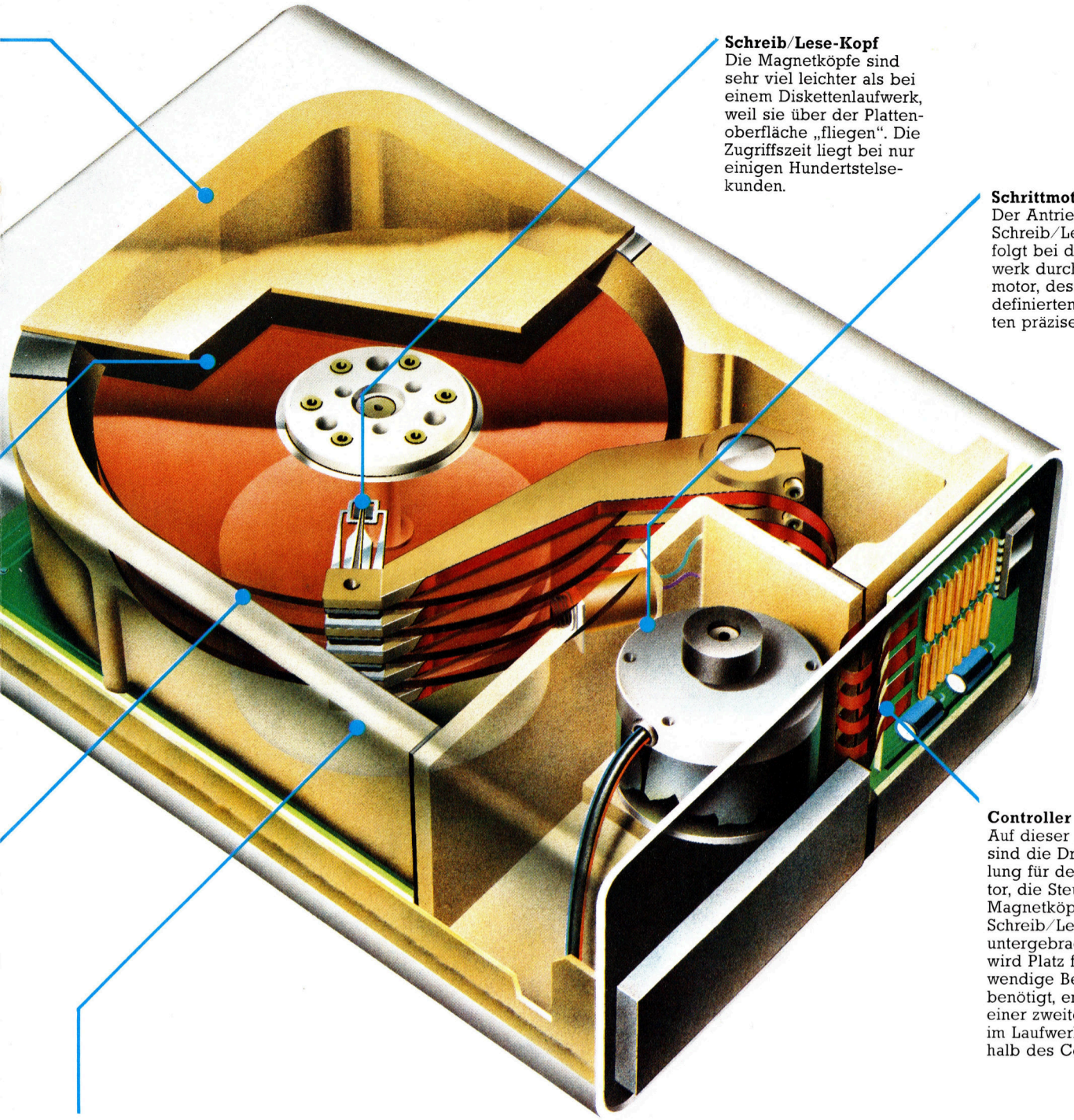
Schwebende Köpfe
Während der Schreib/Lese-Kopf bei einer Diskette auf der Magnetschicht gleitet, „fliegt" er bei einer Winchesterplatte in geringem Abstand über die Platte. Bei der hohen Drehzahl wird die angrenzende Luftschicht mitgerissen und dieser Luftstrom hebt den Kopf. Wenn es durch Fremdkörper zu der berüchtigten „Kopflandung" kommt, wird die magnetische Oberfläche der Platte zerstört und Daten gehen verloren.

Gehäuse
Winchester-Laufwerke sind meist in einem stabilen Spritzgußgehäuse untergebracht, weil sonst die nötige Genauigkeit in der Justierung der Bauteile nicht einzuhalten wäre. Das erklärt mit das erhebliche Gewicht dieser Geräte.



Platten
Winchester-Laufwerke mit höherer Kapazität haben mehrere Platten auf einer Spindel. Die Schreib/Lese-Köpfe sind miteinander verbunden; die erfaßten Spuren nennt man „Zylinder".

Dichtfläche
Die Laufwerkmechanik ist luftdicht versiegelt, um zu verhindern, daß Staub- oder Rauchpartikel zwischen Platte und Magnetkopf geraten und Schäden hervorrufen.



Schreib/Lese-Kopf
Die Magnetköpfe sind sehr viel leichter als bei einem Diskettenlaufwerk, weil sie über der Plattenoberfläche „fliegen". Die Zugriffszeit liegt bei nur einigen Hundertstelsekunden.

Schrittmotor
Der Antrieb des Schreib/Lese-Kopfes erfolgt bei diesem Laufwerk durch einen Schrittmotor, dessen Rotation in definierten Winkelschritten präzise steuerbar ist.

Controller
Auf dieser Leiterplatte sind die Drehzahlregelung für den Spindelmotor, die Steuerung der Magnetköpfe und die Schreib/Leseverstärker untergebracht. Außerdem wird Platz für das aufwendige Betriebssystem benötigt, entweder auf einer zweiten Leiterplatte im Laufwerk oder innerhalb des Computers.

Spindelmotor
Die Spindel sitzt direkt auf der Achse eines Gleichstrommotors, auf der sich außerdem noch ein Generator befindet. Dessen Ausgangsspannung wird mit einem Sollwert verglichen; etwaige Abweichungen wirken auf die Stromversorgung des Spindelmotors ein. Dieser Regelkreis stellt eine gleichbleibende Plattendrehzahl sicher.

ware zu gewährleisten, die auf den Umgang mit normalen Floppy Disks ausgelegt ist, enthält jede einzelne Sektion die Kapazität einer Floppy. Auf diese Weise kann der Computer die verschiedenen Sektionen genauso wie separate Floppies ansprechen.

Micro-Platten

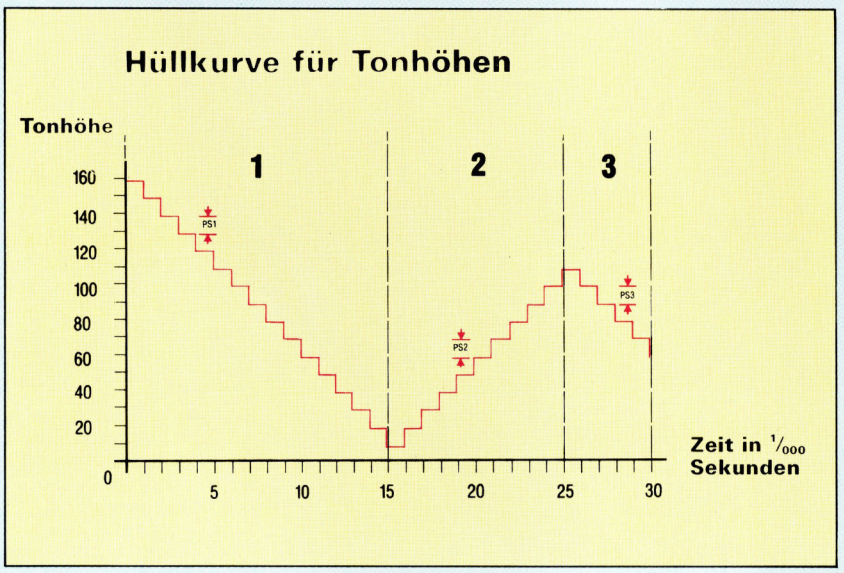
Bei der Formatierung einer Winchesterplatte erkennt das Betriebssystem einzelne defekte Spuren mit fehlerhafter Magnetschicht (bad sector) und sperrt sie für den Gebrauch. Dagegen bewirkt der „bad sector" bei einer gewöhnlichen Diskette, daß der gesamte Inhalt unlesbar wird. Bei der Winchesterplatte verschenken Sie somit nur einige Spuren – und die paar Hundert Byte sind nicht entscheidend, wenn fünf Millionen Byte verfügbar bleiben.

Mit den Fortschritten der Computertechnologie werden auch die Winchesterplatten immer kleiner – schon jetzt gibt es eine 5 Megabyte-Micro-Winchesterplatte. Angesichts solcher Entwicklungen ist ein Heimcomputer für 2000 Mark, an den ein externer Speicher mit 10 Megabyte angeschlossen werden kann, nicht mehr so fern. Eine Hard-Disk für Heimcomputer wurde bereits vorgestellt.

Sphären-klänge

Mit dem Befehl ENVELOPE des Acorn B lassen sich Klänge fast uneingeschränkt steuern.

In einer der letzten Folgen haben wir den SOUND-Befehl und seinen Einsatz auf dem Acorn B behandelt. Die vielseitigen Klangmöglichkeiten dieses Computers lassen sich jedoch nur mit dem Befehl ENVELOPE voll ausschöpfen. Über ENVELOPE können vier Sounds so genau festgelegt werden, daß sie



an das Klangbild des Originalinstruments heranreichen. Auch lassen sich damit Klangeffekte für Spiele herstellen, die z. B. Explosionen oder Gewehrfeuer simulieren.

Der Befehl ENVELOPE hat folgendes Format:
ENVELOPE N,T,PS1,PS2,PS3,NS1,NS2,NS3,
AR,DR,SR,RR,FAL,FDL

Der erste Parameter N setzt die Nummer der Hüllkurve und legt sie damit auch für verwandte Befehle wie SOUND und SOUND & fest. Es stehen vier Hüllkurven zur Verfügung, die an die Stelle der gleichbleibenden Lautstärke der Variablen V (die negative Werte von 0 bis −15 annehmen kann) gesetzt werden können.

T (0 bis 127) & (128 bis 255)

steuert den Zeitablauf des Befehls. Über diese Variable wird die Dauer eines einzelnen „Schrittes" der Hüllkurve in Hundertstelsekunden festgelegt. T = 5 bedeutet, daß jeder Schritt der Hüllkurve die Länge von fünf Hundertstelsekunden hat. Addiert man zu der Schrittdauer die Zahl 128, wird damit die automatische Wiederholung der Tonhöhenkurve ausgeschaltet. T = 5 + 128 = 133 bedeutet eine Schrittdauer von fünf Hundertstelsekunden, während die Tonhöhenkurve für jeden gespielten Ton nur einmal abläuft.

Der Begriff „Hüllkurve für Tonhöhen" mag ein wenig verwirren, da wir diese Bezeichnung zuvor schon für die Lautstärke verwandt haben. In diesem Fall sind jedoch die Schwankungen der Tonhöhe während der Dauer eines Tones gemeint. Für den musikalischen Bereich hat diese Möglichkeit – außer für ein Vibrato – nur wenig Bedeutung. Als Klangeffekt entsteht jedoch ein interessanter Trillerton. Unser Diagramm zeigt, daß die Hüllkurve für Tonhöhen in drei Bereiche unterteilt ist. Jeder dieser Bereiche kann über die entsprechenden Variablen PS und NS individuell strukturiert werden:

PS1,PS2 & PS3 (—128 bis 128)

PS legt die Größe der Schritte fest, um die die Tonhöhe variieren soll. Dabei bestimmt der SOUND-Befehl am Anfang eines Tones die gewünschte Höhe. PS1 legt nun die Tonhöhenveränderung pro Schritt (positiv oder negativ, d. h. aufwärts oder abwärts) für den ersten Bereich fest, PS2 für den zweiten und PS3 für den dritten. Ähnlich wie bei SOUND ist die Schrittgröße der vierte Teil eines Halbtones.

