Einsteigen - Verstehen - Beherrschen

DM 3,80 öS 30 sfr 3,80

computer kurs

Heft 26

Der Tatung Einstein

Klänge mit dem Synthesizer

Integrierte Schaltungen

Assemblerbefehle

Geheimcode-Schlüssel

Ein wöchentliches Sammelwerk

computer kurs

Heft 26

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

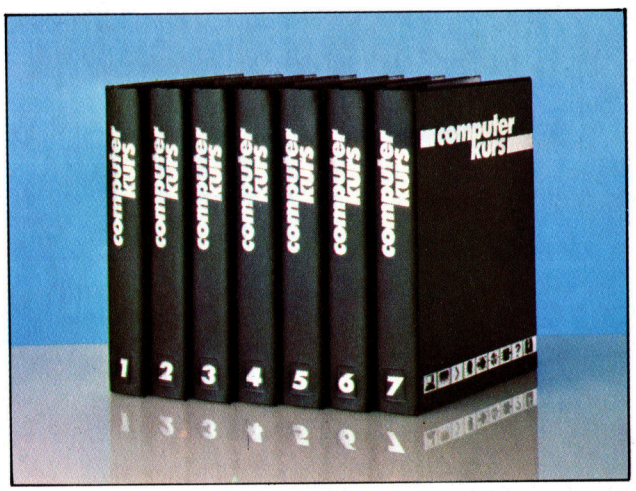
Schweiz: Der Sammelordner kostet sfr 15. Bitte wenden Sie sich an Ihren Kiosk; dort werden Sie jederzeit die gewünschten Exemplare erhalten.

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.

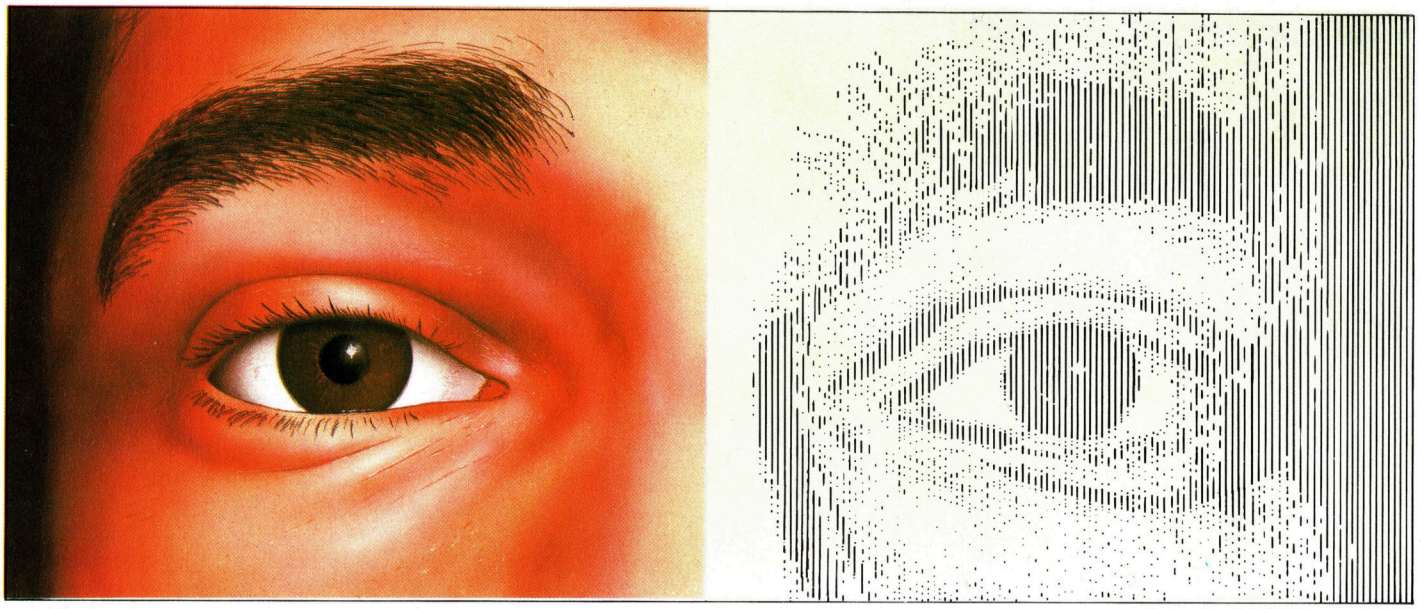
Redaktion: Winfried Schmidt (verantw. f. d. Inhalt), Elke Leibinger, Susanne Brandt, Uta Brandl (Layout), Sammelwerk Redaktions-Service GmbH, Paulstraße 3, 2000 Hamburg 1

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



© APSIF, Copenhagen, 1982, 1983; © Orbis Publishing Ltd., 1982, 1983; © Marshall Cavendish Ltd., 1984, 1985;

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



Sehender Roboter

In dieser Folge wird erläutert, welche Probleme damit verbunden sind, einen Roboter Dinge „sehen" zu lassen.

Die Fähigkeit des menschlichen Sehvermögens beruht im wesentlichen auf dem Wechselspiel zwischen einem komplexen Nervensystem und den Rezeptoren. Obwohl ein Bild aus hellen und dunklen Mustern besteht, die die Retina empfängt, erfolgt das eigentliche „Sehen" im Gehirn. Auch das Hirn eines Roboters berechnet ein Bild aus hellen und dunklen Mustern, arbeitet aber weniger genau und detailliert.

Der vielleicht wichtigste aller menschlichen Sinne ist der Gesichtssinn. Unsere visuellen Eindrücke sind für das Verstehen der Umwelt sehr bedeutsam. Die Redewendung „Das ist, als würde man versuchen, einem blinden Menschen Farbe zu beschreiben" wird daher oft benutzt, um die Schwierigkeiten aufzuzeigen, die mit der Erklärung einer bestimmten Sache verbunden sind. Wie Roboter mit Hilfe von Sensoren das Vorhandensein von Gegenständen erkennen können, haben wir bereits dargelegt. Hier geht es darum, ein System zu entwickeln, das Ihnen ermöglicht, ebenso zu „sehen" wie ein Mensch.

Die Iris des menschlichen Auges dient als Blende. Durch sie wird die Menge des eintretenden Lichtes gesteuert, und das Abbild wird durch die Linse auf die Netzhaut gelenkt. Doch eigentlich „sieht" das Auge überhaupt nichts. Es arbeitet lediglich als „Reizimpulsmittler", der ein Signal in ein anderes umwandelt. Das eigentliche Sehen findet im Gehirn statt, basierend auf den Signalen, die seine Sensoren empfangen haben.

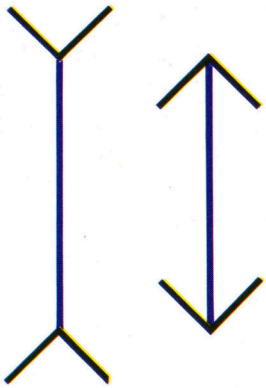
Beim Sehvermögen des Roboters müssen wir also auch zwei unterschiedliche Bereiche betrachten. Einmal geht es um die Konstruktion eines entsprechend geeigneten „Auges", das als Sensor im Roboter-System dient. Zum anderen geht es um die Verarbeitung im Computer. Sie muß beendet sein, bevor der Roboter die Signale umsetzen kann.

Die Konstruktion eines Roboter-Auges ist nicht schwer. In der einfachsten Form kann eine Fotozelle als Auge eingesetzt werden. Diese leitet ein Signal weiter, das der Gesamtausleuchtung des Blickfeldes entspricht. Als „Augensinn" oder „Sehen" kann man dies nur schwerlich bezeichnen. Dafür ist ein visuelles System erforderlich, das ein totales zweidimensionales Abbild der Umwelt vermittelt und so das „Gehirn" des Roboters befähigt, dieselbe Information zu bekommen und zu interpretieren wie das menschliche Hirn. In den meisten Fällen wird eine fotoelektrische Zelle mit einer davor befindlichen Linse verwendet. Diese tastet das Blickfeld vor dem Roboter ab, indem sie so lange über das gesamte Blickfeld geschwenkt wird, bis ein komplettes Bild aufgezeichnet ist. Das Bild wird anschließend im Computer gespeichert. Leider erweist sich diese Methode in der Praxis als zu langsam und aufwendig.

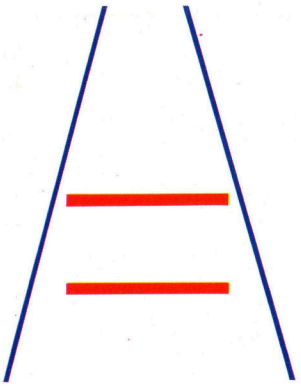
Optische RAMs

Üblicherweise besteht das Auge eines Roboters deshalb aus einer Videokamera. Dabei kann es sich um einen Kameratyp handeln, wie er beim Fernsehen Anwendung findet, oder eine spezielle Kamera, die eigens für Roboter entwickelt wurde. Beim Roboter werden besondere Chips, bezeichnet als „optische RAMs", verwendet. Dabei handelt es sich um RAM-Speicher, in denen der Wert jedes Bytes automatisch durch die Menge des auf das spezielle Byte einwirkenden Lichtes ermittelt wird. Diese Speicher enthalten alle Informationen,

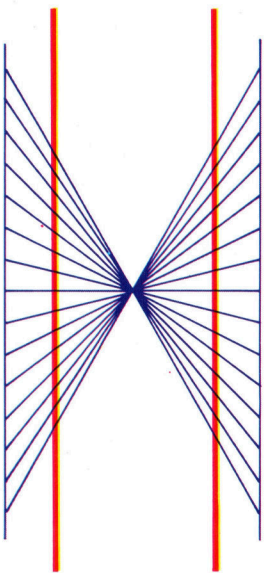
Was man zu sehen glaubt



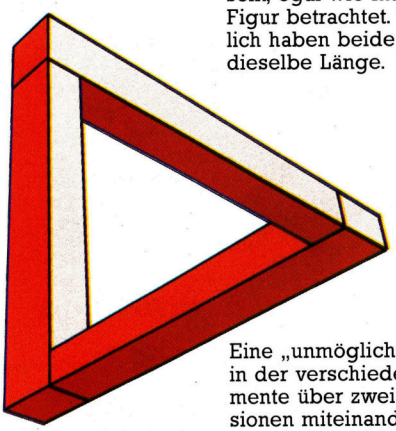
Welches Element ist länger? Muller-Lyer-Täuschung vermittelt den Eindruck, als sei die Senkrechte in der linken Figur länger als die in der rechten. Tatsächlich sind beide gleich lang.



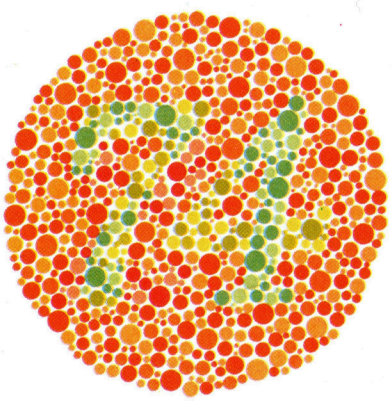
Bei der Schienendarstellung scheint der obere Querbalken länger zu sein, egal wie man die Figur betrachtet. Tatsächlich haben beide Balken dieselbe Länge.