NS1,NS2 & NS3 (0 bis 255)

NS bestimmt die Anzahl der Schritte pro Tonbereich. In Verbindung mit der Variablen PS läßt sich so die Geschwindigkeit der Tonhöhenschwankungen für jeden Bereich einzeln angeben und damit auch die Dauer der gesamten Hüllkurve festlegen. In unserem obigen Beispiel nehmen PS und NS folgende Werte an:

T=1 PS1 = —10 NS1 = 15
PS2 = +10 NS2 = 10
PS3 = —15 NS3 = 5

Die Tonhöhe wurde über SOUND = 160 gesetzt. Der Befehl lautet daher:

ENVELOPE 1,1,—10,10,
—15,15,10,5,0,0,0,0,0,0

Die Dauer der Hüllkurve ist mit (NS1 + NS2 + NS3) × T angegeben, in diesem Fall also (15 + 10 + 5) × 1 = 0,3 Sekunden. Die Hüllkurve der Tonhöhe wird in diesem Fall während der Dauer des Klanges wiederholt, die Wiederholung läßt sich aber durch das Setzen der Variable T abschalten.

Rasante Grafik

Sprites auf dem Commodore 64.

Eine der interessantesten Fähigkeiten des Commodore 64 ist der Einsatz von Sprites. Diese haben den gleichen Aufbau wie frei definierbare Zeichen, sind jedoch mit 21 Reihen zu je 24 Pixeln (bei hochauflösender Darstellung) pro Sprite weitaus größer. Sprites lassen sich in Pixelschritten über den Bildschirm bewegen, da sie nicht wie normale Zeichen behandelt werden, die sich nur in Schritten von acht Pixeln verschieben lassen. Bis zu acht Sprites lassen sich gleichzeitig auf dem Bildschirm darstellen, wobei jedes individuell programmiert werden kann.

Ein Sprite läßt sich ebenso leicht konstruieren wie ein Zeichen, das aus acht mal acht Pixeln aufgebaut ist. Zur Speicherung seines Bitmusters belegt es jedoch einen Block von 63 Bytes. Jedes Sprite hat einen „Pointer", d. h. eine Variable, die die erste Speicheradresse des Spriteblocks enthält. Dieser Pointer kann durchaus mehreren Sprites gleichzeitig zugeordnet sein, wobei die Sprites dann natürlich identisch aussehen. Die Form eines Sprites läßt sich durch den Tausch des Pointers aber auch verändern.

Jedes Sprite kann eine von 16 verfügbaren Farben annehmen. Mehrfarbige Sprites sind ebenfalls möglich, allerdings wird dabei die horizontale Auflösung auf die Hälfte reduziert. Sprites können horizontal, vertikal oder in beiden Richtungen bis auf das Doppelte ihrer ursprünglichen Größe vergrößert werden. Die größte Ausdehnung eines Sprites liegt bei 48 × 42 Pixeln, wobei auch hier die Auflösung in Richtung der Sprite-Ausdehnung um die Hälfte reduziert wird. Sprites lassen sich in Pixelschritten auch über den Bildschirmrand hinaus bewegen.

```
90 REM ** SPRITES 64 **
100 PRINT"□"
110 V=53248
120 REM----READ SPRITE DATA----
130 FORI=12288TO12350:READA:POKEI,A:NEXT
140 FORI=12352TO12414:READA:POKEI,A:NEXT
150 FORI=832TO894:READA:POKEI,A:NEXT
160 FORI=896TO958:READA:POKEI,A:NEXT
170 FORI=12416TO12478:READA:POKEI,A:NEXT
180 REM----EXPAND SPRITES----
190 POKEV+23,7:POKEV+29,7
200 REM----COLOR SPRITES----
210 POKEV+39,10:POKEV+40,10
220 POKEV+41,1
230 REM--MULTI COLOR--
240 POKEV+28,3:POKEV+37,7:POKEV+38,9
300 REM----MEMORY POINTERS----
310 POKE2040,192:POKE2041,193:POKE2042,194
320 REM----SET Y COORDS----
330 Y0=150:Y1=Y0+42:Y2=Y0+34
340 POKEV+1,Y0:POKEV+3,Y1:POKEV+5,Y2
400 REM----TURN ON SPRITES----
410 POKEV+21,7
500 GOSUB3000:REM OMIT IF NO SUBROUTINE
1000 X0=20
1010 POKE2040,13:POKE2041,14
1020 POKEV,X0:POKEV+2,X0:POKEV+4,X0+48
1030 FORI=1TO500:NEXT
1040 POKE2040,192:POKE2041,193
1050 X0=X0+5
1060 POKEV,X0:POKEV+2,X0:POKEV+4,X0+48
1070 FORI=1TO500:NEXT
1080 X0=X0+5
1090 IFX0>200THEN1110
1100 GOTO1010
1110 FORJ=1TO10
1120 POKE2040,13:POKE2041,14
1130 FORI=1TO50:NEXT
1140 POKE2040,192:POKE2041,193
1150 FORI=1TO50:NEXT
1160 NEXT
1170 GOTO1170
9000 REM----DATA WOMAN TOP----
9010 DATA0,0,0,0,21,0,0,21,0,0,22,0,0,86,0
9020 DATA0,86,0,0,86,0,0,40,0,0,252,0
9030 DATA15,255,0,255,255,0,255,255,0
9040 DATA195,255,0,195,255,0,195,243,254
9050 DATA207,243,254
9060 DATA143,240,0,143,252,0,15,252,0
9070 DATA15,252,0,15,252,0
9100 REM----DATA WOMAN BOTTOM----
9110 DATA15,252,0,15,252,0,15,252,0
9120 DATA15,252,0,5,84,0,5,84,0,5,84,0
9130 DATA5,84,0,10,40,0,234,40,0,234,40,0
9140 DATA234,40,0,192,40,0,192,40,0,0,40,0
9150 DATA0,40,0,0,63,0,0,63,0,0,0,0,0,0,0
9160 DATA0,0,0
9200 REM----DATA WOMAN TOP #2----
9210 DATA0,0,0,0,20,32,32,85,32,32,105,48,48,105,48
9220 DATA48,105,48,48,105,48,48,48,48,48,252,48
9230 DATA63,255,240,63,255,240,63,255,0
9240 DATA3,255,0,3,255,0,3,240,0
9250 DATA15,240,0
9260 DATA15,240,0,15,252,0,15,252,0
9270 DATA15,252,0,15,252,0
9300 REM----DATA WOMAN BOTTOM #2----
9310 DATA15,252,0,15,252,0,15,252,0
9320 DATA15,252,0,5,84,0,5,84,0,5,84,0
9330 DATA5,84,0,10,40,0,58,168,0,58,168,0
9340 DATA58,0,0,58,0,0,10,0,0,10,0,0
9350 DATA10,0,0,15,192,0,15,192,0,0,0,0,0,0,0
9360 DATA0,0,0
9400 REM----TROLLEY DATA----
9410 DATA192,0,0,224,0,0,118,0
9420 DATA0,55,192,0,32,60,0,53
9430 DATA87,240,32,0,15,53,85,85
9440 DATA32,0,3,53,85,85,0,0,3
9450 DATA21,85,85,31,255,255,24,0
9460 DATA0,12,0,0,12,0,0,31,255
9470 DATA240,31,255,255,1,0,2,7
9480 DATA0,14,7,0,14
```

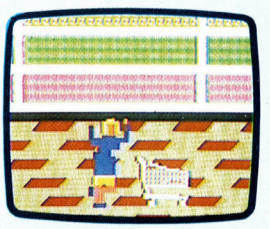
Zusammenstöße

Kreuzen sich die Wege zweier Sprites, scheint sich das eine über das andere hinwegzubewegen. Dabei werden durch Löcher des vorderen Sprites Teile des dahinterliegenden sichtbar. Der Vorrang eines Sprites vor anderen läßt sich programmieren und ergibt interessante dreidimensionale Effekte. Jedem Sprite wird dabei eine Nummer zwischen 0 bis 7 zugeordnet, wobei niedrigere Nummern vor höheren Vorrang haben, d. h. sich über diese hinwegbewegen können. Normalerweise überdecken Sprites auch die normalen Bildschirmelemente (Zeichen oder Grafik), es gibt aber auch die Möglichkeit, sie hinter einzelnen Elementen verschwinden zu lassen.

Treffen zwei Sprites aufeinander, so wird dieser Zusammenstoß in einem speziellen Speicher angezeigt, der sich mit dem Befehl PEEK abfragen läßt. Dabei kann auch festgestellt werden, welche Sprites daran beteiligt sind. Ein weiteres Register gibt Auskunft darüber, mit welchen Hintergrundelementen die Sprites zusammengestoßen sind.

Mit all diesen Möglichkeiten lassen sich auch in BASIC Spiele mit schnellen Bewegungsabläufen schreiben. Leider gibt es keine Spezialbefehle zur Spritesteuerung, und die entsprechenden Funktionen können nur über POKE aufgerufen werden.

Simon's BASIC
Etwa 200 Mark kostet die Zusatzcartridge, mit denen die Steuerung der Sprites und der Aufbau von hochauflösender Grafik in BASIC auf dem Commodore 64 stark vereinfacht wird. Mitgeliefert wird ein Handbuch, das die 114 zusätzlichen Befehle ausführlich erläutert. Darin enthalten sind Befehle zum Aufrufen der hochauflösenden Grafik, zur Auswahl von Vorder- und Hintergrundfarben und zum Zeichnen von Kreisen, Ellipsen, Rechtecken und geraden Linien.



Die links aufgeführten Programmzeilen können in das zuvor gezeigte Supermarktprogramm eingefügt werden. Dieser Programmabschnitt setzt zwei vergrößerte mehrfarbige Sprites zur Darstellung einer menschlichen Form ein und einen weiteren gedehnten Sprite für den Einkaufswagen. In dem Programm werden die Spritepointer der „Kundin" ständig verändert, so daß sie tänzelnd den Bildschirm zu durchqueren scheint. Soll das Supermarktprogramm als Unterroutine eingesetzt werden, muß die Zeile 3270 in 3270 RETURN geändert werden.

Drachentöter

Vier Sprite-Bitraster für das Commodore-LOGO sind Ihnen inzwischen bekannt. Als Ergänzung zu den Spritemöglichkeiten wird hier der Algorithmus „Verfolgungskurve" anhand des Drei-Käfer-Problems erläutert und das Spiel „Drache gegen Ritter" entwickelt.

Im nachstehenden Spielprogramm werden LOGO-Sprites verwendet. Sie steuern einen Drachen, der versucht, eine Stadt zu erreichen und zu zerstören. Die Verteidigung der Stadt obliegt einem fliegenden (vom Computer gesteuerten) Ritter, der den Drachen zu töten versucht. Die Bewegung des Drachens erfolgt durch den Joystick. Gelingt es, dem Ritter auszuweichen und nahe genug an die Stadt heranzukommen, löst sich diese durch den Atem des Drachens in Flammen auf.

Voraussetzung zum Spielen sind das Einlesen des SPRITES-File, die Definition der Formen, die Eingabe der Befehle und schließlich die Eingabe GAME. Nach Abschluß dieser Vorbereitungen ruft die Prozedur GAME das Hauptprogramm PLAY. PLAY bewegt abwechselnd den Drachen und den Ritter und prüft, ob der Drache die Stadt erreicht oder der Ritter den Drachen getötet hat.

Die verfügbaren Farb-Befehle sind sehr einfach. Die Hintergrundfarbe wird mittels BACKGROUND, gefolgt von einer Farbnummer, gewählt. Die Festlegung einer Spritefarbe (und die Farbe der Linie bei PENDOWN) erfolgt durch PENCOLOR. Die Farben werden durch Zuweisung mit dem Befehl INIT.VARIABLES definiert, zum Beispiel mit PENCOLOR :RED.