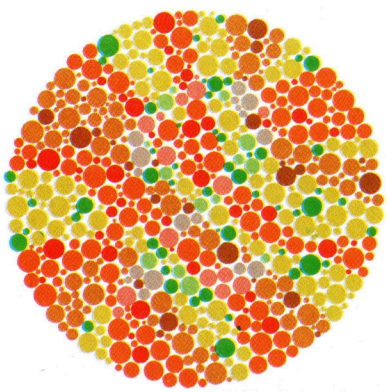


Obwohl die beiden vertikalen Linien gebeugt zu sein scheinen, sind sie in Wirklichkeit völlig gerade. Das Auge „beugt" das Bild, um den Strahlenverlauf auszugleichen.



Eine „unmögliche" Figur, in der verschiedene Elemente über zwei Dimensionen miteinander verbunden sind. Forschungen ergaben, daß Menschen solche Figuren nicht als Gegenstand erkennen können.





Im ersten Bild wird ein Mensch mit normalem Farbunterscheidungsvermögen die Zahl 74 lesen, wogegen Personen, die Schwierigkeiten bei der Rotgrün-Unterscheidung haben, die Zahl 21 erkennen. Im zweiten Beispiel sieht ein Normalsehender entweder gar nichts oder nur eine vage Kontur der Zahl 2, wogegen ein Rotgrün-Blinder deutlich eine 2 erkennt.

die das vom Roboterauge wahrgenommene Umfeld betreffen.

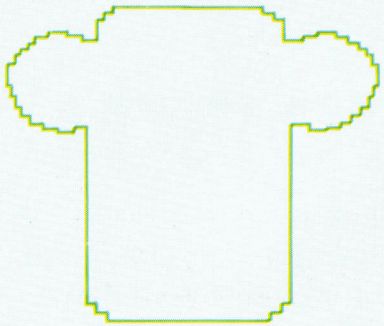
Grundsätzlich gilt: Die Ausgabe eines Roboterauges wird in einem zweidimensionalen Array festgehalten. (Array = rasterartige Anordnung von Elementen in einem System. Die Anordnung besteht meist in Mustern, die eine bestimmte Regelmäßigkeit aufweisen. Typisches Beispiel für ein biologisches Sensor-Array ist die Netzhaut des menschlichen Auges.) Jedes Element enthält einen Wert, der der Helligkeit des einfallenden Lichtes entspricht. Die Anzahl der Elemente in diesem Array vermittelt die Auflösung des Bildes, und die Menge der Zahlen, die in jedem Array-Element festgehalten werden können, bestimmt die Anzahl der wahrnehmbaren Grauwerte. Traditionell werden Array-Elemente in optischen Systemen als „Pixels" oder „Bildelemente" bezeichnet. Ein Array von 500 x 250 Punkten stellt Helligkeitswerte dar, indem er jedem Punkt ein Byte zuweist. Horizontal beträgt die Auflösung dann 500 Punkte, vertikal 250 Punkte, woraus sich insgesamt 125 000 Punkte ergeben. Die 256 Grauwerte reichen von Tiefschwarz bis zum reinen Weiß. Betrachten wir einmal ein normales Fernsehgerät, um uns diesen Auflösungsgrad zu vergegenwärtigen. Beim PAL-System werden 625 vertikale Zeilen benutzt. Die vertikale Auflösung beträgt also 625 Bildpunkte. Um eine ähnlich hohe Auflösung auf dem gesamten Schirm zu erreichen, wären etwa 1000 Punkte erforderlich (da der Schirm breiter als höher ist). Ein Roboter-System mit entsprechender Auflösung würde dem Computer ein für die Verarbeitung akzeptables Bild liefern.

Die zum Erkennen eines Bildes erforderlichen Schritte eines Roboter-„Hirns" folgen einem vorgegebenen Muster. Beim ersten Schritt werden die Grauwerte so geordnet, daß angrenzende Bildpunkte mit ähnlichen Grauwerten auf einen „gemeinsamen Nenner" gebracht werden. Der Computer rechnet diese Grauwerte durch und paßt sie einander an. Danach vergleicht der Computer die Grauwerte nochmals und registriert die von diesem Mittelwert abweichenden Bildpunkte, die dann hervorgehoben werden. Somit können besonders wichtige Elemente wie Linien und Kanten, die als starke Abweichungen in der Grauwertskala auftreten, speziell dargestellt werden.

Die visuelle Wahrnehmung

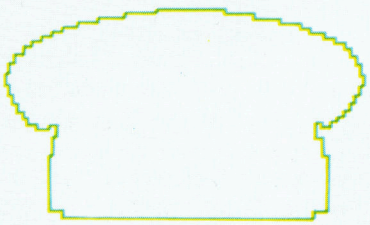
Danach erfolgt eine weiteres Abtasten des Bildes, wobei der Rechner die extremen Grauwertabweichungen registriert. Diese werden dann so einander zugeordnet wie die Einzelteile eines Puzzles, die zu einem Bild zusammengesetzt werden sollen. Am Ende dieses Prozesses steht dem Roboter ein internes Bild zur Verfügung, das „geglättet" ist. Das bedeutet: Die unwichtigen Bildelemente sind eliminiert und die wichtigen hervorgehoben.

Aber ist das wirkliches „Sehen"? Tatsächlich hat der Roboter nichts weiter getan, als eine bestimmte Bildumwandlung zu vollziehen. Er „weiß" immer noch nicht, was er sieht. Für dieses Problem gibt es zwei Lösungen. Die eine besteht darin, den Roboter mit einem Programm auszustatten, das einfache Feststellungen zum Erkennen des Umfeldes enthält. Das wird als „bottom-up" der visuellen Wahrnehmung bezeichnet. Diese Methode wird so benannt, weil der Roboter mit dem Erkennen einfacher Gegenstände anfängt und darauf basie-

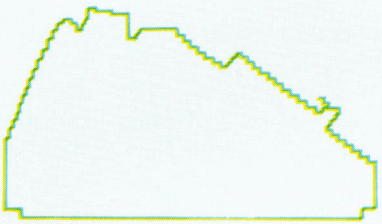


Kontur von oben

Gegenstand



Kontur von vorn



Kontur von der Seite

Objekt-Erkennung

Um ein Objekt kennenzulernen, folgt ein Roboter einem Lernprozeß, der dem des Menschen ähnlich ist: Er vergleicht den Gegenstand so lange mit einem bekannten Objekt, bis er etwas Passendes gefunden hat. Roboter haben allerdings begrenztere Sehmöglichkeiten und können nur wenige Objektmuster speichern. Ohne menschliche Erfahrung und die hochentwickelten Vergleichsmöglichkeiten des Menschen könnte der Roboter die sich aus den Sichtinformationen ergebenden Konturen und Formen nicht als „Telefon" erkennen, selbst wenn er es aus verschiedenen Perspektiven sieht.

rend eine komplexere Verständnisebene entwickelt. Die zweite Möglichkeit besteht darin, den Roboter mit einer Reihe von Gegenständen zu programmieren, die er identifizieren kann, und ihn dann diese Abbilder mit der Wirklichkeit vergleichen zu lassen. Dies wird als „top-down" bezeichnet, weil der Roboter mit detaillierten Vorgaben programmiert ist und diese mit den tatsächlichen visuellen Eingaben zu vergleichen hat.

Um den Unterschied der beiden Methoden zu verdeutlichen, nehmen wir einmal an, daß der Roboter einen Tisch sieht. Bei der „bottom-up"-Erkennung erfolgt eine Analyse des Abbildes und daraus resultierend die Feststellung, daß der Gegenstand aus vier vertikalen Teilen und einer großen horizontalen Fläche besteht. Das entspricht dem vorprogrammierten Wissen, daß eine große Oberfläche auf vier Beinen ruhen kann und die Gesamtheit dann als „Tisch" bezeichnet wird. Die „top-down"- Erkennung würde damit beginnen, daß der Roboter auf den Tisch schaut und im Speicher abfragt „Ist dies ein Tisch?". Er kann diese Frage stellen, da er intern über ein optisches Modell des Tisches verfügt, das er mit der visuellen Eingabe zu vergleichen hat.

Grundsätzlich bedeutet das: Mit der „bottom-up"-Erkennung vermag der Roboter Dinge zu identifizieren, denen er zuvor nicht begegnet ist, und sie zu verstehen. Das setzt aber eine entsprechend umfangreiche, detaillierte Programmierung voraus. Dagegen erkennt der Roboter mittels der „top-down"-Methode lediglich Gegenstände, die bereits gespeichert sind. Alles, was neu ist, verursacht Probleme.

Doch die Sehfähigkeit der Roboter ist noch immer nicht optimal. Dafür gibt es verschiedene Gründe. Einer der wichtigsten ist die zur Berechnung eines Bildes erforderliche Speicherkapazität. In unserem Beispiel waren die 125 000 Punkte jeweils ein-Byte-weise gespeichert. Für die Erzeugung jedes Bildes sind also insgesamt 122 KBytes Speicherkapazität erforderlich. Wenngleich wir unsere Beschreibung vereinfacht haben, sind die für jeden Punkt auszuführenden Berechnungen mathematisch weitaus komplizierter. Sollte der Roboter die Umwelt in „Echtzeit" betrachten können, müßte er pro Sekunde 25 verschiedene Bilder erfassen (was der Aufnahmegeschwindigkeit einer Fernsehkamera entspricht). Gleichbedeutend wäre dies einer Berechnung von jeweils 3050 KBytes an Daten pro Sekunde.

Limitierte Möglichkeiten

Zwei Lösungsmöglichkeiten stehen für dieses Problem an. Die eine ist, eine spezielle Hardware zur Bildberechnung zu verwenden. Alternativ ließen sich Bildauflösung und Grauwerte so reduzieren, daß sie dem derzeit gültigen Hardware-Anspruch genügten. Die Bildverarbeitung erfolgte damit schneller, die Bildqualität wäre durch dieses Verfahren allerdings erheblich reduziert.

Zum gegenwärtigen Zeitpunkt unterliegt das Sehvermögen der Roboter noch zahlreichen Einschränkungen. Bei der Benutzung eines Sehsystems treten häufig Fehler auf. Die einzige Lösung mag darin bestehen, Systeme zu entwickeln, mit denen Roboter zu sehen „lernen", statt mit bestimmten Vorgaben programmiert zu werden. Möglicherweise werden Roboter so lange Dinge nicht „sehen" können, bis ein Weg gefunden ist, ihnen Wissen um die Umwelt zu vermitteln – ein Wissen, das letztlich dem unseren entspricht.

KHLHMPH CHMFLHQVSUDFKH

Eine der ersten Aufgaben für Computer war die Ver- und Entschlüsselung von Informationen. Heutzutage kann man schon mit BASIC einen Text in Geheimcodes umsetzen – und natürlich auch wieder entschlüsseln.

Jede Art von Kommunikation geht nach einem mehr oder weniger festgelegten Code vor sich. Egal, ob man spricht oder schreibt – sinnvoll wird jede Nachricht erst, wenn der Empfänger sie verstehen kann. Das ist beim Umgang mit dem Computer nicht anders. Als Verbindung zur Außenwelt „versteht" fast jeder Rechner BASIC. Um die Befehle jedoch zu verarbeiten, müssen sie noch in eine rein numerische Form übersetzt werden. Erst nach dieser Übersetzung kann der Computer das Gewünschte in Gang setzen.

Andere Formen der Verschlüsselung haben genau das Gegenteil von Kommunikation zum Ziel: Nachrichten sollen ausschließlich vom eingeweihten Empfänger und sonst niemandem verstanden werden. Bis in die zweite Hälfte des 20. Jahrhunderts bedienten sich nur die Großindustrie und die Regierung solcher „Kryptogramme" (lat. Krypto- = geheim, verborgen). Die Benutzung des leicht abhörbaren Telefonnetzes machte die Verschlüsselung später zu einem „Muß", speziell bei der Übermittlung von Daten mit kommerziellem Wert.