Bei der Prozedur PLAY wird die Zeile

IF HIT? THEN DRAGON. DESTROYED

genutzt, um festzustellen, ob der Ritter den Drachen getroffen hat. Der Befehl HIT? verdeutlicht, wie mit LOGO eigene Bedingungsabfragen umgesetzt werden können. Als Auf-

Ritter und Drachen

Spielablauf

Stadt in Flammen

Niederlage des Drachens

Ritter gegen Drache

```
TO GAME
   INIT. VARIABLES
   SET.SCREEN
   PLAY
END

TO INIT.VARIABLES
   MAKE "FLAME 1
   MAKE "FLAME1 2
   MAKE "DRAGON 3
   MAKE "KNIGHT 4
   MAKE "CITY 5
   MAKE "CITY1 6
   MAKE "RED 2
   MAKE "BLACK 0
   MAKE "BLUE 6
   MAKE "ORANGE 8
   MAKE "YELLOW 7
END

TO SET.SCREEN
   DRAW
   FULLSCREEN
   BACKGROUND :BLUE
   TELL 0
   PU
   HT
   TELL :FLAME1
   HT
   HT
   TELL :FLAME
   HT
   POSITION :DRAGON 100 100 :RED
   BIGX BIGY
   POSITION :CITY ( - 52) ( - 80) :BLACK
   BIGX BIGY
   POSITION :CITY1 ( - 100 ) ( - 80 ) :BLACK
   BIGX BIGY
   TELL 4
   PU
   POSITION :KNIGHT ( 100 ) ( - 100 ) :YELLOW
   SMALLX SMALLY
END

TO PLAY
   DRAGON.MOVE
   IF DISTANCE :DRAGON :CITY < 50 THEN CITY.
      DESTROYED STOP
   IF DISTANCE :DRAGON :CITY1 < 50 THEN CITY.
     DESTROYED STOP
   KNIGHT.MOVE
   IF HIT? THEN DRAGON.DESTROYED STOP
   PLAY
END

TO DRAGON.MOVE
   TELL :DRAGON
   MOVEJOY JOYSTICK 1
   FD 10
END

TO MOVEJOY :DIR
   IF :DIR < 0 STOP
   SETH :DIR * 45
END

TO DISTANCE :A :B
   TELL :A
   MAKE "X1 XCOR
   MAKE "Y1 YCOR
   TELL :B
```

gabe erscheint ein "TRUE- oder "FALSE-Wert, der in die IF-Statements eingelesen wird. Das Ergebnis "TRUE hat die Ausführung der davon abhängigen Aktion zur Folge.

Mit dem Befehl JOYSTICK wird 0 oder 1 als Eingabe aufgerufen (korrespondierend zu den Ports 1 und 2). Steht der Joystick auf Mitte, beträgt der Wert –1; wird er nach vorne gedrückt, ist der Wert 0; 1, wenn er im Winkel von 45 Grad steht; 2 beim 90 Grad-Winkel usw. bis 7.

Explosionen und ähnliche Effekte sind mit Hilfe der Sprites einfach zu erzeugen. So wird z. B. durch eine flackernde Kontur über dem zerstörten Objekt eine Explosion dargestellt. Das FLAME-Sprite erhält die höchste Priorität, so daß es vor den anderen Sprites erscheint.

Zur Kontrolle des Ritters gebraucht der Computer eine einfache Strategie: Der Ritter bewegt sich direkt auf den Drachen zu.

Wie könnte man die Defensiv-Strategie des Ritters verbessern? Eine Möglichkeit wäre, die Bewegungsgeschwindigkeit des Ritters etwa von 10 auf 11 zu erhöhen. Ein anderer, besserer Weg wäre der, den Ritter so zu programmieren, daß er sich entsprechend der Richtung bewegt, die der Drache gerade eingeschlagen hat und ihn auf der Linie Drache-Stadt zu plazieren.

Das Drei-Käfer-Projekt

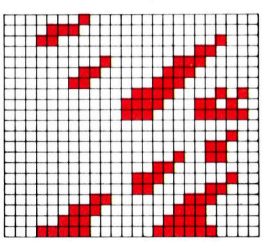
The bugs start out from the corners of a triangle:

```
TO BUGS
   SETUP MOVE.BUGS
END

TO SETUP
   DRAW FULLSCREEN TELL 0 HT PU SETXY
   ( – 100 ) ( – 100 ) TRI 200 POSITION 1 ( – 100 )
   ( – 100 ) POSITION 2 0 73 POSITION 3 100
   (–100)
END

TO TRI :SIDE
   PD REPEAT 3 [FD :SIDE RT 120] PU
END

TO POSITION :NO :X :Y
   TELL :NO SETSHAPE 3 PU SETXY :X :Y PD ST
END

TO MOVE.BUGS
   FOLLOW 1 2 FOLLOW 2 3 FOLLOW 3 1
   MOVE.BUGS
END

TO FOLLOW :A :B
   TELL :B MAKE "X XCOR MAKE "Y YCOR TELL :A
   SETH TOWARDS :X :Y FD 10
END
```

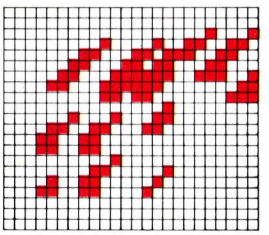
Sprite Editor

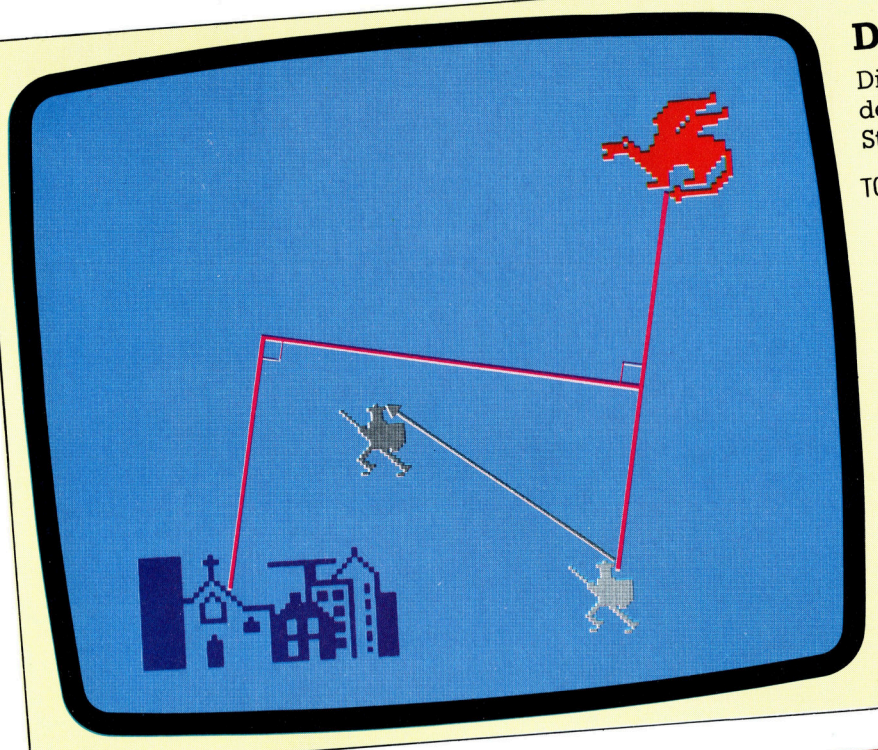
```
   MAKE "X2 XCOR
   MAKE "Y2 YCOR
   OUTPUT SQRT ( ( :X1 – :X2 ) * ( :X1 – :X2 ) +
      ( :Y1 – :Y2 ) * ( :Y1 – Y2 ) )
END

TO CITY.DESTROYED
   TELL :CITY
MAKE "X XCOR
   MAKE "Y YCOR
   FLASH :X :Y :ORANGE
   TELL :CITY1
   MAKE "X2 XCOR
   MAKE "Y2 YCOR
   FLASH1 :X2 :Y2 :ORANGE
   HIDE.SPRITE
   SPLITSCREEN
   REPEAT 3 [PRINT "]
   PRINT [CITY DESTROYED!]
END

TO FLASH :X :Y :COLOR
   TELL :FLAME
   PENCOLOR :COLOR
   SETXY :X :Y
   ST
   REPEAT 6 [SMALLX SMALLY WAIT BIGX BIGY
      WAIT]
END

TO FLASH1 :X2 :Y2 :COLOR
   TELL :FLAME1 PENCOLOR :COLOR
SETXY :X2 :Y2
   ST REPEAT 6 [SMALLX SMALLY WAIT BIGX BIGY
      WAIT]
END

TO KNIGHT.MOVE
   TELL :DRAGON
   MAKE "X XCOR
   MAKE "Y YCOR
   TELL :KNIGHT
   SETH TOWARDS :X :Y
   FD 10
END

TO HIT?
   TELL :DRAGON
   IF TS? THEN OUTPUT "TRUE
   OUTPUT "FALSE
END

TO DRAGON.DESTROYED
   TELL :DRAGON
   MAKE "X XCOR
   MAKE "Y YCOR
   FLASH :X :Y :BLACK
   HIDE.SPRITE
   SPLITSCREEN
   REPEAT 3 [PRINT "]
   (PRINT [DRAGON KILLED AT A DISTANCE OF]
      DISTANCE :DRAGON :CITY )
END

TO HIDE.SPRITE
   TELL :FLAME HT
   TELL :FLAME1 HT
   TELL :CITY HT
   TELL :CITY1 HT
END

TO WAIT
   REPEAT 100 [ ]
END
```

Flamme 1



Flamme 2





Die Strategie

Die beste Strategie (für beide Figuren) ist, den angrenzenden, senkrecht über der Stadt liegenden Sektor anzusteuern:

```
TO KNIGHT.MOVE
   TELL :DRAGON MAKE "DX XCOR MAKE "DY
   YCOR TELL :KNIGHT MAKE "KX XCOR MAKE
   "KY YCOR TELL :CITY MAKE "CX XCOR MAKE
   "CY YCOR
   MAKE "SX ( :DX + :KX ) / 2
   MAKE "SY ( :DY + :KY ) / 2
   MAKE "VX ( :DY – :KY )
   MAKE "VY ( :KX – :DX )
   MAKE "FACT ( :VX * (:CD – :SX ) + :VY *
   ( :CY – :SY ) ) / ( ( :VX * :VX) * ( :VX * :VX ) +
   ( :VY * :VY ) * ( :VY * :VY ) )
   MAKE "X :SX + :FACT * :VX
   MAKE "Y :SY + :FACT * :VY
   TELL :KNIGHT SETH TOWARDS :X :Y
   FD 10
END
```

Verbesserte Steuerung

Hier ist eine Möglichkeit, wie Sie die Bewegung des Ritters optimieren können:

```
TO KNIGHT.MOVE
   TELL :DRAGON MAKE "X XCOR MAKE "Y YCOR
   TELL :CITY
   MAKE "BEARING TOWARDS :X :Y
   TELL :KNIGHT SETH 270 + :BEARING
   IF XCOR < :X THEN LEFT 180
   FD 10
END
```

Abkürzungen

BACKGROUND	BG
PENCOLOR	PC

Steuerung über die Tastatur

Steuerung des Drachens durch Eingaben:

```
TO DRAGON.MOVE
   TELL :DRAGON MOVE READKEY FD 10
END

TO MOVE :DIR
   IF :DIR = "W THEN SETH 0
   IF :DIR = "S THEN SETH 90
   IF :DIR = "Z THEN SETH 180
   IF :DIR = "A THEN SETH 270
END

TO READKEY
   IF RC? THEN OUTPUT READCHARACTER
   OUTPUT "
END
```

LOGO-Dialekte

Weder das Spectrum-, noch das Apple-LOGO unterstützt die Sprite-Grafik. Beim Atari-LOGO sind folgende Unterschiede zu beachten:

1) TS? gibt es in dieser LOGO-Version nicht. Löschen Sie die Zeile IF HIT? . . . in der PLAY-Prozedur, und geben Sie statt dessen am Ende der SET.SCREEN-Prozedur diese Befehle ein:

```
WHEN TOUCHING :DRAGON :KNIGHT
[DRAGON.DESTROYED STOP]
```

2) Die Angaben BIGX, BIGY, SMALLX und SMALLY müssen weggelassen werden, da es in dieser Version keine Befehls-Äquivalente gibt.

3) BACKGROUND ist durch SETBG und PENCOLOR durch SETPC zu ersetzen. Die Farbangaben unterscheiden sich ebenfalls.

3) Auch der Befehl TOWARDS ist beim Atari-LOGO nicht vorgesehen. Ersetzen Sie die Zeilen in der FOLLOW-Prozedur wie folgt:

```
SETH TOWARDS :X :Y
FD 10
```

KNIGHT.MOVE muß so eingegeben werden:

```
MAKE"FRAC 10 / (SQRT ((XCOR—:X) *
   (XCOR — :X) + (YCOR — :Y) * (YCOR —
   :Y)))
SETPOS LIST (XCOR + (:X—XCOR)* :FRAC))
   (YCOR+ (:Y — YCOR)* :FRAC)
```

Action ins Spiel

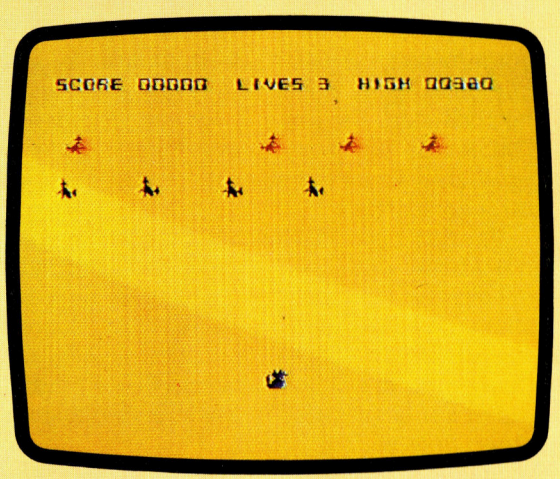
Jahr für Jahr werden Millionen Spielprogramme für Microcomputer verkauft. Dieses Marktpotential für fertige Programme veranlaßte vorausdenkende Software-Anbieter, Spiel-Generatoren zu entwickeln. Überraschend ist die Ähnlichkeit der angebotenen Spiele.