Verschieben oder Austauschen?

Das Addieren oder Subtrahieren eines vorgegebenen Wertes zu jedem Byte oder das Ersetzen bestimmter Buchstaben durch andere nach einem festen Schema zählt zu den einfacheren Methoden der Verschlüsselung. Es gibt jedoch auch Techniken, die sich nur mit den Erkenntnissen der Zahlentheorie umsetzen lassen. Bei Codes dieser Art kommen Wiederholungen nicht vor.

Eine der einfachsten Methoden sinnvoller Verschlüsselung ist „Cäsars Code", von dem behauptet wird, daß er bereits im alten Rom eine Rolle gespielt haben soll. Zum Dechiffrieren sind dabei nur die Nachricht selbst und der richtige Schlüssel nötig. Eine einfache Nachricht sieht beispielsweise so aus:

```
HMQVWHMKHQ YHUVWHLHQ
EHLHUUVFLHQ
```

Es scheint offensichtlich zu sein, daß die Nachricht aus drei Wörtern besteht: Das erste hat zehn, das zweite neun und das dritte elf Buchstaben. Außerdem ist noch bemerkenswert, daß alle Wörter mit dem gleichen Buchstaben enden und daß der Buchstabe H besonders häufig ist.

Bei einfachen Codes ist die Häufigkeit bestimmter Buchstaben oft eine große Hilfe zum Entziffern; denn in jeder geschriebenen Sprache werden bestimmte Buchstaben häufiger verwendet als andere. Nutzbringend wird diese Aussage allerdings nur, wenn auch bekannt ist, in welcher Sprache die Botschaft geschrieben ist. So ist etwa im Englischen der häufigste Buchstabe das E, gefolgt vom T, während im Deutschen E und A führen.

Neben der Häufigkeit sollte man auch auf die Rechtschreibung achten: Bestimmte Kombinationen von Buchstaben sowie Anfangs- und Endbuchstaben lassen oft Vermutungen zu, die beim „Codeknacken" hilfreich sind. Lassen Sie uns davon ausgehen, daß die Nachricht in Deutsch geschrieben ist, das „H" wäre also vermutlich durch „E" zu ersetzen:

```
eMQVWeMKeQ YeUVWeLeQ EeLeUUVFLeQ
```

Das gibt keinen Sinn, aber mit Nachdenken kommen wir weiter: Das H liegt im Alphabet drei Plätze hinter dem E. Also versuchen wir unser Glück und ersetzen den zweithäufigsten Buchstaben (Q) ebenfalls durch seinen drittnächsten Nachbarn, das N. Jetzt sieht die Nachricht folgendermaßen aus:

```
eMnVWeMKen YeUVWeLen EeLeUUVFLen
```

Sie sehen, daß alle drei Wörter mit „en" aufhören, was im Deutschen besonders bei Verben recht häufig vorkommt. Die Endung „-en" könnte also bedeuten, daß wir uns auf dem richtigen Weg befinden. Lassen Sie uns also auch die anderen Buchstaben des ersten Wortes im Alphabet um drei Plätze verschieben – schon haben wir das Wort "EINSTEIGEN"!

Cäsars Code ist eine Verschlüsselungsmethode, nach der die Buchstaben im Alphabet um einen festgelegten Wert verschoben werden. Sie läßt sich verfeinern, indem dieser feste Wert durch eine Folge von Werten ersetzt

Cäsars Code

Dieses Programm (in Commodore-BASIC geschrieben) chiffriert einen Text nach Cäsars Code und variiert dabei den Schlüssel fünffach. Bei der Eingabe erscheint die Nachricht ganz normal auf dem Bildschirm. Nach Drücken von RETURN wird die verschlüsselte Botschaft ausgedruckt. In der Botschaft dürfen keine Leerzeichen oder Punkte vorkommen.

```
10 INPUT "GEBEN SIE EINEN
   5-STELLIGEN SCHLUESSEL EIN";K$
20 INPUT "GEBEN SIE DIE MELDUNG
   EIN";M$
30 FOR I=1 TO LEN(M$)
40 LET J=I — INT(I/5)*5+1
50 REM ***SCHLUESSELKOMBINA-
   TIONEN
60 LET M=ASC(MID$(M$,I,1)) — VAL
   (MID$(K$,J,1))
70 IF M<65 THEN LET M=M+26
80 PRINT CHR$(M);
90 NEXT I
```

Beim Spectrum muß Zeile 60 wie folgt ersetzt werden:

```
60 LET M=CODE(M$(I)) — VAL(K$(J))
```

wird – zum Beispiel 1234. In diesem Fall würde der erste Buchstabe um 2 Plätze, der zweite um 4 Plätze, der dritte wieder um 2 Plätze verschoben – und so fort. Am Ende der Ziffernfolge beginnt man wieder mit der ersten Verschiebungsanweisung. Nach Bearbeitung mit diesem Schlüssel sähe das erste Wort unserer Geheimnachricht jetzt so aus:

FLQWUGMLFP

Eine Entzifferung nach der Methode der Häufigkeiten ist jetzt unmöglich, weil jeder Buchstabe in Abhängigkeit von seiner Position im Gesamttext durch andere Buchstaben ersetzt wird. Ein anderer in sich geschlossener Code verwandelt die ganze Nachricht in

ETEEE ECISEGNVRTHNBHRSHNNI SEERE

Bei genauem Hinsehen können Sie feststellen, daß diese Zeichenfolge ein komplettes Anagramm (Buchstabenvertauschung) von "EINSTEIGEN VERSTEHEN BEHERRSCHEN" ist, sogar die beiden Leerzeichen sind da. Wenn sowohl der Quell- wie auch der verschlüsselte Text vorhanden sind, ist die Codier-Methode sehr leicht zu finden. Der Empfänger braucht zum Entziffern nur eine einzige Zahl als Schlüssel – in diesem Fall ist es die 3. Nehmen Sie die ersten acht Zeichen und schreiben Sie sie mit jeweils drei Leerstellen dazwischen hintereinander auf:

E***T***E***E***E*** ***E***C

Merken Sie etwas? Der Urtext wird in drei übereinanderliegenden Zeilen geschrieben, wobei man im Zickzack auf- und abspringt:

```
E   T   E   E   E   *   E   C
 I S EG NV R T H NB H R S H N
 N   I   *   S   E   E   R   E
```

Die Sternchen stehen für die Leerstellen – damit ist die Chiffriermethode klar.

Die hier vorgestellten Beispiele haben alle gemeinsam, daß einzelne Buchstaben nach einem bestimmten Schlüssel im Alphabet verschoben bzw. durch andere Buchstaben ersetzt werden. Es gibt aber auch andere Methoden, bei denen ganze Blöcke durch andere, meist kleinere Blöcke, ersetzt werden. Leider haben sie den Nachteil, daß Sender und Empfänger der Nachricht über ein Codebuch verfügen müssen. Dafür läßt sich etwa ein Roman oder eine Zeitung verwenden. Die Nachricht besteht dann nur aus einer Ziffernfolge, etwa 17,13,9. Wenn Sie dazu den richtigen Text haben, in unserem Fall: „Gathmann konnte sich vor Hunger einfach nicht mehr beherrschen. Keiner konnte nachher verstehen, wie ihm das Einsteigen in die Kantine des Landeskrankenhauses mit Gipsbein und Zwangsjacke gelungen war", dann ist es natürlich ganz leicht, auf die Mitteilung zu kommen. Oder?

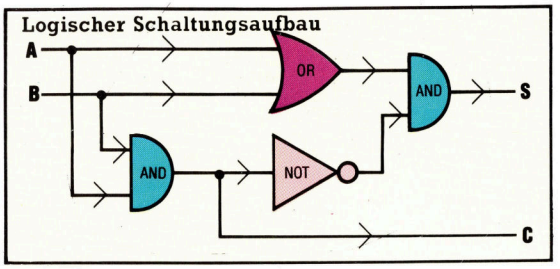
Computer eignen sich ganz ausgezeichnet zum Ver- und Entschlüsseln von Nachrichten. Um mit Cäsars Code zu arbeiten, muß der Rechner ausschließlich Zeichen eines alphanumerischen Strings verschieben. Richtig programmiert, macht's

YMHO YHUKQXHKHQ

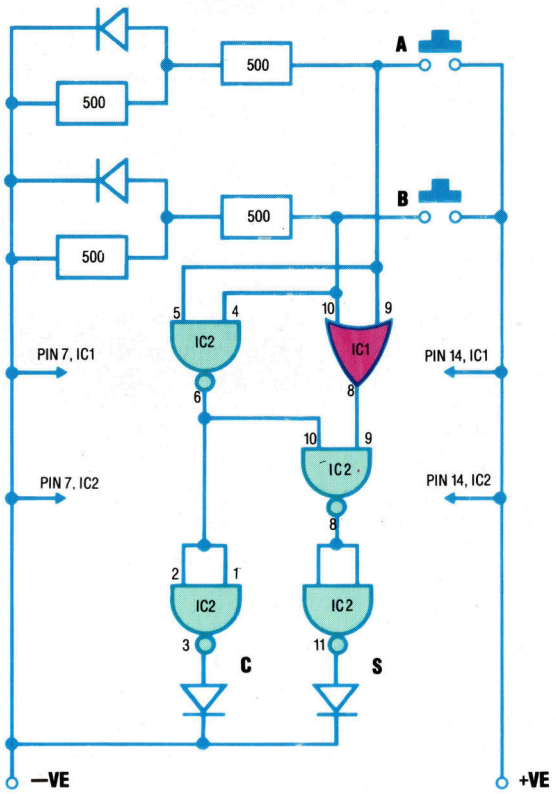
Eine der ersten Aufgaben für die gerade erfundenen Computer war die Dechiffrierung komplizierter Geheimcodes im Zweiten Weltkrieg. In Deutschland hatte man zur Erzeugung neuer Codes eigens eine Maschine entwickelt, die legendäre ENIGMA. Die Entschlüsselung der ständig wechselnden ENIGMA-Codes bereitete den Alliierten große Schwierigkeiten. Erst der Colossus-Entwicklungsgruppe in Bletchley Park gelang schließlich diese mühsame Aufgabe. In der Colossus-Gruppe war auch Alan Turing tätig – er gilt heute als einer der „Väter" des Computers.

Halbaddierer mit Logik-Gattern

In dieser Folge unseres Bastelkurses wollen wir über die AND-, OR- und NOR-Gatter hinausgehen und mit zwei integrierten Schaltkreisen einen Halbaddierer aufbauen.



Elektrischer Schaltungsaufbau



Die einzelnen Logik-Gatter aus dem letzten Kursteil stellen die Grundlage komplexerer Digital-Schaltkreise dar. Mit einem Halbaddierer lassen sich zwei Einzel-Bits addieren. Er stellt zwei Eingänge für die zu addierenden Bits sowie zwei Ausgänge bereit: einen Summen- und einen Übertragsausgang. Mit einer Wahrheitstabelle läßt sich das Verhalten des Halbaddierers so darstellen:

A(=1.Bit)	B(=2.Bit)	S(=Summe)	Ü(=Übertrag)
0	0	0	0
1	0	1	0
0	1	1	0
1	1	0	1

Der Ausgang S zeigt immer die Summe der beiden Eingangsbits. Sind beide Bits 1, ergibt sich im binären System 10, für deren Darstellung ein zusätzlicher Ausgang Ü (Übertrag) nötig ist. Ausgang S zeigt bei 10 also 0, Ausgang Ü die 1.