Die meisten Heimcomputer-Spiele lassen sich in zwei Kategorien unterteilen: Abenteuerspiele mit und ohne grafischen Darstellungen wie z. B. „The Hobbit" und Actionspiele wie „Space Invaders" oder „Asteroids". Schon eine flüchtige Analyse der beiden Spieltypen enthüllt die Ursachen für deren Ähnlichkeit. Bei Programmen im Stil vom Space Invaders, um auf die Actionspiele zuerst einzugehen, sind zwei Grundbestandteile erforderlich: Feuerstellung und Ziel. Gelingt es also, Varianten dieser beiden Voraussetzungen zu abstrahieren, und kann der Programmierer die Position des Verteidigers, die Feuerkraft und Frequenz wie Intensität der Angriffswellen frei gestalten, so lassen sich eine Reihe verschiedener Spiele entwickeln, die sich nur geringfügig voneinander unterscheiden, weil lediglich die Parameter zu modifizieren sind. Eine Analyse der zahlreichen Spiele mit ähnlich klingenden Namen, innerhalb einer „Programm-Generation" produziert, belegt, daß diese Philosophie von professionellen Programmierern praktiziert wird.

Ein hervorragendes Beispiel für Software, die dies dem User ebenfalls ermöglicht, ist „Games Designer" (Software Studios/Quicksilva für den 48-K-Spectrum) von John Hollis. Games Designer beinhaltet acht Spieltypen. Hat man einen davon gewählt, lassen sich die genannten Grundparameter ändern. Die Entwicklung eines davon unabhängigen neuen Spieles ist allerdings nicht möglich.

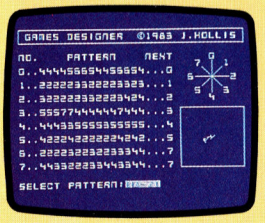
Die Software wird menügeführt. Unter dem Gesichtspunkt, eine Reihe vorprogrammierter Spiele zu bekommen, ist der Games Designer

Spiel-Designer



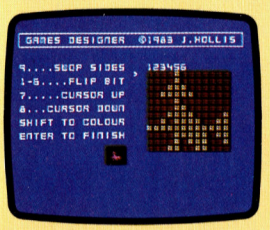
Konfigurations-Menü
Dieses bietet die Option, die Bewegungsrichtung eines Raumschiffes oder einer Laserbasis zu wählen, ferner Wahlmöglichkeiten für Vorder- und Hintergrundfarben.

Bewegungs-Menü
Damit wird die Angriffsart festgelegt. Am oberen rechten Bildschirmrand ist eine bezifferte Richtungskarte zu sehen, unten rechts befindet sich zudem ein Muster-Display.

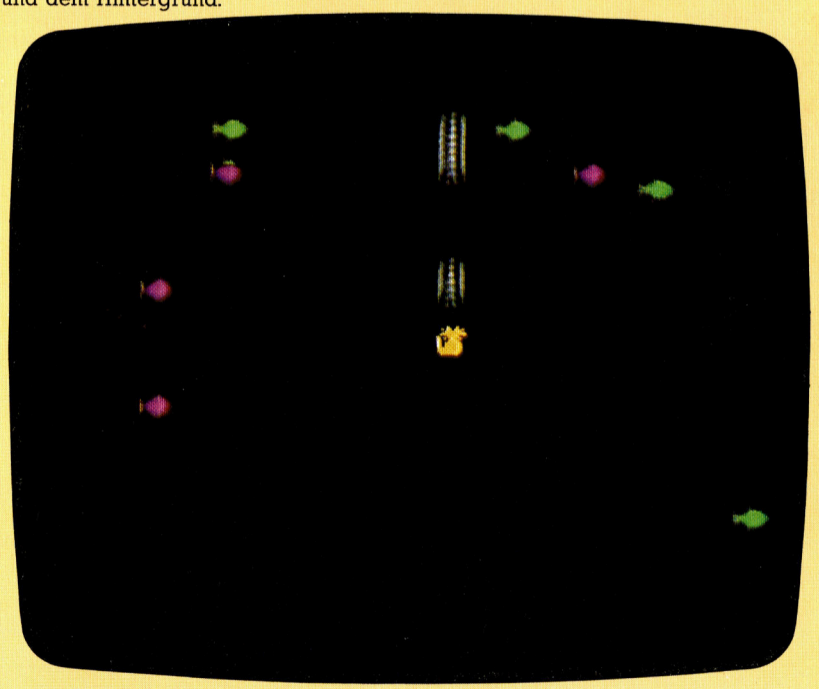


Basisformat
Die Sprites stehen vor einem einfachen Hintergrund und gleiten langsam nach unten.

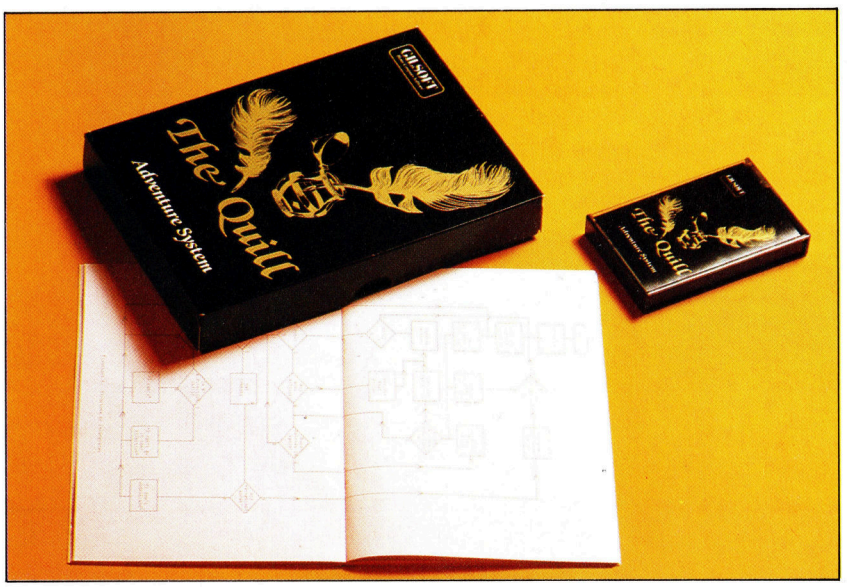
Endergebnis
Die Abbildung zeigt die Unterschiede zu den Original-Sprites und dem Hintergrund.



Sprite-Konfiguration
Die zu füllende oder zu löschende Koordinate auf der Tabelle wird ausgewählt, wobei am unteren Bildschirmrand die tatsächliche Größe und Farbe des Sprite zu sehen ist.



sein Geld wert. Nach Wahl des zu modifizierenden Programms erscheint das Menü, mit dem Art und Umfang der Modifikation festzulegen sind. Das beginnt bei Form und Farbe der Sprites, aus denen die Protagonisten geschaffen werden. Wenngleich ein Menü bereits existierender Sprites gezeigt wird, bietet das doch Vorteile, da man mit fertigen Bausteinen arbeiten kann, die sich innerhalb einer 12 × 12-Matrix bitweise ändern lassen. Alternativ gibt es die Option, mit einem leeren Screen zu beginnen und völlig neue Gestalten zu entwickeln. Das Format 12 × 12 (zwei auf zwei Zeichen) läßt ein perfekt nutzbares, eigenständiges Sprite zu.



Mit „The Quill" kann der Spieler Abenteuerprogramme selbst gestalten. Im Prinzip ähnelt The Quill herkömmlichen Datenbankprogrammen. Orte und Objekte, die im Abenteuer enthalten sind, müssen definiert werden. Durch objektorientierte Programmiertechniken können die Elemente aufgerufen und dargestellt oder als Parameter benutzt werden, sobald sie im Spiel selbst Verwendung finden.

Die Sprite-Definition selbst erfolgt menügeführt: Am linken Bildschirmrand werden die zur Verfügung stehenden Befehle dargestellt, rechts erscheint das entsprechende Sprite vergrößert. Als hilfreich erweist sich eine zweite Darstellung des Sprites in originaler Größe und Farbe unter dem Text.

Daraufhin erfolgt die Definition, wie sich diese Sprites auf dem Spielfeld bewegen sollen. Dieser Teil, Konfiguration genannt, erlaubt mehr als nur einfache Richtungsanweisungen der Einzelelemente. Mit verschiedenen Optionen des Hauptmenüs kann nun ein Bewegungsmuster der Elemente ausgewählt werden. Um die Bewegung komplexer zu gestalten, lassen sich in diesem Stadium zwei und mehrere Muster miteinander verbinden.

Angriffswellen und Effekte

Anschließend erfolgt die Auswahl der Angriffswellen-Frequenz sowie das Einfügen von Spezialeffekten, beides sowohl visuell als auch hörbar. Damit ist die Spielgestaltung beendet. Bleibt noch die Abspeicherung auf Cassette, um es später wieder spielen zu können. Hier aber liegt unter Anwendergesichtspunkten der Nachteil des Programms: Das Spiel wird nicht in spielbarer Form gespeichert. Lediglich die Eingabeparameter für den Games Designer sind festgehalten. Womit der Aufwand für all jene nutzlos ist, die auf diese Weise kommerziell verwendbare Spiele entwickeln wollen.

Genau umgekehrt ist der Sachverhalt bei „The Quill", einem Programmpaket zur Entwicklung und Vermarktung von Abenteuerspielen. In dem ausführlichen Begleithandbuch werden unter der Überschrift „Selling your Adventures" nützliche Hinweise gegeben, wie man Programme fehlerfrei gestaltet.

Obwohl als Generator für Spiele im Stile von „Dungeons & Dragons" angeboten, werden bei The Quill Techniken verwendet, die denen kommerzieller Dateiverwaltungen ähnlicher sind. Das Programm ist in drei Hauptteile gegliedert: die eigentliche Datenbank, einen Datenbank-Editor, mit dem die Parameter festgelegt werden, und einen Datenbank-Interpreter, mit dem das Spiel zum Laufen gebracht wird.

Begriffslabyrinth

Könnte man Spiele wie Space Invaders als Games analysieren, bei denen Protagonisten in Position gebracht werden, so träfe auf die Abenteuerspiele die Bezeichnung „Begriffslabyrinth" zu, da es sich hierbei um ein Labyrinth handelt, das nicht nur in räumlichen Dimensionen existiert. Folglich muß der Spieler nicht nur nach dem Ausgang suchen, sondern ist aufgefordert, Gegenstände zu finden, die aber nur unter ganz bestimmten Voraussetzungen nützlich sind. Es gibt z. B. keinen Grund, eine Glühlampe mit sich herumzutragen, wenn im Spiel kein Raum mit einer leeren Glühlampenfassung existiert.

Um das Spiel interessanter zu gestalten, ist die Anzahl der Gegenstände, die der Spieler mit sich führen kann, normalerweise begrenzt. Diese bewußte Limitierung nutzt der Programmierer, um den Spieler in ständiger Aktion zu halten und die Objekte beschaffen zu lassen, die er zur Lösung der Aufgabe im fortgeschrittenen Spielverlauf braucht.

Erster Schritt bei der Entwicklung eines Adventure-Games ist die Entwicklung eines Drehbuchs und einer Umgebung, in der die Handlung spielt. Im Quill-Handbuch wird zunächst ein einfaches Spiel dargestellt, bei dem der Spieler zwischen zehn Gegenständen wählen kann und sich in nur sechs Räumen bewegt. Dieses Spiel ist nicht Programmbestandteil. Um es spielen zu können, müssen die im Handbuch dargelegten Parameter eingegeben werden, wie es bei der Gestaltung eines völlig eigenen Programms auch erforderlich wäre.

Die Anzahl der im Spiel zu verwendenden Parameter ist auch bei The Quill limitiert, doch dürfte es einen Mangel an Optionen kaum geben, da man zwischen 252 Orten und 210 Gegenständen wählen kann.

Wassersport

„Ballerspiele“ im Stil von „Space Invaders“ dominierten lange den Software-Markt. Um so erfreulicher ist es, Programme zu entdecken, die neue Wege aufzeigen.

Bei dem Spiel „Scuba Dive“ von Durell Software handelt es sich um ein Unterwasserabenteuer, das schon nach kurzer Zeit in den Hitparaden ganz oben stand. Drei Scuba Dive-Versionen sind erhältlich: Die Spectrum-Version schrieb Mike Richardson, von Ron Jeffs stammt die für den Oric-1 und Nigel Dewdney stellte das Programm für den C 64 fertig. Die Oric-1-Version wurde auch für den Oric Atmos lauffähig gemacht.

Der Spieler übernimmt die Rolle eines Tauchers, der auf dem Meeresgrund Schätze sammelt und dabei einiges riskiert. Hauptaufgabe des Bildschirm-Helden ist es, Perlen einzusammeln, die mit entsprechenden Gewinnpunkten honoriert werden. Die Perlen befinden sich in Austern und Riesenmuscheln. Im fortgeschrittenen Spielstadium müssen wertvolle Gegenstände aus Schatzkisten geborgen werden, die in einem Tiefsee-Labyrinth versteckt sind.

Tiefsee-Abenteuer

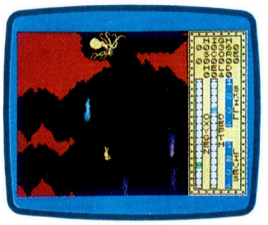
Es gibt eine Reihe guter Gründe, sehr behutsam vorzugehen. Das Wasser ist von Kreaturen bevölkert, die dem Taucher zum Verhängnis werden können: Quallen, Kraken, Tintenfische, Zitteraale und – in der Spectrum-Version – Haie! Bei Berührung eines dieser Tiefsee-Monster verliert der Taucher sein Leben. Er muß den Kontakt mit ihnen um jeden Preis vermeiden. Das Einsammeln der Perlen aus den Riesenmuscheln ist ebenso gefährlich, da sie ihre Schalen unvermittelt schließen und die Spielfigur einsperren.

Die schmalen Eingänge zur Haupthöhle und den tieferen Ebenen werden von Riesenkraken bewacht. Da sie ihre Fangarme ständig bewegen, ist ein Vorbeikommen schwer. Doch mit entsprechender Übung schafft man auch das. In der Commodore-Version gibt es keine Kraken. Dafür stehen Falltüren im Weg, die sich rhythmisch öffnen und schließen. Sie betäuben den Taucher, wenn er getroffen wird.

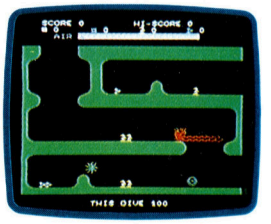
Der Spieler hat in diesem Programm drei Leben zu verlieren. Vier Schwierigkeitsgrade stehen zur Verfügung. Er verliert ein Spiel durch Berührung der Kreaturen oder wenn der Sauerstoff ausgeht. Jede Spielvariante beginnt mit einem Blick auf die Meeresoberfläche und einer Teilansicht der unteren Regionen. Dabei dient ein Boot als Tauchbasis. Beim Abtauchen besteht die Gefahr, unter dem Boot gefangen zu werden. In der Spectrum-Version ist ein guter Orientierungssinn nötig, weil das Boot driftet, wenn man sich unter Wasser befindet. Im Programm für den Oric stellt dies kein Problem dar, da sich das Boot stets von links nach rechts bewegt und im Blickfeld ist, so lange man sich an der Oberfläche aufhält.

Die grafische Qualität des Spectrum-Programms (übrigens das beste der drei) ist hervorragend. Die Farbigkeit besticht und die Gestaltung der Meeresungeheuer wie der Höhlen wirkt äußerst realistisch. Der Taucher wird durch die „X“- und „Z“-Taste im Uhrzeigersinn oder entgegengesetzt gesteuert, mit den Tasten „Space“ und „Shift“ bewegt man ihn vorwärts. Beim Spectrum kann ebenfalls ein Joystick verwendet werden; allerdings arbeitet das Programm nicht mit dem Kempston Joystick-Interface. Commodore-Spieler haben auch die Wahl zwischen Tasten- und Joysticksteuerung, wogegen das Oric-Programm nur mit dem Keyboard zu spielen ist.

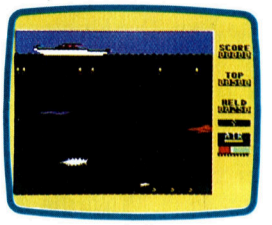
Die Bildschirmfotos zeigen die Qualitätsunterschiede der einzelnen Scuba Dive-Versionen.



Sinclair Spectrum

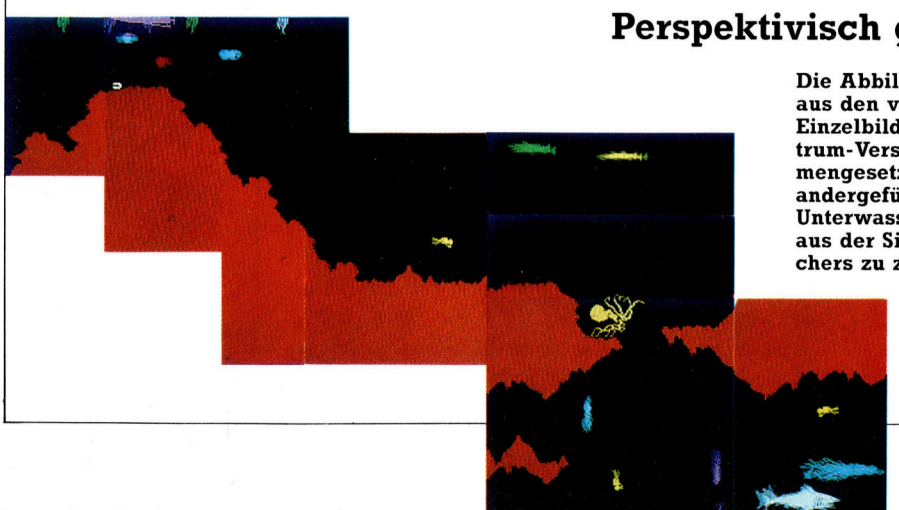


Oric-1



Commodore 64

Perspektivisch gesehen



Die Abbildung wurde aus den verschiedenen Einzelbildern der Spectrum-Version zusammengesetzt und aneinandergefügt, um das Unterwasser-Labyrinth aus der Sicht des Tauchers zu zeigen.

Micro-Überwachung

Heimcomputer können mit Sensoren für Licht, Wärme oder Geräusche verbunden werden. Mit den ermittelten Daten läßt sich die Heizung regeln oder eine Alarmzentrale aufbauen.

Microprozessoren finden heute in einer Vielzahl von Haushaltsgeräten wie Waschmaschinen, Toastern, Videorecordern oder auch Zentralheizungen Verwendung. Zum Informationsaustausch können die Chips miteinander verbunden oder auch an eine Kontrollzentrale angeschlossen werden. Gerade im Haushalt wäre eine zentrale Steuerung verschiedener Arbeitsvorgänge wünschenswert. Bei Steuerungssystemen wird zwischen drei unterschiedlichen Kategorien unterschieden: Online-Systeme, Interrupt (Unterbrecher-) Systeme und Netzwerk-Systeme.

Online-Systeme kann man fertig kaufen, der Selbstbau ist aber nicht unmöglich. Bei Online-Anlagen ist ein normaler Heimcomputer mit speziellen Interfaces an elektromechanische Geräte angeschlossen, die z. B. Thermostate steuern. Die Montage erfordert allerdings gute Kenntnisse der Hardware. Umfangreiches Wissen vorausgesetzt, können Sie auch das spezifische Kontroll-Programm selbst entwerfen. Die Einschränkung der Online-Systeme besteht darin, daß das Programm ständig laufen muß. Wird die Stromversorgung auch nur kurz unterbrochen, reagieren die angeschlossenen Regler nicht mehr auf die Anweisungen des Rechners.

Interrupt-Systeme

Die zweite Gruppe von Kontrolleinrichtungen arbeitet mit Interrupts. An der Zentralheizung, der Alarmanlage oder einem Rauchmelder sind Geräte angeschlossen, die im Fall einer Störung ein Signal an den Rechner geben, wodurch das gerade laufende Programm sofort unterbrochen wird. Auch ein Interrupt-System muß ständig laufen. Kurzfristiger Stromausfall schadet jedoch nicht, weil sich die überwachten Anlagen auch ohne den Computer selbsttätig regeln können.

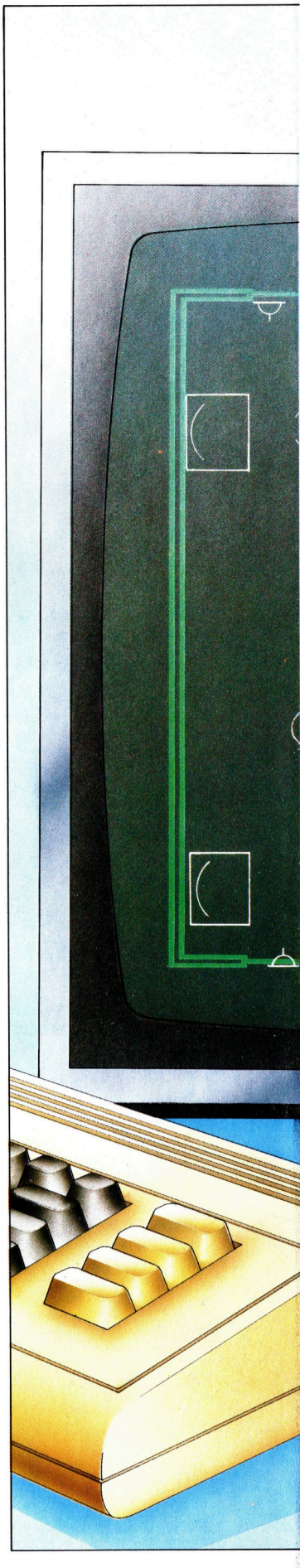
Der Rechner hält für jedes angeschlossene Gerät ein besonderes Programm bereit. Angenommen, vom Rauchmelder kommt eine Interrupt-Botschaft, während Sie gerade ein Abenteuerspiel am Computer ausprobieren. Daraufhin bricht der Computer das Spiel ab (wobei er den Spielstand automatisch speichert) und setzt das Programm „Rauchmeldung" in Gang. Auf dem Bildschirm könnte zum Beispiel ein Hinweis auf Feuergefahr erscheinen oder, würde gerade niemand mit dem Computer arbeiten, eine Sirene ertönen. Nachdem die Ursache für den Alarm entdeckt und die eventuellen Schäden beseitigt worden sind, können sie sofort wieder an der Stelle Ihres Abenteuerspiels weitermachen, an der Sie vorher unterbrochen wurden. Wäre das Signal aber vom Zeitschalter der Zentralheizung gekommen, hätte der Rechner lediglich die Zeit sowie die Innen- und Außentemperaturfühler abgefragt und daraufhin den Heizkessel höher eingestellt – all dies geschieht so schnell, daß Sie wahrscheinlich nicht einmal eine Unterbrechung Ihres Spiels wahrgenommen hätten.