Wenn (in einem Acht-Bit-Rechner) zwei Acht-Bit-Worte addiert werden sollen, kommt man mit einem einzigen Halbaddierer nicht weit. Erst 16 Halbaddierer zusammen vollbringen dieses Kunststück: Die beiden ersten Bits werden vom ersten Halbaddierer zusammengezählt, der am Summenausgang die letzte Stelle des Ergebnisses zeigt. Der jeweilige Übertrag wird zum Additionsergebnis der nächsten zwei Bits hinzugezählt, der Übertrag dieser Addition wird zur Summe des dritten Bits addiert – und so weiter.

Auch ein ganz primitiver Halbaddierer würde aus mindestens zehn Transistoren bestehen, wenn er aus den uns bekannten Gatterschaltungen aufgebaut wäre. Zum Glück gibt es AND, NAND, OR, NOR und andere logische Gatter als IC. Mit jeweils vier Gattern pro Bauteil wird somit die Schaltung nicht nur einfacher, sondern auch preisgünstiger.

Einen einfachen, aus OR-, AND- und NOT-Gattern aufgebauten Addierer zeigt der abgebildete Schaltplan. Da die hier verwendeten ICs nur mit jeweils gleichen Gattern erhältlich sind, haben wir die Schaltung durch Verwendung möglichst weniger Gatter-Typen vereinfacht: Es sind nur vier NAND- und ein einziges OR-Gatter nötig; daher brauchen Sie insgesamt nur zwei ICs.

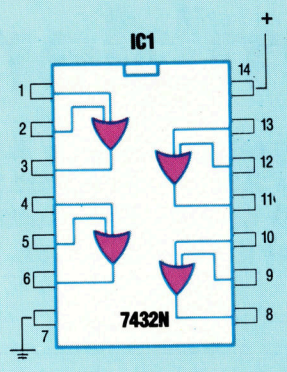
Sie werden wahrscheinlich bei der Montage feststellen, daß für die vergleichsweise geringe Leistung dieser Konstruktion doch recht viel Mühe nötig ist. Es geht zwar sehr viel schneller als mit Einzel-Bauteilen – aber soll man einen kompletten Computer auf diese Art herstellen?

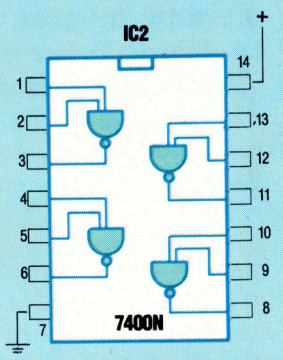
In der Praxis werden ICs nur selten in dieser Weise verwendet – viel häufiger kommt es vor, daß sie im Computer für einen kleinen „Aushilfs-Job" abkommandiert werden. Die komplexeren Aufgaben sind den größeren Chips mit mehr Ein- und Ausgängen vorbehalten. Damit lassen sich dann etwa zwei 4-Bit-Zahlen addieren.

Das brauchen Sie:

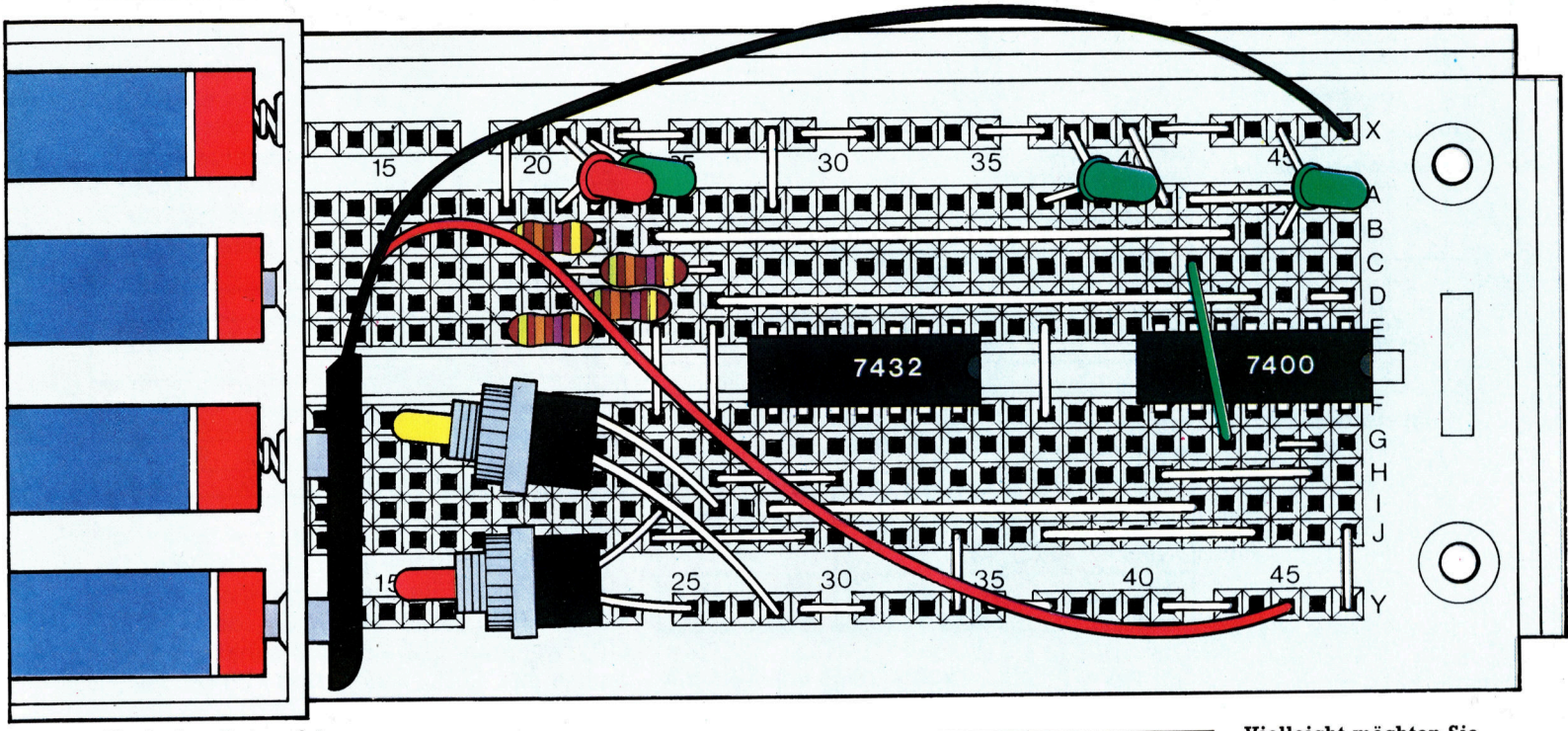
4 Widerstände, 500 Ohm, 0,25 Watt
4 Leuchtdioden (LEDs)
2 Taster (Schließer)
1 IC 7400N
1 IC 7432N
4 Baby- oder Monozellen 1,5 Volt
1 Batteriehalterung
1 Batterieclip
1 Experimentier-Stecksockel
einige kurze Drahtstücke

Anschlußbelegung

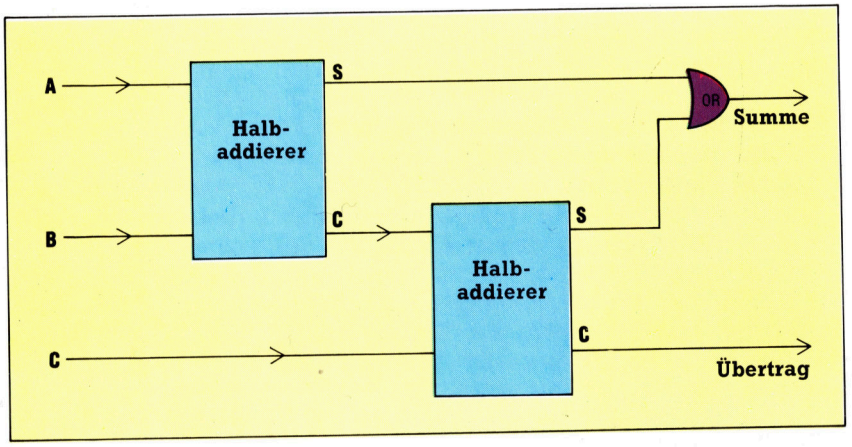




Bei den verwendeten ICs handelt es sich um 14-Pin-TTL-Bauteile. Der 7400N enthält vier NAND-Gatter, der 7432N stellt vier OR-Gatter zur Verfügung. Auf der Zeichnung können Sie erkennen, wie die Gatteranschlüsse liegen. Sie brauchen nur noch für die richtige Verdrahtung zu sorgen. Auf einer Seite des Chips ist ein Ausschnitt, der zeigt, wie das Bauteil eingesetzt werden muß. Denken Sie auch an die Versorgungsspannung, die über Pin 7 und 14 eingespeist wird.



Nach dem Entwurf der Schaltung muß man sich Gedanken über die Anordnung der Bauteile auf der Platine machen. Für die Konzeption können Sie spezielle Vordrucke verwenden, die Fotokopie einer leeren Platine erfüllt jedoch den gleichen Zweck. Am einfachsten wird es, wenn der reale Schaltungsaufbau dem Plan ähnelt – man irrt sich nicht so leicht bei der Verdrahtung. Wenn Sie sich genau an das Original halten, kann nichts schiefgehen – alle Bauteile sitzen an der richtigen Stelle.



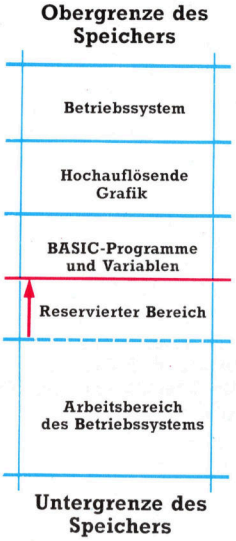
Vielleicht möchten Sie zur Übung Ihren Halbaddierer zu einem Volladdierer aufrüsten. Mit der hier abgebildeten Schaltung werden nicht einfach nur zwei Bits addiert, sondern auch der Übertrag vom vorhergehenden Bit ist berücksichtigt. Eine Gruppe von Volladdierern kann komplette Bit-Worte zusammenrechnen. Im einfachsten Fall werden dazu zwei Halbaddierer wie hier im Bild zusammengeschaltet. Dabei gelangt das Summensignal des ersten Halbaddierers auf einen Eingang des nächsten Addierers.

Assemblerbefehle

Maschinencodeprogramme, die im Assemblerformat geschrieben werden, müssen am Anfang bestimmte Anweisungen enthalten. In dieser Folge wird untersucht, welche Funktionen einige dieser „Assembleranweisungen" ausführen.

Acorn B

Direkte Eingabe des Maschinencodes:
PRINT. ~ PAGE
gibt die Anfangsadresse des reservierten Speicherbereiches in Hex an. Geben Sie dann ein:
PAGE=PAGE+N
wobei N die Anzahl (Dezimalzahl) der Bytes ist, die Sie für Ihren Maschinencode reservieren wollen.



In der letzten Folge haben wir ein einfaches Maschinencodeprogramm geschrieben, das zwei Zahlen im Akkumulator addiert und im Speicher ablegt. Obwohl das Programm sehr einfach war, wurden dadurch viele grundlegende Abläufe der Maschinencodeprogrammierung deutlich. Sehen Sie sich dieses Programm nochmals mit den zugehörigen Adressen an. Darin wurden der Code von der Speicherstelle $0000 an geladen und das Resultat auf die Speicheradresse $0009 gelegt. (Dies ist jedoch nur für Anschauungszwecke gedacht – jeder Versuch, diese Speicheradressen einzusetzen, würde den sofortigen Absturz des Systems verursachen.) Hier die Programmversionen:

Speicheradresse	Maschinencode	Assemblerbefehle
	Z80	
0000	A7	AND A
0001	3E 42	LD A,$42
0003	CE 07	ADC A,$07
0005	32 09 00	LD BYTE1,A
0008	C9	RET
	6502	
0000	18	CLC
0001	A9 42	LDA #$42
0003	69 07	ADC #$07
0005	8D 09 00	STA BYTE1
0008	60	RTS

Der vierte Befehl (der den Inhalt des Akkumulators in $0009 speichert) enthält im Assemblercode beider Programme keine Hexadezimaladresse, sondern ein Symbol: BYTE1. In der Maschinencodeversion dieses Befehls finden wir jedoch den Op-code für "Lade die folgende Speicherstelle aus dem Akkumulator", gefolgt von 09 00, den zwei Bytes der Adresse $0009 im lo-hi-Format.

Hier zeigt sich ein interessanter Mechanismus der Übersetzung des in Assembler geschriebenen Programms in den Maschinencode. Ebenso wie zur Vereinfachung der Programme die Hexzahlen der Befehle durch mnemotische Kürzel ersetzt wurden (beispielsweise 8D und C9 durch STA und RET), können schwer lesbare Adressen wie $0009 durch Symbole wie BYTE1 ersetzt werden. Dieser Vorgang ist der gleiche wie die Initialisierung von Variablen in BASIC. Auch die Gründe sind die gleichen: Programme werden übersichtlicher, die Möglichkeit, Fehler einzugeben, verringert sich, und der Code ist leichter zu bearbeiten. Wenn beispielsweise im gesamten Programm eine Adresse geändert werden soll, genügt es, dem entsprechenden Symbol anfänglich einen anderen Wert zuzuordnen, den das gesamte Programm ohne weiteres Editieren verwendet.

In der BASIC-Programmierung ist dieser Vorgang leicht zu verstehen. Wie aber sieht der Assemblerbefehl für den BASIC-Befehl LET BYTE1=$0009 aus? Wenn zur Übersetzung in den Maschinencode kein Assemblerprogramm eingesetzt wird, müssen wir diese Arbeit selbst ausführen. Bei Verwendung des Assemblers aber brauchen wir nur den entsprechenden Befehl an den Anfang unseres Codes zu setzen. Hier die Version für den Z80:

Z80		
0000		BYTE1 EQU $0009
0000	A7	AND A
0001	3E 42	LD A,$42
0003	CE 07	ADC A,$07
0005	32 09 00	LD BYTE1,A
0008	C9	RET

BYTE1 erhält dabei eine eigene Spalte, die Symbol-(oder Label-)feld genannt wird. Im Op-codefeld steht ein neues mnemotisches Kürzel (EQU bedeutet "Weise den Wert ... zu"), während das Operandenfeld den Wert enthält, der BYTE1 zugeordnet wird (in diesem Fall $0009).

Beachten Sie, daß EQU zwar im Op-codefeld eingetragen ist und wie ein Maschinencodebefehl aussieht, aber dennoch weder Sprachelement des Assemblers noch des Maschinencodes ist. Ein derartiges mnemotisches Kürzel wird „Pseudo-Befehl" genannt. EQU teilt dem Assembler mit, daß er bei der Übersetzung in den Maschinencode das vorstehende alphanumerische Symbol (hier: BYTE1) im gesamten Programmtext durch den nachstehenden Wert (hier: $0009) ersetzen soll. Wie Sie bereits wissen, wird beim Einsatz eines Assemblerprogramms der Code in der Assemblersprache geschrieben (entweder zur Speicherung auf Cassette oder Diskette oder direkt in den Arbeitsspeicher) und dann vom Assembler in die Maschinensprache übersetzt.

Der Ausdruck eines Assemblerprogramms besteht normalerweise aus Assemblerbefeh-

len wie oben aufgeführt und ihrer Entsprechung im Maschinencode in Form von Hex-Bytes. Der Maschinencode kann dabei für den späteren Einsatz gespeichert werden oder zur direkten Ausführung vorgesehen sein. Ein Assembler kann aber auch andere Aufgaben ausführen, z. B. die Eingabe von Programmadressen. Diese Aufgabe übernimmt der Pseudobefehl ORG. Das Programm sieht dann folgendermaßen aus:
Interessant ist hierbei, daß die Adresse der ersten Zeile $0000 ist, die der darauf folgenden Zeile jedoch $A000. Hierin zeigt sich die Auswirkung des ORG-Befehls. In den Op-codefeldern der Pseudobefehle stehen keine Hex-Bytes, da diese nicht Teil des Programms sind und daher auch nicht in den Maschinencode übersetzt werden. Pseudobefehle sind von Assembler zu Assembler verschieden. EQU wird beispielsweise manchmal durch "=" dargestellt und ORG durch ".=". Die Wirkung ist jedoch die gleiche, und wir werden ORG und EQU weiterhin verwenden.

Z80		
0000		ORG $A000
A000		BYTE1 EQU $0009
		AND A
A001	3E 42	LD A,$42
A003	CE 07	ADC A,$07
A005	32 09 00	LD BYTE1,A
A008	C9	RET

Mnemotische Kürzel

Es mag Ihnen vielleicht aufgefallen sein, daß wir von Anfang an schon einen Pseudobefehl eingesetzt haben: "$" – die Markierung der Hexadezimalzahlen. Dieses Zeichen ist eine Anweisung an den Assembler, den folgenden Wert als Hexzahl anzusehen. Auch das Zeichen "#", das wir in der letzten Folge vorgestellt hatten, kennzeichnet eine „Konstante". Es legt fest, daß der darauf folgende Wert eine direkte Zahlenangabe ist und kein Zeiger oder Symbol. Genauer betrachtet ließe sich die gesamte Assemblersprache als eine Folge von Pseudobefehlen ansehen. Und Sie können ihre eigenen mnemotischen Kürzel erfinden – vorausgesetzt natürlich, sie entsprechen unmittelbar dem Befehlssatz des Maschinencodes.

In diesem Kurs verwenden wir natürlich auch weiterhin die vorgegebenen mnemotischen Kürzel. Von Zeit zu Zeit sollten wir uns aber daran erinnern, daß alles, was als Maschinencode bezeichnet wird, nur Symbolcharakter hat. Die CPU verarbeitet Stromimpulse, die über ihre Kontakte geleitet werden. Wie diese Impulsfolgen dargestellt werden, ist nur eine Sache der Übereinkunft.

Nun zu einem anderen Gebiet: den unterschiedlichen Übersetzungen von LDA und LD A. Sehen Sie sich folgende Zeilen an:

```
AD ?? ??    LDA $????    (6502)
3A ?? ??    LD A, ($????)    (Z80)
```

Sie bedeuten beide: "Lade den Akkumulator mit dem Byte der Adresse ????". Der Op-code für diesen Ladevorgang ist $AD (für den 6502) und $3A (für den Z80). Wenn Sie diese Codes mit dem Befehl "Lade den Akkumulator mit der Konstanten $42" unseres Programms vergleichen, dann sehen Sie aber die Op-codes $A9 (6502) und $3E (Z80). Die Unterschiede lassen sich leicht erklären: Obwohl in beiden Fällen die gleiche Befehlsart ausgeführt wird (Daten werden in den Akkumulator geladen), kommen die Daten aus unterschiedlichen Quellen. Die CPU muß daher auch unterschiedliche Op-codes erhalten.

Im ersten Fall werden die Daten aus einem Speicherbyte geladen, dessen Adresse angegeben ist. Über den Inhalt des Bytes ist dabei nichts bekannt. Die CPU erhält nur die Anweisung, das Byte in den Akkumulator zu kopieren. Die drei Bytes des Maschinencodes AD ?? ?? und 3A ?? ?? befehlen der CPU "Interpretiere die zwei Bytes, die diesem Op-code folgen, als die absolute Adresse der Datenquelle".

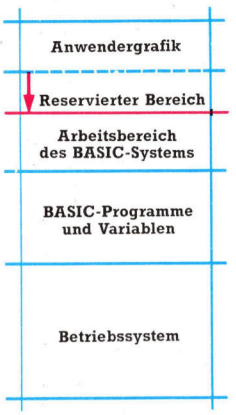
Im zweiten Fall bestehen die Daten, die in den Akkumulator geladen werden sollen, aber aus dem Byte, das dem Op-code unmittelbar folgt. Die Maschinencodebytes A9 42 und 3E 42 teilen der CPU mit: "Interpretiere das Byte, das dem Op-code folgt, als Datenquelle". Die Op-codes A9 oder 3E enthalten daher den Befehl, das auf den Op-code folgende Byte in den Akkumulator zu laden. Da der Programmzähler immer die Adresse des nächsten Befehls enthält, kann die CPU die Adresse des Quellenbytes berechnen und dann das Byte der berechneten Adresse laden.

Diese beiden Beispiele zeigen, wie einfach und unkompliziert die Vorgänge innerhalb der CPU sind. Eine Befehlsklasse (etwa ein Fünftel des gesamten Instruktionssatzes) besteht nur aus Kopiervorgängen, die die Daten eines adressierten Bytes in eines der internen Register laden. All diese „primitiven" Kopiervorgänge unterscheiden sich nur durch das Format, das die Adresse des Quellenbytes darstellt.

Es mag zunächst vielleicht etwas verwirrend sein, wenn wir derart ausführlich auf die Vorgänge im Inneren der CPU eingehen. Wenn Sie nur den Assemblercode eines bestimmten Prozessors lernen wollen, genügt es, sich den Befehlssatz und ein paar Tricks anzueignen, und Sie können mit der Programmierung anfangen. Sind Sie jedoch mit den Grundlagen vertraut, dann haben Sie nicht nur eine weitere Programmiersprache erlernt, sondern können sich leichter in die Abläufe anderer Prozessoren und Assemblerprogramme einfinden. Dadurch wird die Programmierarbeit leichter, und die Codes laufen schneller.

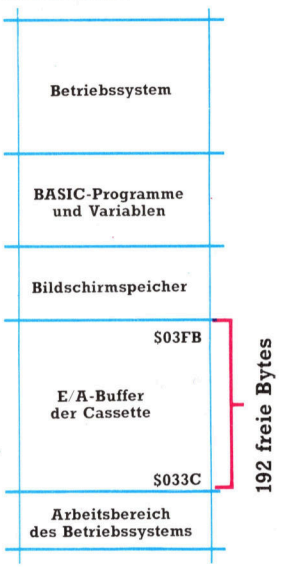
Spectrum

Direkte Eingabe des Maschinencodes:
LET RTOP=PEEK(23730)
+256*PEEK(23731)
LET RTOP=RTOP-N
PRINT "RTOP= ";RTOP
N ist die Anzahl der Bytes, die Sie für ihren Maschinencode reservieren wollen. Der reservierte Bereich beginnt bei 1+RTOP.



Commodore

Verwenden Sie den Cassettenbuffer von
$033C bis $03FB
zur Speicherung des Maschinencodes.



Achtung!
Diese Speicherzeiger müssen unmittelbar nach dem Anschalten der Maschine umgestellt werden. Es darf sich noch kein BASIC-Programm im Speicher befinden.

Assembler-übung

1) In dem nebenstehenden Kasten haben wir ein einfaches Programm im Assemblerformat abgedruckt. Assemblieren Sie das Programm in den Maschinencode und bestimmen Sie die Speicheradressen.
2) Welche Befehle fehlen in dem Programm?
3) Welche Auswirkungen hat das Programm auf die Register und die entsprechenden RAM-Bytes?
4) Was bedeutet der Begriff „Konstante"? Welche Arten von Daten gibt es noch?
5) Wenn BYTE1 des Programms als Adresse angesehen würde, welche Speicherseite würde das betreffen?
Die in den Programmen angegebenen Werte haben nur Übungscharakter und sind nicht für einen speziellen Computertyp gedacht.