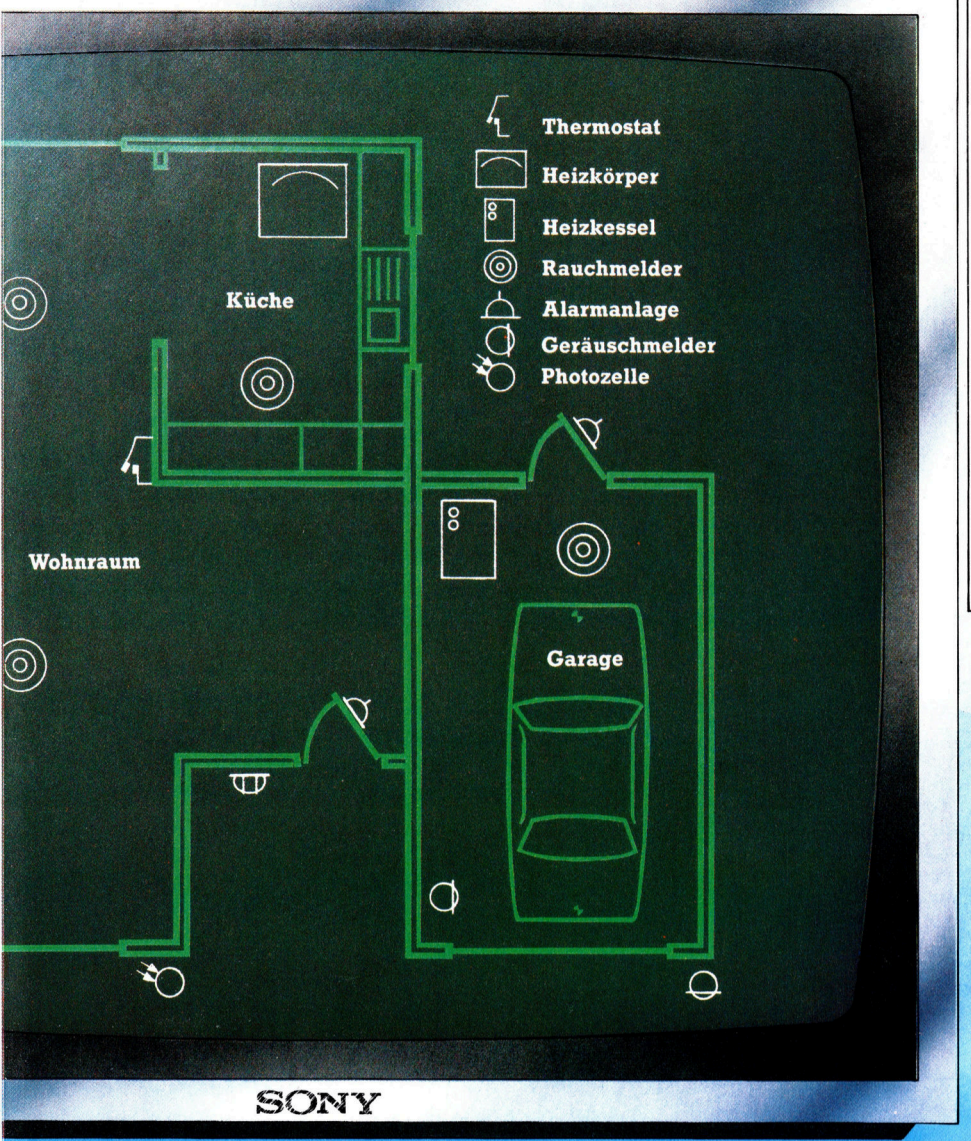
Ein Beispiel für die Kategorie der Netzwerk-Systeme ist der BSR-Home Controller. Diese Anlage steuert über einen Stromkreis der Hausinstallation alle daran angeschlossenen Geräte. Jeder Regler hat eine Codenummer und kann einzeln mit einem über das Leitungsnetz gesendeten Hochfrequenzsignal ein- oder ausgeschaltet werden. Die Verbindung eines Computers mit dem Stromnetz ist aber nicht ungefährlich und sollte ausschließlich von Fachleuten vorgenommen werden!

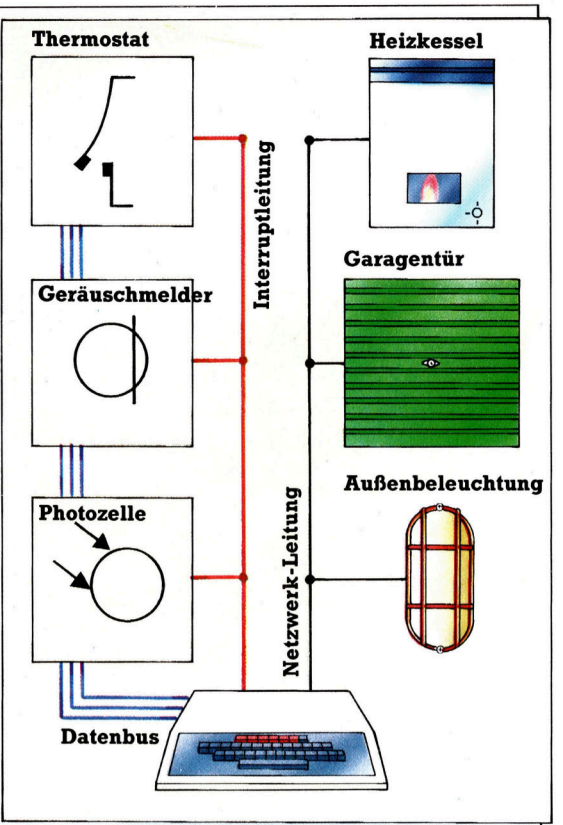
Der zweite Schritt nach der Installation eines Kontroll-Systems ist dessen Fernbedienung – etwa das Ausschalten der Heizung vom Büro aus. Einige dieser Anlagen können mit einem Telefon-Modem versehen werden – allerdings sollte das System, in diesem Fall durch ein „Paßwort", vor unberechtigtem Zugang geschützt werden.

Ebenso wie die beschriebenen Systeme kann man eine Vielzahl von Sensoren dafür kaufen – vom einfachen Microschalter der Alarmanlage bis zum digitalen Temperaturfühler. Eine große Zahl von Heimcomputern erlaubt den problemlosen Ausbau, etwa der Apple II, Acorn B oder der Commodore 64, bei anderen Rechnern müssen Sie jedoch mit umfangreichen Umbauten rechnen, um ähnliche Ergebnisse zu erhalten.

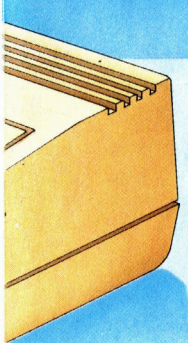
Die Kosten des „rechnergesteuerten Hauses" sind nicht ganz niedrig – die Verbindung Rechner–Stromnetz muß schließlich nicht nur elektrisch sicher, sondern auch ansonsten verläßlich sein.







Die Zeichnung zeigt zwei verschiedene Techniken, mit denen der Rechner die einzelnen Geräte anspricht: Die drei Sensoren links melden sich über die normale Interrupt-Leitung, die direkt zum Microprozessor führt. Das laufende Programm wird gestoppt und der Datenbus für die Meldung des Sensors freigemacht. Die Geräte auf der rechten Seite sind so gekoppelt, daß der Computer jedes einzelne getrennt aktivieren kann. Dazu sendet er Daten, die zum Beispiel die Gerätenummer der Garagentür und den Befehl „Öffnen" beinhalten.



Hauszentrale

Ein Schema des Hauses auf dem Bildschirm darzustellen, ist nicht mehr unmöglich – der moderne Werkschutz mit seinen computergestützten Sicherheitsanlagen kennt diese Technik schon lange. Wenn der Computer nach der Interrupt-Methode arbeitet, kann der Schirm natürlich auch frei bleiben – die Software verrichtet ihre Arbeit diskret im Hintergrund, ohne Sie beim Umgang mit dem Rechner zu stören.

„Verhexte" Befehle

In der Fortsetzung unserer Serie über den Maschinencode untersuchen wir die vielfältigen Formen, in denen ein Programm eingegeben werden kann – von binär bis Assemblersprache.

Anfänger auf dem Gebiet des Maschinencodes haben oft Probleme mit den vielen unterschiedlichen Formaten, in denen die Programme geschrieben werden können. Obwohl im Speicher des Computers alle Daten über Binärzahlen (mit der Länge von je acht Bit) abgelegt werden, sind diese Zahlenreihen in gedruckter Form sehr unübersichtlich, schwer zu behalten und anfällig für Tippfehler. Für ihre Ein- und Ausgabe wird daher die hexadezimale Form verwendet. Sie hat den Vorteil, daß der Inhalt eines Bytes als zweistellige Zahl ausgedrückt werden kann und daß sich jede Adresse des Arbeitsspeichers (0–65535) vierstellig darstellen läßt.

Damit sie sich von einer Dezimalzahl unterscheidet, wird eine Hexadezimalzahl normalerweise mit einem vorangehenden „$"-Zeichen geschrieben. Bei einigen „Opcodes" besteht außerdem die Notwendigkeit, einen aus zwei Bytes bestehenden Operanden (z. B. LDA $3F80) in umgekehrter Folge einzugeben. D. h., es wird zunächst das niederwertige Byte eingegeben und dann das höherwertige. In unserem Beispiel ist die Eingabereihenfolge AD (die hexadezimale Darstellung des Opcodes für LDA auf dem 6502), gefolgt von 80 und 3F.

Normalerweise wird ein Maschinenprogramm als „Hex Dump" gedruckt, in Form einer langen Liste mit zweistelligen Hexadezimalwerten. Dabei muß die Speicherstelle (in dezimal oder in hex) angegeben werden, die mit dem ersten Hexadezimalwert belegt werden soll. Der zweite Hexadezimalwert wird dann in die nächsthöhere Speicherstelle geladen usw. Der Ladevorgang läßt sich mit dem BASIC-Befehl POKE ausführen. Die Anfangsadresse wird dabei auf $1000 (dezimal 4096) gesetzt, und die hexadezimale Liste sieht folgendermaßen aus:

```
AD (dezimal 173)
80 (dezimal 128)
3F (dezimal 63)
```

Die folgenden drei BASIC-Befehle laden das Programm:

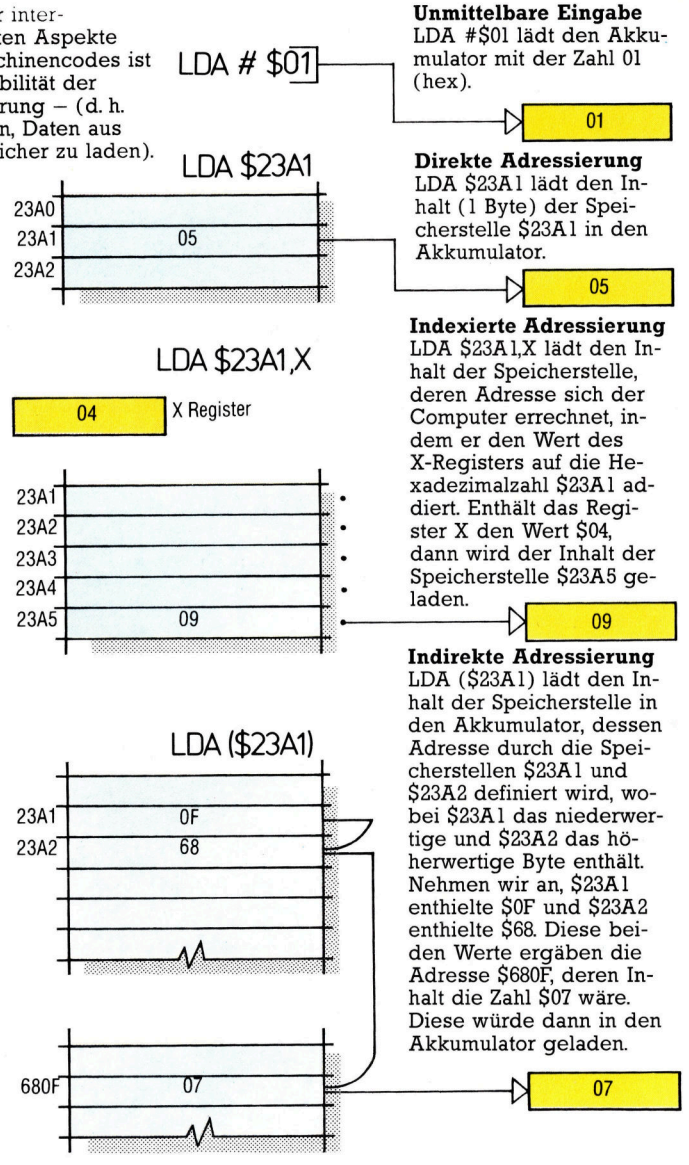
```
POKE 4096, 173
POKE 4097, 128
POKE 4098, 63
```

Hierbei müssen alle hexadezimalen Werte in Dezimalzahlen umgewandelt werden, bevor sie in dem POKE-Befehl eingesetzt werden können.

Längere Maschinenprogramme lassen sich über ein kurzes BASIC-Programm – ein „Lademodul für Maschinencode" – eingeben. Dabei wird zunächst die Anfangsadresse des Programms angegeben und dann jeder einzelne Hexadezimalwert. Nach den Eingaben wandelt das Lademodul den Hexwert in eine Dezimalzahl um und setzt diese mit POKE an die näch-

Adressierungsarten

Einer der interessantesten Aspekte des Maschinencodes ist die Flexibilität der Adressierung – (d. h. Methoden, Daten aus dem Speicher zu laden).



Unmittelbare Eingabe
LDA #$01 lädt den Akkumulator mit der Zahl 01 (hex).