Speicheradressen	Maschinencode	Assemblerformat		
		6502		
		START	EQU	$A000
		BYTE1	EQU	$45
		BYTE2	EQU	$38
			ORG	START
			LDA	#BYTE1
			CLC	
			ADC	#BYTE1
			STA	BYTE1
			ADC	#BYTE2
			STA	BYTE2
		Z80		
		START	EQU	$A000
		BYTE1	EQU	$45
		BYTE2	EQU	$38
			ORG	START
			LD	A,BYTE1
			AND	A
			ADC	A,BYTE1
			LD	(BYTE1),A
			ADC	A,BYTE2
			LD	(BYTE2),A

Die Auswirkungen von Maschinencodebefehlen

Befehl 1: Lade den Akkumulator mit dem Inhalt der Adresse 1.
Z80 : LD A, (ADDR1)
6502 : LDA ADDR1

Befehl 2: Addiere dazu den Inhalt der Adresse 2.
Z80 : ADC A, (ADDR2)
6502 : ADC ADDR2

Befehl 3: Speichere den Inhalt des Akkumulators in die Adresse 2.
Z80 : LD (ADDR2),A
6502 : STA ADDR2

Mit Ladeoperationen wie LDA ADDR1 oder LD(ADDR2),A wird immer der Inhalt der Quellenadresse in die Bestimmungsadresse kopiert. Der bisherige Inhalt der Bestimmungsadresse wird dabei überschrieben, während sich der Inhalt der Quellenadresse durch den Kopiervorgang nicht verändert.

Musik aus dem C64

Der Commodore Music Maker ist ein Programmpaket, das ein zwei Oktaven umfassendes Keyboard beinhaltet.

Bestimmte Möglichkeiten von Heimcomputern laden förmlich dazu ein, Peripherien zu entwickeln, die die gesamten Fähigkeiten der Rechner ausschöpfen. Das gilt besonders für die Sound- und Grafik-Möglichkeiten, für die eine Vielzahl ergänzender Produkte produziert worden ist. Eine Reihe von Musik-Softwareprogrammen wurde bereits für den Commodore 64 entwickelt. Doch was immer sie an Vorteilen brachten, die Melodien mußten stets auf der Tastatur gespielt werden und nicht auf einem klavierähnlichen Keyboard, mit dem Musiker vertraut sind. Mit der Einführung des Music Maker stellt Commodore nun ein Anwendungspaket zur Verfügung, das jetzt den unmittelbaren Zugriff zum ausgezeichneten SID (Sound Interface Device)-Chip des Rechners erlaubt.

Die Auflage, mit Klaviertasten ausgestattet und aus solidem Kunststoff hergestellt, wird auf das Gehäuse des 64er gesetzt. Diese eigentliche Tastatur des Music Maker umfaßt zwei Oktaven. Jede Taste ist einzeln an der Auflage befestigt. An ihrer Unterseite befindet sich ein zahnartiger Vorsprung. Drückt man nun eine Taste, greift dieser Vorsprung auf eine korrespondierende Taste der Computertastatur, die zur Erzeugung einer bestimmten Note programmiert ist.

Diese Methode ist einfach, funktioniert aber überraschend gut. Die solide Plastikauflage sitzt sehr fest. Und selbst wenn man schnell auf der Tastatur spielt, verfehlt man äußerst selten eine Note.

Die zum Music-Maker-Paket gehörige Software ist auf Diskette und Cassette erhältlich. Das Programm wird menügesteuert, wobei alle Funktionstasten genutzt werden. Damit ist es dem Anwender möglich, den Klang – oder die Zahl der Klänge – , der über Tastatur erzeugt wurde, zu modifizieren. Das geschieht entweder durch Veränderung der Wellenform oder der Tonhöhe. Zusätzlich gibt es als Begleitmöglichkeit die Percussion-Option sowie einen Baß-Rhythmus. Ein Sequenzer erlaubt schließlich Aufnahme und Wiedergabe gespielter Melodien. Programmierte Töne und Sequenzen können mittels SAVE oder LOAD von Diskette und Cassette geladen bzw. gespeichert werden.

Jede der acht Stimmen ist auf einen anderen Klang programmierbar. Nach Betätigen von F 6 (Stimm-Modifizierung) kann der Anwender bestimmen, welche Stimme geändert werden soll. Darauf erscheint auf dem Schirm eine Reihe von Optionen: Attack, Decay, Sustain und Release. Zur Festlegung dieser Parameter können Zahlen von 0 bis 15 eingegeben werden. Danach wird der Programmierer gefragt, ob er Filter benutzen will. (Damit lassen sich bestimmte Ober- und Unterfrequenzen bei Klängen entfernen.) Antwortet man mit „Ja", erfolgen weitere Fragen nach Frequenzbestimmung und Filterebene.

Die Möglichkeiten der drei Percussion-Rhythmen sind begrenzt, aber ausreichend. Die Baß-Rhythmen folgen demselben Muster wie die Percussion. Drei Optionen erlauben dem Programmierer, den Baß ein- oder auszuschalten und seine Höhe zu modifizieren. Ab-

Der Music Maker für den Commodore 64 besteht aus einer Auflage mit Klaviertastatur und dazugehöriger Software. Die zwei Oktaven umfassende Auflage paßt genau auf das Computergehäuse. Die Klaviertasten haben dann Verbindung mit den Computertasten, die so programmiert sind, daß entsprechende Noten erzeugt werden. Auf die Funktionstasten wirkt die Auflage nicht ein, da diese zur Klangprogrammierung benötigt werden. Mit der Software können qualitativ gute Klänge erzeugt werden.

Music Maker

PREIS

ca. 200 Mark

SOFTWARE

Die mitgelieferte Software ist auf Diskette und Cassette erhältlich.

DOKUMENTATION

Zum Music Maker gehören zwei Handbücher. Die Bedienungsanleitung gibt alle notwendigen Informationen. Das zweite Handbuch, „Start Playing Keyboard", beinhaltet eine Reihe von Noten populärer Melodien.

STÄRKEN

Der Music Maker ist sehr preiswert. Er verwandelt den Commodore 64 in ein vollwertiges Musikinstrument.

SCHWÄCHEN

Die Optionen sind limitiert, besonders auf vorprogrammierte Schlagzeug- und Baßrhythmen.

hängig vom gespielten Rhythmus, wird die Höhe der Baßlinie entweder um eine Fünftel- oder eine ganze Oktave gesenkt. Das Tempo kann man durch Druck auf die Cursorsteuerungstasten ändern: Pfeil nach unten verlangsamt das Tempo, nach rechts wird es beschleunigt.

Allerdings können viele Optionen nicht gemeinsam benutzt werden. Wenngleich Baß und Percussion gemeinsam spielbar sind und der Anwender auch eine Melodie via Keyboard darüberlegen kann, ist dies nur monophon möglich. Akkorde sind also nicht mit Hintergrundrhythmus spielbar.

Ausreichende Literatur

Das gilt auch für die Sequenzer-Option. Viele Pop-Gruppen verzichten völlig auf die traditionelle Baß/Rhythmus-Gruppe und benutzen statt dessen den Sequenzer für diesen Zweck. Beim Music Maker ist das nicht möglich. Diese Limitierung ist jedoch Hardware-bedingt. Da der SID-Chip nur über drei Stimmen verfügt, kann man gleichzeitiges Spielen von Baß, Rhythmus und polyphoner Melodie nicht erwarten. Selbst ein Glissando (eine Option, mittels der die Tonhöhe einer Note durch Drücken der entsprechenden Taste bei gleichzeitiger Betätigung der Leertaste erhöht wird) läßt sich im Rhythmusbereich nicht durchführen. Das mag aber daran liegen, daß der Klang digital erzeugt wird. Gleichzeitiges Spielen von Baß und Schlagzeug und dabei eine winzige Tonhöhenveränderung zur Erzielung eines Gleitens scheinen die Möglichkeiten des Prozessors zu überfordern.

Eine andere Schwierigkeit ergibt sich beim Spielen in „Echtzeit". Befindet sich der Anwender nämlich im „Play Mode", können die Parameter für Stimme und Oktave nicht verändert werden. Das bedeutet, daß der Spieler mit zwei Oktaven vorliebnehmen muß

Zum Lieferumfang des Music Maker gehören zwei Handbücher. Die eigentliche Bedienungsanleitung enthält Ladeanweisungen und eine kurze Erläuterung, wie die einzelnen Funktionen genutzt werden können. Für den absoluten Anfänger reicht das, obwohl es nicht sehr detailliert ist. Die Menü-Anweisungen reichen jedoch zur Nutzung der Software völlig aus. Das zweite Handbuch, „Start Playing Keyboard", umfaßt kurze Anweisungen zum Keyboardspiel und eine Erläuterung der Musiknotation. Darauf folgen Noten für 28 bekannte Musikstücke.

Der Commodore Music Maker ist ein interessanter Versuch, die Soundmöglichkeiten des Commodore 64 voll auszuschöpfen. Wer im Gebiet der Computermusik neu ist, bekommt ein einfach zu benutzendes Paket und kann, nach Gewöhnung an die Tastatur, leicht Melodien spielen. Fortgeschrittenere Anwender werden die Möglichkeiten für zu eingeengt halten, zumal das einfache Keyboard keine besonderen Effekte erlaubt. Wer aber als Anfänger mit dem Commodore C 64 die Möglichkeiten der Computermusik erforschen will, bekommt mit dem Music Maker ein preiswertes Instrument.

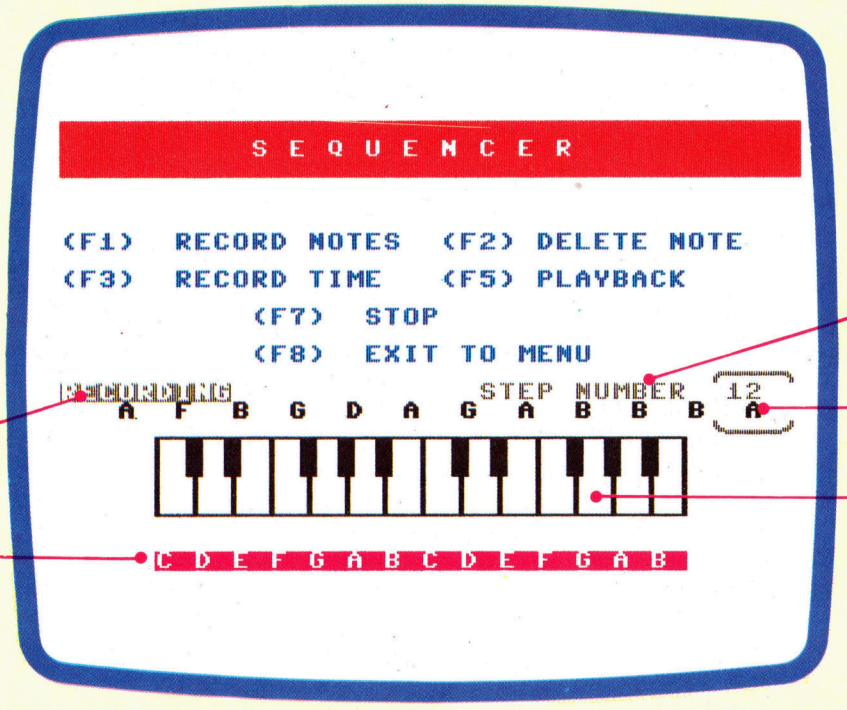
Musikalisches Menü

Dieser Bildschirm zeigt das Menü des Music Maker nach Wahl der Sequenzer-Option.