Direkte Adressierung
LDA $23A1 lädt den Inhalt (1 Byte) der Speicherstelle $23A1 in den Akkumulator.

Indexierte Adressierung
LDA $23A1,X lädt den Inhalt der Speicherstelle, deren Adresse sich der Computer errechnet, indem er den Wert des X-Registers auf die Hexadezimalzahl $23A1 addiert. Enthält das Register X den Wert $04, dann wird der Inhalt der Speicherstelle $23A5 geladen.

Indirekte Adressierung
LDA ($23A1) lädt den Inhalt der Speicherstelle in den Akkumulator, dessen Adresse durch die Speicherstellen $23A1 und $23A2 definiert wird, wobei $23A1 das niederwertige und $23A2 das höherwertige Byte enthält. Nehmen wir an, $23A1 enthielte $0F und $23A2 enthielte $68. Diese beiden Werte ergäben die Adresse $680F, deren Inhalt die Zahl $07 wäre. Diese würde dann in den Akkumulator geladen.

Bits und Mnemonics

Programme im Maschinencode können auf mehrere Arten geschrieben werden. Normalerweise wird dafür die Assemblersprache eingesetzt, in der die Opcodes (Maschinenbefehle) als Mnemonics eingegeben und symbolische Bezeichnungen (Labels) verwendet werden können. Beispiel:

```
LDA GEWICHT
ADC BENZIN
STA GEWICHT
```

Für symbolische Bezeichnungen müssen allerdings die Adressen angegeben werden wie z. B.:

```
BENZIN = $03EE
GEWICHT = $031F
```

Ein Assembler-Programmpaket übersetzt diese Angaben in einen Hex Dump und speichert sie auf einer Diskette. Der „Pseudoassembler" ist nicht so einfach zu lesen, kann aber von einem „Spot Assembler" ohne ein Diskettenlaufwerk unmittelbar umgesetzt werden.

```
LDA $031F
ADC $03EE
STA $031F
```

Ein Hex Dump (Listing im Maschinencode) besteht aus einer Anfangsadresse (links) und einer Folge von zweistelligen Hexadezimalwerten, die dem Inhalt des Arbeitsspeichers entsprechen. Die Operanden (z. B. $031F) werden dabei in umgekehrter Reihenfolge gespeichert (1F03), und die Opcodes werden durch die entsprechenden Hexadezimalwerte ersetzt.

```
19C4 AD 1F 03 6D EE 03 8D 1F 03
```

ste erreichbare Speicherstelle.

Die meisten Computer verfügen über Befehle, mit denen das Gerät den BASIC-Modus verlassen und von einer bestimmten Speicherstelle an mit der Ausführung des Maschinencodes beginnen kann. So bedeutet der Befehl SYS 4096 (RETURN): „Übergib die Steuerung dem System, und beginne mit der Ausführung der Instruktion in Speicherstelle 4096". Eine weitere Möglichkeit ist CALL $E651. Hier wird eine Routine im Maschinencode aufgerufen, die an der Speicherstelle (hex) E651 anfängt.

BASIC-Beschleunigung

Bei Eingabe dieser Befehle führt das Gerät die Maschinenroutine aus. Ist diese fehlerfrei und verfügt sie an ihrem Ende über einen korrekten Abschluß, wird die Steuerung danach wieder an das BASIC zurückgegeben. Mit Maschinenroutinen lassen sich einzelne Funktionen eines BASIC-Programms erheblich beschleunigen, wobei die gleichen Routinen durchaus von mehreren Stellen aus aufgerufen werden können.

Leider werden Fehler im Maschinencode nicht mit einem hilfreichen „SYNTAX ERROR" angezeigt, sondern lassen das System „abstürzen": Das Gerät unterbricht die Arbeit und nimmt keine Eingaben mehr entgegen. Dieser Vorgang schadet zwar dem Computer nicht, Ihr Programm ist allerdings gelöscht und muß neu eingegeben werden. Mit dem Maschinencode läßt sich daher nicht so bequem experimentieren wie mit einem BASIC-Programm.

Es gibt jedoch ein Hilfsprogramm, das große Unterstützung bietet: der „Monitor für Maschinencode" (nicht zu verwechseln mit dem Bildschirmmonitor). Einige Hersteller haben den Monitor als ROM bereits im Gerät integriert, normalerweise ist er jedoch auf einer Cassette oder als Cartridge erhältlich. Ein Monitor für Maschinencode ist ein einfaches Betriebssystem, das den Inhalt des Arbeitsspeichers in Blöcken auf dem Bildschirm anzeigt. Mit dem Monitor lassen sich Speicherstellen leicht verändern oder überschreiben. Er ist eine der schnellsten Methoden, Hexadezimalcode einzugeben. Mit diesem Programm läßt sich außerdem Maschinencode ohne ein BASIC-Lademodul auf Cassette speichern oder davon einlesen.

Hex Dumps sind zwar eine bequeme Methode zur Darstellung des Maschinencodes, lassen sich aber nur schwer lesen. Es ist so gut wie unmöglich, Opcode und Operanden voneinander zu unterscheiden, wenn man nicht die hexadezimalen Äquivalente aller Opcodes im Gedächtnis hat. Maschinenprogramme werden daher mit Kürzeln geschrieben, die aus drei Buchstaben bestehen – sogenannten „Mnemonics" – und anhand der Befehlstabelle des Prozessors in hexadezimale Form übersetzt.

Die Assemblersprache ist eine Programmiermethode für Maschinencode, bei der nicht nur die Mnemonics des Opcodes verwendet werden, sondern anstelle von Hexadezimalzahlen auch Namen (oder symbolische Bezeichnungen) als Operanden eingesetzt werden können. Legt ein Spiel z. B. in der Speicherstelle $07B2 die Anzahl der verschossenen Raketen ab, dann kann dieser Wert mit folgendem Befehl in den Akkumulator geladen werden:

```
LDA RAKETEN
```

Am Anfang des Programmes muß natürlich angegeben werden, daß die Speicherstelle für RAKETEN = $07B2 ist und daß dieser Speicher beim Start auf den Wert $09 (neun Raketen) gesetzt wird.

Ist das Programm in der Assemblersprache (dem sogenannten Source Code) abgeschlossen, wird es von einem Hilfsprogramm, dem „Assembler", bearbeitet. Der Assembler ersetzt die Mnemonics und symbolischen Bezeichnungen jedes Befehls durch ihre hexadezimale Entsprechung und stellt eine neue Version des Programms her, den „Object Code". Dieser Code kann in den Arbeitsspeicher eines Computers geladen und anschließend gestartet werden.

Opcode

Einige Opcodes, die für einen Microprozessor typisch sind:

Jump SubRoutine (Springe zur Unterroutine)
Diese Funktion entspricht dem BASIC-Befehl GOSUB. JSR $354D verändert den Wert des Programmzählers (PC-Register), so daß der Code von der Speicherstelle $354D an ausgeführt wird.

ReTurn from Subroutine (von Subroutine zurückspringen)
Trifft das Programm auf den Befehl RTS, springt der Prozessor auf die Speicherstelle zurück, von der aus die Unterroutine aufgerufen wurde (entspricht hier dem Befehl RETURN in BASIC).

Branch if MInus (verzweige bei Minuswert)
Dieser Befehl ist eine von mehreren bedingten Verzweigungsmöglichkeiten im Maschinencode. (Der BASIC-Befehl IF...THEN GOTO ist ebenfalls ein bedingter Sprung.) Befindet sich ein negativer Wert im Akkumulator, dann verzweigt die Programmausführung zu der angegebenen Adresse. BPL veranlaßt den Sprung bei einem positiven Wert.

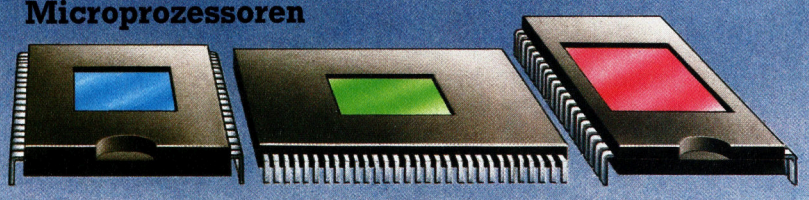
LoaD X register (Lade das X-Register)
Das X-Register ist ein weiterer interner Speicher. Es kann zwar keine Rechenvorgänge ausführen, wird aber für die „indexierte Adressierung" eingesetzt. LDX lädt einen Wert in das Register X, während STX (STore X) diesen Wert wieder in den Speicher zurückschreibt.

INcrement X (Erhöhe X um 1)
Addiert man eine 1 auf den Wert des Registers X (DEX – DEcrement X – subtrahiert 1 vom Wert des Registers X) und verwendet die indexierte Adressierung, kann man Schritt für Schritt durch eine Anzahl Speicherstellen hindurchgehen.

Motorola 68000

Seit Motorola im Jahre 1982 mit den Microprozessoren der 68000er Serie den Nachfolger der vielgerühmten 8-Bit-CPU 6809 herausbrachte, dominiert dieser Prozessor auf einem wesentlichen Teil des Marktes für 16- und 32-Bit-Prozessoren.

Microprozessoren



6502
Wie der Z80 von Zilog ist auch der von MOS Technology entwickelte 6502 einer der Hauptprozessoren der Microcomputerindustrie. Er besitzt einen 16-Bit-Adreßbus und einen 8-Bit-Datenbus. Größte Eigenheit des Prozessors ist seine Registeranordnung. Die gesamte Zeropage (der erste Block des RAM-Bereiches) kann als Allzweckregister verwandt werden.

68000
Motorola verbesserte zwar den 6800 zum 6809 Chip, konnte sich aber nicht mehr rechtzeitig einen größeren Anteil des 8-Bit-Marktes sichern. Im nachhinein gesehen war das ein Vorteil, da die Firma sich dadurch auf die Entwicklung des 68000, einen 16/32-Bit-Prozessor, konzentrieren konnte. Der 68000 läßt sich in Verbindung mit vielen Zusatzchips betreiben.

Z80
Theoretisch verfügt der Z80 von Zilog über größere Kapazitäten als der MOS 6502. Er verwendet ähnliche Adreß- und Datenbusstrukturen, hat jedoch wesentlich mehr Register (12 8-Bit-Allzweckregister, zwei 16-Bit-Indexregister) und einen weitaus größeren Befehlssatz. Sein vielleicht größter Vorteil gegenüber dem 6502 ist die Unterstützung des Betriebssystems CP/M.

Der 68000 hat starke Ähnlichkeit mit dem MC6800 Prozessor von Motorola, der als intelligenter Steuerchip für Peripheriegeräte weit verbreitet ist. Der weitaus höher entwickelte 68000 läßt sich aus diesem Grund leicht mit Peripheriegeräten verbinden, wobei eine breite Palette einsatzfähiger Hardware bereits zur Verfügung steht. Ein- und Ausgabeplatinen mit dem 6821 PIA Chip (Peripheral Interface Adaptor) lassen sich dabei ebenso leicht anschließen wie Bildschirmgeräte mit 6845 CRTC Steuerchips (Cathode Ray Tube Controller), Taktgeber mit dem programmierbaren Zeitmodul des 6840 oder Diskettencontroller.

Der 68000 ist für die Entwicklung von Computern besonders wegen der großen Anzahl seiner Daten- und Adreßleitungen interessant. Beide Busnetze haben voneinander unabhängige Leitungen, wobei jedes einzelne Bit über einen eigenen Kontaktpin verfügt. Bei dem 8086, 8088 und Z8000 laufen die Kontakte im Multiplexverfahren zusammen, d. h., zwei Busse teilen sich einen Kontaktpin, wobei die ausgegebenen Signale von den angesprochenen Peripheriegeräten wieder getrennt und decodiert werden müssen.

Der 68000 kann daher so schnell arbeiten, wie seine Systemumgebung es erlaubt. Beim Einsatz entsprechend schneller RAM-Chips lassen sich Warteperioden auf ein Minimum reduzieren. Der schnellste Prozessor dieser Serie ist der 68000L12, der in vielen Funktionsbereichen bis 16 Megahertz getaktet werden kann.

Sinclair setzte für seinen QL den 68008 (den Nachfolger des 68000) ein. Intern arbeitet der Prozessor auf die gleiche Weise wie die anderen Prozessoren der Serie. Da er jedoch mit einem Acht-Bit- statt mit einem 16-Bit-Datenbus ausgerüstet ist, ist er mit bestehenden Acht-Bit-Systemen kompatibel. Dieser Chip wird im Format von 40 Kontaktpins geliefert.