Das Menü führt die sechs zur Verfügung stehenden Optionen auf. Zu beachten ist, daß die gespielten Noten und ihre Dauer (TIME) separat eingegeben werden.

Das erinnert den Anwender daran, daß sich der Music Maker im Aufzeichnungsmodus befindet.

Diese Zeile zeigt dem Anwender die Namen der einzelnen Noten auf der Tastatur.



Die Schrittzahl zeigt an, wie viele Noten bisher in den Speicher des Sequenzers eingegeben worden sind.

Dies sind die bisher eingegebenen Noten.

Ein blauer Cursor auf der Tastatur zeigt, welche Note gerade gespielt wird.

Jedem seinen Einstein?

Der in England hergestellte Tatung Einstein bietet 80 KByte RAM und eine breite Palette von Schnittstellen. Die Maschine verfügt außerdem über ein umfassendes BASIC und gute Grafik- und Klangmöglichkeiten.

Der Preis zeigt, daß der Einstein für den „ernsthaften" Heimgebrauch gedacht ist. Obwohl das Gerät auch für Spiele eingesetzt werden kann, bietet es auf diesem Gebiet nur wenig Vorteile gegenüber anderen Heimcomputern, die für ein Viertel des Preises zu haben sind.

Das oberhalb der Tastatur eingebaute 3-Zoll-Diskettenlaufwerk bietet große Vorteile. Durch diese Integration stellt Tatung sicher, daß alle Programme von Anfang an für dieses Speichermedium geschrieben werden. Auf jeder Seite der 3-Zoll-Disketten lassen sich 190 KByte Daten speichern, der Einstein kann aber jeweils nur eine Seite nutzen. Ein zweites Laufwerk kann in das Gehäuse eingebaut werden. Zwei weitere Laufwerke lassen sich über eine Schnittstelle auf der Rückseite des Gerätes anschließen.

Zur Diskettensteuerung setzt der Einstein ein eigenes Betriebssystem (DOS) ein. Das System ist dem CP/M-Standard ähnlich, der auf vielen kommerziellen Maschinen läuft. Tatung hofft darauf, daß viele Softwarehäuser ihre CP/M-Programme dem Einstein anpassen werden. Betriebssystem und BASIC sind nicht wie bei anderen Maschinen im ROM untergebracht, sondern werden von der Diskette geladen. Daraus ergeben sich zwei Vorteile: Das BASIC wird nur dann geladen, wenn es benötigt wird. Sonst steht der Arbeitsspeicher für andere Programmiersprachen oder für Programme im Maschinencode vollständig zur Verfügung. Weiterhin läßt sich das Betriebssystem problemlos gegen neuere Versionen austauschen (es wird einfach eine andere Diskette eingesetzt). Zusätzlich liefert Tatung eine Diskette mit der Programmiersprache DR LOGO kostenlos mit.

Das LOGO-Handbuch muß jedoch separat gekauft werden. Nach dem Laden des BASIC verfügt der Einstein über einen Arbeitsspeicher von 43 KByte, – mehr als fast alle anderen Heimcomputer. Dieser große Arbeitsspeicher wird möglich, da der RAM-Bereich des Einstein weitere 16 K für die Bildschirmdarstellung enthält.

Das BASIC des Einstein ist eine Mischung aus Acorn-BASIC und Microsoft-Extended-BASIC. Befehle zur Reorganisation von BASIC-Zeilen und zur automatischen Zeilennumerierung erleichtern die Programmeingabe. Mit dem Bildschirmeditor lassen sich Änderungen problemlos auf dem Monitor ausführen. Die Grafikbefehle können Linien, Kreise und Ellipsen zeichnen und Umrisse mit Farbflächen füllen. Maximale Auflösung ist dabei 256 x 192 Pixel. Der Grafikchip kann von den 15 verfügbaren Farben allerdings nicht mehr als zwei pro Zeile (acht Pixel) einsetzen. Bis zu 32 Sprites, die sich über BASIC steuern lassen, ermöglichen die Programmierung von Spielen mit schnellen Bewegungsabläufen. Und im ROM befindet sich ein Monitorprogramm für die Pro-

Der Einstein von Tatung (früher Decca) ist für den ernsthaften Anwender gedacht. Der Computer wird mit einem Diskettenlaufwerk geliefert. Sein Gehäuse ist größer als die meisten anderen Heimcomputer und stabil genug, um einen Monitor zu tragen.

grammierung im Maschinencode.

Der Grafikchip unterstützt nur eine Bildschirmanzeige von 40 Zeichen pro Zeile. Die zusätzliche 80-Zeichen-Karte soll den Einsatz von CP/M-Programmen mit diesem Bildschirmformat möglich machen. Im 80-Zeichen-Format zeigt der Bildschirm eine monochrome Darstellung, eine Farbversion soll jedoch verfügbar sein.

Die Klangqualität des Einstein ist gut. Die Töne werden über einen großen Lautsprecher oberhalb der Tastatur ausgegeben. Die Lautstärke ist regelbar. Und es gibt viele BASIC-Befehle für die einfache Klangsteuerung.

QWERTY-Tastatur mit Funktionstasten

Das Gerät bietet acht programmierbare Funktionstasten; in einen klaren Plastikstreifen über den Tasten lassen sich Etiketten für die Beschriftung einstecken. Der Aufbau des Tastenfeldes ist gut, dadurch wird Blindschreiben möglich. Tatung bietet jedoch nur zwei Cursorsteuerungstasten statt der üblichen vier. Die Cursorsteuerung kann daher nur zusammen mit der Shift-Taste erfolgen, was auf einer Maschine mit diesem Preis allerdings lästig ist. Mit der Grafiktaste läßt sich auch eine Anzahl Grafikzeichen erzeugen, diese Funktion ist aber nur begrenzt einsetzbar.

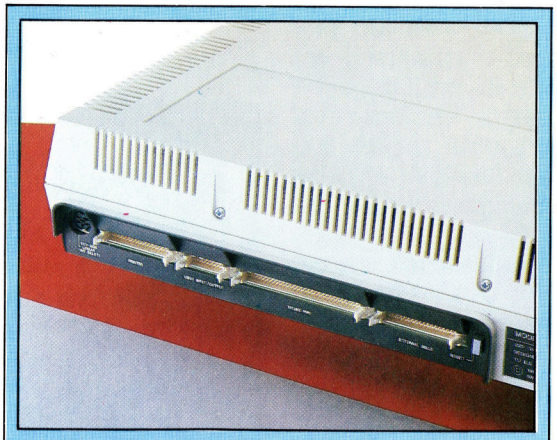
Mit Schnittstellen ist der Einstein bestens ausgerüstet. Enthalten ist eine Standard-Centronics-Druckerschnittstelle, eine RS232-Buchse für den Anschluß von weiteren Drukkern, Modems und anderen Zusatzgeräten, eine RGB-Monitorbuchse, ein Fernseherausgang und zwei Joystickbuchsen. Da diese an einen Analog/Digital-Wandler angeschlossen sind, lassen sich die Joystickausgänge auch für Aufgaben einsetzen, bei denen analoge Stromwerte digitalisiert werden müssen. Die vier A/D-Kanäle dieser beiden Buchsen werden durch einen 8-Bit-Ausgang ergänzt, der digitale Signale an elektronische Geräte senden oder von dort empfangen kann. Durch diese Kombination von Benutzerausgang und A/D-Buchsen eignet sich der Einstein für den Anschluß wissenschaftlicher Geräte.

Speichererweiterungen werden durch die „Pipe" möglich (ein ähnliches Konzept wie die „Tube" des Acorn B), über die eine Vielzahl von Zusatzgeräten angeschlossen werden kann. Durch einen Steckplatz im Innern des Gerätes lassen sich die acht KByte ROM auf 32 KByte ausbauen.

Zweifellos bietet der Einstein gute Leistungen für seinen Preis. Es stellt sich allerdings die Frage, wieviele Käufer bereit sein werden, für einen Heimcomputer relativ viel Geld anzulegen, auf dem im Augenblick nur wenig Software läuft. Diese Situation wird sich nur ändern, wenn die Maschine in größeren Stückzahlen verkauft wird.

Farbmonitor
Der Farbmonitor von Tatung kann mit einem RGB- oder einem YUV-Signal arbeiten. Das letztere wurde von Tatung selbst entwickelt und soll dem RGB-System überlegen sein.

Diskettenlaufwerk von Hitachi
Als Diskettenstation wurde das 3-Zoll-System von Hitachi verwandt, das sich auf dem Markt für Microcomputer mehr und mehr durchsetzt. Die Disketten können bis zu 190 KByte speichern. Es kann nur auf eine Seite zur Zeit zugegriffen werden.



Die Schnittstellen des Einstein
Der Einstein verfügt über eine ganze Palette von Schnittstellen, darunter die „Pipe", die der „Tube" des Acorn B ähnlich ist.

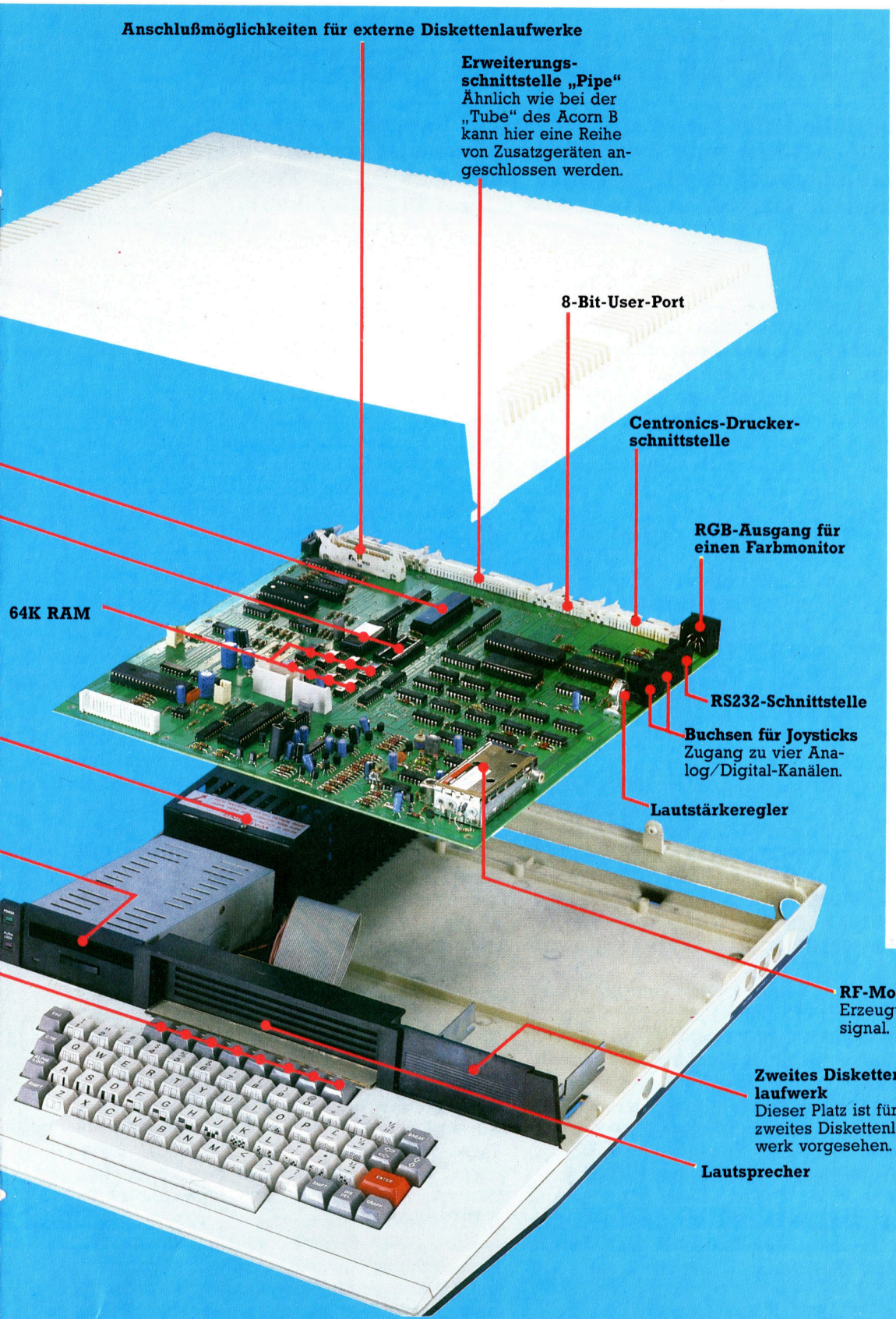
Microprozessor Z80

8K-ROM
Dieser Chip enthält das Monitorprogramm im Maschinencode. Der angrenzende Steckplatz ist für zukünftige Erweiterungen vorgesehen.