Motorola wird bald einen Microprozessor mit noch größeren Verarbeitungsgeschwindigkeiten vorstellen. Es handelt sich um den 68020, einen 32-Bit-Prozessor mit 96 Kontaktpins, dessen Entwicklung jedoch noch nicht abgeschlossen ist. Ein speziell für mathematische Fließkommaarithmetik ausgelegter Prozessor ist ebenfalls in Vorbereitung.

Zusätzliche Funktionen

Weitere Chips der 68000-Serie bieten Ein- und Ausgabefunktionen, die denen früherer Modelle strukturell ähnlich sind, jedoch weitaus schneller arbeiten und über zusätzliche Funktionen verfügen.

Im Vergleich zu anderen 16-Bit-Prozessoren bietet der 68000 aufgrund seiner symmetrisch angelegten Adreß- und Datenregister und seines reichhaltigen Befehlssatzes eine ganze Reihe von Vorteilen. Vollkommen ist er jedoch nicht. Obwohl Adreß- und Datenregister die gleiche Breite (32 Bits) haben und in vielen Funktionen mit identischen Befehlen arbeiten, können sie nur separat eingesetzt werden. Daten müssen daher zur Bearbeitung oft von dem Adreßregister erst in ein Datenregister übertragen und nach der Bearbeitung wieder an das Adreßregister zurückgesandt werden.

Generell finden die schnellen und effektiven Prozessoren der 68000-Serie mehr und mehr Verbreitung. Sie sind in den Lisa und Macintosh von Apple ebenso eingebaut wie in den QL von Sinclair, und auch viele Mehrplatzcomputer, die auf dem Markt weniger bekannt sind, verwenden diesen Chip. Fähigkeiten, die vor einigen Jahren noch viel Geld gekostet hätten, sind nun zu vernünftigen Preisen verfügbar. Für die nächste Computergeneration scheint die 68000-Serie den gleichen Anklang zu finden, wie der Z80 und der 6502 sie heute besitzen.

Fachwörter von A bis Z

Architecture = Architektur
In der Umgangssprache beinhaltet der Begriff „Architektur" nicht nur die äußere Gestaltung eines Bauwerks, sondern auch seine funktionale Ordnung. Bei Rechnern wird dieser Ausdruck ausschließlich für die interne Struktur gebraucht. Unter der „Architektur" eines Microprozessor-Bausteins versteht man meist den logischen Aufbau der internen Register und ihre Arbeitsweise. Es besteht jedoch auch die Möglichkeit, daß die spezielle Halbleitertechnologie des Bausteins gemeint ist. Wird der Begriff dagegen für den ganzen Rechner angewandt, bezieht er sich auf die Art und Anordnung der Systembausteine und die Aufteilung der verfügbaren Speicherkapazität. Bei Rechnern mit der gleichen Architektur ist eine weitgehende Kompatibilität gegeben.

Array = Datenfeld
In einem „Array" werden Daten, die zueinander in Beziehung stehen, in systematischer Anordnung, etwa in Form einer Tabelle, mit Zeilen und Spalten abgelegt. Handelt es sich beispielsweise um eine zweidimensionale Tabelle, brauchen Sie zur eindeutigen Spezifizierung der Daten zwei Kennzahlen, nämlich den Zeilen- und den Spaltenindex. Ihr Rechner akzeptiert auch ein eindimensionales Datenfeld – das entspricht dann einer (einspaltigen) Liste. In BASIC werden zum Beispiel Dimension und Umfang eines Feldes durch eine DIMension-Anweisung spezifiziert.

Nichtmathematikern fällt das Verständnis für Felder mit mehr als drei Dimensionen schwer. Für den Rechner ist es aber nur eine Frage des Betriebssystems und des Speichers, ob er nun zwei oder 13 Dimensionen (d. h. 13 Indizes) verarbeiten kann.

Das Wort „Array" wird manchmal auch bei der Hardware benutzt, und zwar meist in der Verbindung „Arrayprozessor". Dieser enthält eine Serie konventioneller Rechenbausteine, so daß Rechenoperationen mit großen Datenmengen stark beschleunigt werden können.

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.

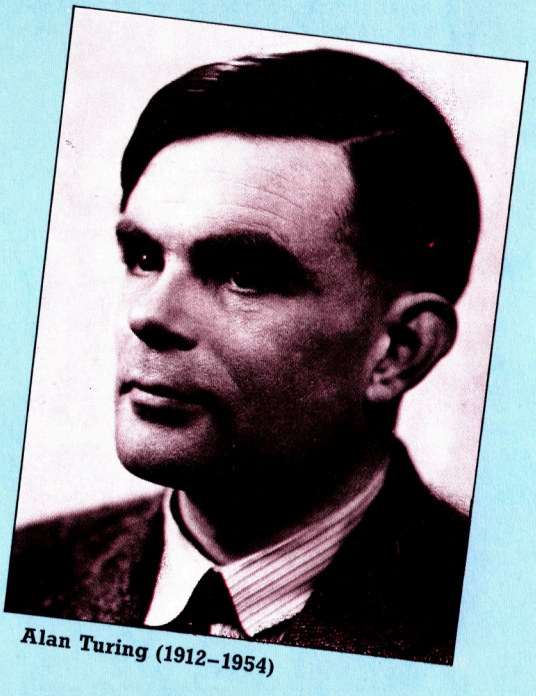
Artificial Intelligence = Künstliche Intelligenz
Viele Menschen sehen in der Vorstellung von einer „intelligenten Maschine" einen inneren Widerspruch, weil sie einem Apparat, der nur eingegebene Daten nach einem von Menschen entwickelten Schema (Programm) verarbeitet, keine Kreativität zutrauen.

Es gibt aber seit einiger Zeit Programme mit einer gewissen Lernfähigkeit – z. B. Schachprogramme, die nach zahllosen Spielen soviel „Erfahrung" gewinnen, daß sie sogar den Programmierer mattsetzen. Andere „intelligente" Programme führen selbständige Korrekturen aus, so daß das Listing nach einigen Durchläufen nicht wiederzuerkennen ist. Ob es sich nun tatsächlich um Intelligenz handelt, sei dahingestellt; immerhin werden an Hochschulen Forschungen unter dem Titel „Künstliche Intelligenz" durchgeführt.

Der englische Mathematiker Alan Turing hat in den fünfziger Jahren folgenden „Intelligenztest" für Computer vorgeschlagen: Sie setzen eine Person vor zwei Fernschreiber, von denen einer mit einem (versteckten) menschlichen Partner, der andere mit dem Test-Gerät in Verbindung steht. Wenn der Prüfer an seinen Fernschreibern durch keine noch so raffinierte Frage herausbekäme, wann der menschliche Partner und wann die Maschine antwortet, dann wäre dem Rechner Intelligenz nicht abzusprechen.

Für die meisten Forschungsergebnisse auf dem Gebiet der KI gibt es nur begrenzte Anwendungen. Dazu gehören Sprach- und Bildmuster-Erkennung und sogenannte Expertensysteme, bei denen versucht wird, Spezialkenntnisse zu simulieren.

Bei der KI gibt es zwei gegenläufige Vorgehensweisen: Beim „top-down"-Verfahren (von oben nach unten stufenweise Verfeinerung; „Baumstruktur") versucht man, Denkprozesse wie den Umgang mit Sprache auf normalen Rechnern nachzuahmen. Die „bottom-up"-Anhänger



Alan Turing (1912–1954)

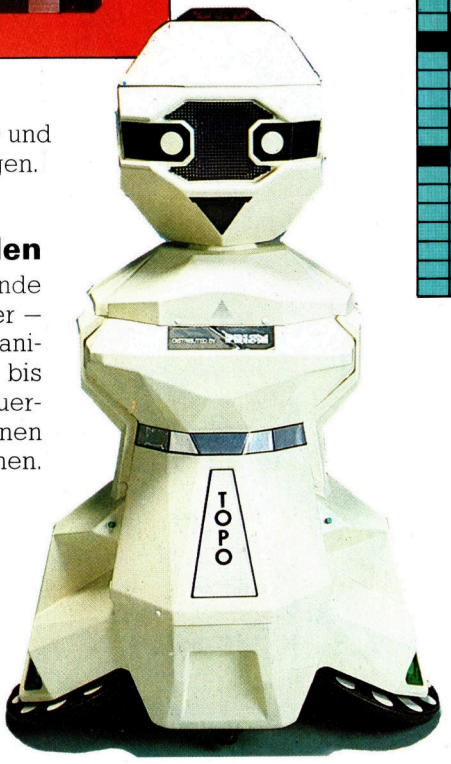
dagegen beginnen mit den Details. Sie versuchen, ein elektronisches Äquivalent zu den Gehirn-Nervenzellen (Neuronen) zu entwickeln. Im Verbund großer Netzwerke zeigen solche Elemente sogar schon Ansätze, aus Umgebungserfahrungen zu lernen.

Bildnachweise
449: Paul Chave
450: Tony Sleep
451: Steve Cross
452: Royal Aeronautical Society
453, 469, 470, 471: Ian McKinnell
454, 455: Chris Stevens
457, 462, 472, 473, 476: Kevin Jones
463: David Weeks

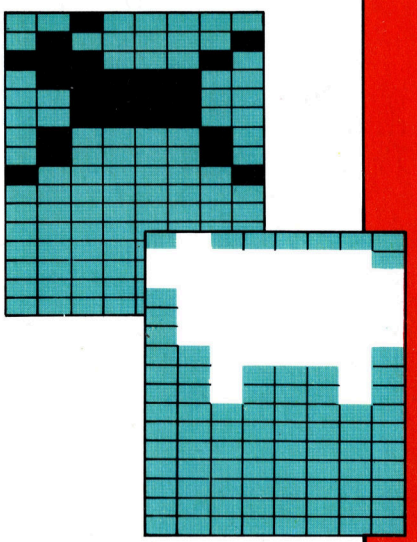
++ Vorschau +++ Vorschau +++ Vorschau +++

computer kurs Heft 18

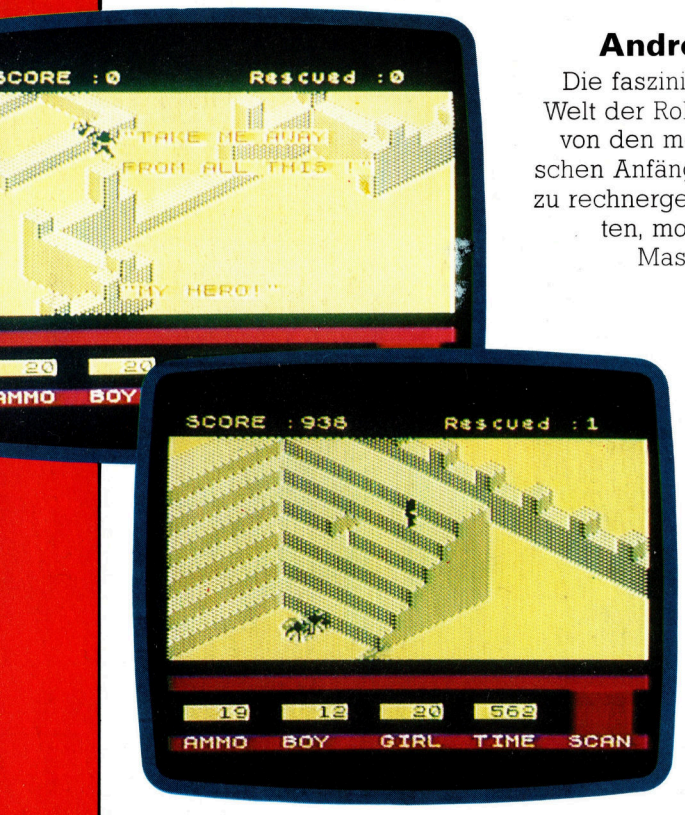
Atari XL
heißen die Nachfolger der bewährten Modelle 400 und 800. Die XL-Serie bietet interessante Verbesserungen.



Androiden
Die faszinierende Welt der Roboter – von den mechanischen Anfängen bis zu rechnergesteuerten, modernen Maschinen.



Anwendungen
Im LOGO-Kurs geht es diesmal um Sprite-Überschneidungen: So setzt man eigene Formen im Programm ein.



Ameisen-Angriff
Dieses Labyrinth-Spiel verfügt über eine herausragende Grafik-Qualität.

+++ Programme für die Schule +++ BASIC: Modul-Montage +++ Heimcomputer für die Zukunft +++ Maschinensprache +++ Tips für die Praxis +++ Strategie-Spiel +++