„Switch Mode"-Netzteil
Dieses „Switch Mode"-Netzteil setzt keinen normalen Transformator ein. Es ist kleiner und erzeugt weniger Hitze.

Diskettenlaufwerk
3-Zoll-Laufwerk im Hitachi-Format, Kapazität pro Seite: 190K.

Funktionstasten
Vom Anwender frei programmierbar.



Tatung Einstein

ABMESSUNGEN

510 x 430 x 105 mm

ZENTRALEINHEIT

Z80

SPEICHERKAPAZITÄT

80K-RAM, einschließlich 16K-Bildschirmspeicher; 8K-ROM, erweiterbar auf 32K.

BILDSCHIRM-DARSTELLUNG

Text: 24 Zeilen mit je 32 Zeichen, 24 Zeilen mit je 40 Zeichen, 80-Zeichen-Darstellung durch Zusatzkarte; Grafik: 256 x 192 Pixel, 15 Farben.

SCHNITTSTELLEN

Centronics, RS232, zwei Joystickbuchsen (A/D-Wandler), RGB, „Pipe“, User Port, Anschlußmöglichkeit für externe Diskettenlaufwerke.

PROGRAMMIER-SPRACHEN

BASIC und DR LOGO werden mitgeliefert; PASCAL, FORTH, FORTRAN, COBOL und C sind verfügbar.

TASTATUR

67 Schreibmaschinentasten im QWERTY-Format, darunter acht programmierbare Funktionstasten.

STÄRKEN

Integriertes Diskettenlaufwerk, viele Schnittstellen, großer Arbeitsspeicher, gutes BASIC, Sprite-Grafiken.

In Reih und Glied

Sequentielle (oder serielle) Dateien sind Überbleibsel der Zeit, da in der Datenverarbeitung noch Bänder eingesetzt wurden. In diesem Artikel untersuchen wir, wie man von einem BASIC-Programm aus sequentielle Dateien aufbauen und darauf zugreifen kann.

Die binäre (oder Programm-)Datei ist eine vereinfachte Form der sequentiellen Datei. Bei der Eingabe von SAVE"Programmname" führt das Betriebssystem folgende Abläufe aus: Es stellt zunächst die Kommunikation mit dem Band- oder Diskettenlaufwerk her und speichert dann einen Vorspann mit Dateinamen und Informationen über den Dateiinhalt darauf ab. Dahinter wird der Inhalt der RAM-Blöcke mit dem zu speichernden Programm geschrieben. Nach Beendigung dieses Vorgangs setzt das Betriebssystem eine Endmarkierung und schließt die Kommunikation zwischen Computer und Laufwerk ab.

Die BASIC-Befehle für den Aufbau sequentieller Dateien sind von Computer zu Computer verschieden. Hier als Beispiel die Version des Commodore 64:

```
100 R$="DIES IST EIN DATENSATZ"
  150 OPEN 5,1,2, "DATAFILE,SEQ,WRITE"
  200 PRINT#5,R$
  250 CLOSE 5
```

Der erste Befehl OPEN 5,1,2, veranlaßt das Betriebssystem, den Kommunikationskanal 5 zur Peripherie (Device 1: Cassettengerät) zu öffnen. Da sequentielle Dateien auf mehreren Peripheriegeräten eröffnet werden können und auch der gleichzeitige Zugriff auf mehrere Dateien möglich ist, müssen der Kanal und das Peripheriegerät (englisch: device) angegeben werden.

Hinter dem Befehl OPEN steht der Dateiname. Auf dem Commodore besteht diese Angabe beispielsweise aus dem Namen „DATAFILE", gefolgt von der Dateiart (SEQ bedeutet sequentielle Datei) und der Zugriffsmethode (WRITE eröffnet die Datei zum Schreiben). Sequentielle Dateien können nur entweder zum Schreiben oder zum Lesen eröffnet werden, niemals für beides gleichzeitig. WRITE schaltet dabei den Schreib-Lesekopf ein und schreibt den Dateinamen und Systeminformationen in einen „Vorspann", der auch den Dateianfang markiert.

Der Befehl PRINT#5 in Zeile 200 schreibt die Daten in die Datei. PRINT führt auch hier seine normale Funktion aus, nur zeigt das „#"-Zeichen an, daß die Daten nicht zum Bildschirm, sondern zu dem spezifizierten Ausgabekanal gesendet werden. Der Inhalt von R$ wird daher über Kanal 5 an die sequentielle Datei „DATAFILE" weitergegeben, die dann als Datensatz den String „DIES IST EIN DATENSATZ" enthält. Der letzte Befehl, CLOSE 5, schreibt die Endmarkierung in die Datei und schließt die Kommunikation auf Kanal 5 ab.

Mit den folgenden Programmzeilen läßt sich die gespeicherte Information lesen:

```
500 OPEN 3,1,2, "DATAFILE,SEQ,READ"
  550 INPUT#3,A$
  600 CLOSE 3
  650 PRINT A$
```

Beide Programmteile werden mit dem Befehl OPEN eröffnet. OPEN bedeutet hier: „Finde den Anfang der Datei DATAFILE und bereite den Lesevorgang vor", bei dem ersten Programm jedoch: „Eröffne eine Datei namens DATAFILE und bereite den Schreibvorgang vor". Obwohl sich die Kanalnummern beider Sequenzen voneinander unterscheiden, hat dies keinen Einfluß auf den Vorgang. Die Kennung des Peripheriegerätes muß in beiden Fällen jedoch identisch sein, da die Datei mit dem Cassettengerät gespeichert wurde, dem die Kennung Device 1 zugeordnet ist. Name und Art der Datei sind identisch, da beide die Datei eindeutig kennzeichnen. Lediglich die Zugriffsart ist verschieden – READ statt WRITE.

Der nächste Befehl lautet INPUT#3,A$ und bedeutet „Eingabe über Kanal 3" statt über die Tastatur. Der erste vollständige Datensatz der Datei wird gelesen und über Kanal 3 in die Variable A$ geladen – genau der umgekehrte Vorgang wie beim Speichern der Variablen mit PRINT#5,R$.

Die beiden Beispiele machen einige Vor- und Nachteile von sequentiellen Dateien deutlich: Zwar lassen sich beliebig viele Variablen unabhängig von Strukturen oder Dateigrößen speichern, die Dateiinhalte können jedoch normalerweise nur beim Anfang beginnend Datensatz für Datensatz gelesen werden. Auch gibt es bei diesem Vorgang keine Möglichkeit, bestimmte Datensätze zu überspringen, zu löschen, einzufügen oder nochmals zu lesen.

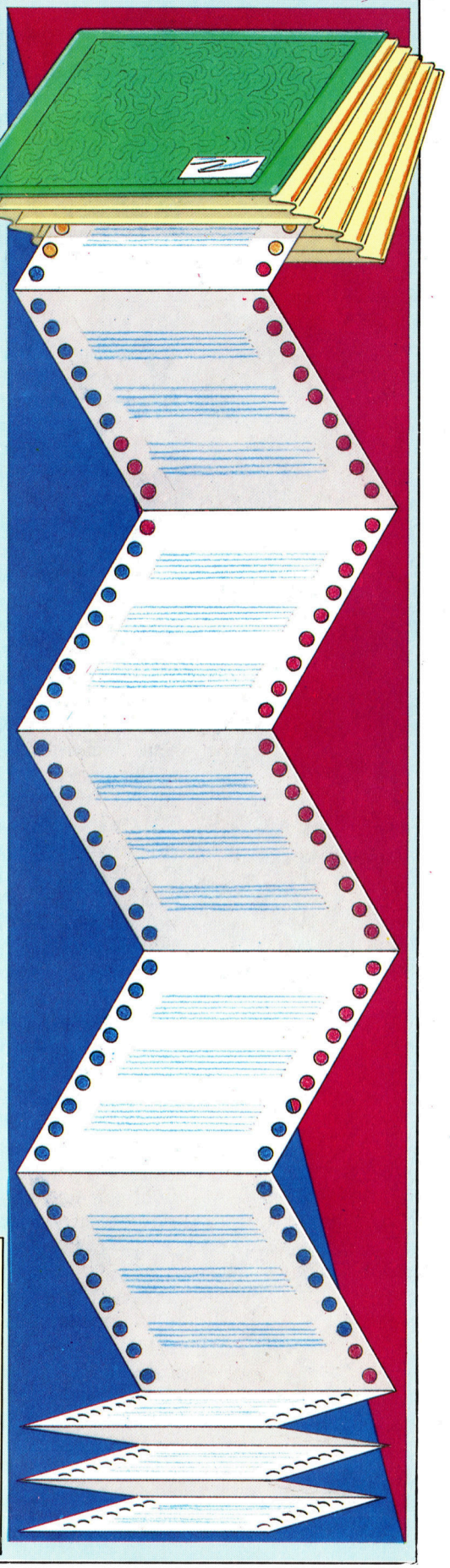
Sequentielle Dateien wurden ursprünglich für die Bandspeicherung entwickelt. Sie lassen sich nur durch die Zusammenstellung einer neuen Version sortieren oder editieren. Diese neue Version wird dann an einer anderen Stelle des Bandes gespeichert. Ein ähnlicher Vorgang spielt sich beim Editieren von Musikcassetten ab, die entweder neu aufgenommen werden müssen oder bei denen einzelne Musikstücke mit dem Band herausgeschnitten werden.

Dateiverwaltung

```
199 REM++++++++++CBM C64++++++++++
200 REM+       WRITE      FILES      +
201 REM++++++++++CBM C64++++++++++
220 GOSUB 1500
240 FOR K=65 TO 90
260 Z$=CHR$(K)+X$:C$=D$+"WRITING "+CHR$(K)
280 GOSUB 2000
300 NEXT K
399 REM++++++++++++++++++++++++++++
400 REM+        READ     FILES       +
401 REM++++++++++++++++++++++++++++
420 FOR L=0 TO 1 STEP 0:FOR M=1 TO 1
440 PRINT D$;"ACCESS TO RECORDS"
460 INPUT"SEARCH STRING ( *=QUIT )";N$
480 L$=LEFT$(N$,1):IF L$="*" THEN GOTO 5000
500 IF L$<"A" OR L$>"Z" THEN M=0
520 NEXT M
540 Z$=L$+Y$
560 GOSUB 3000
580 PRINT TAB(5)"RECORD ";N;"(HIT ANY KEY)"
600 GET GT$:IF GT$="" THEN 600
620 NEXT L
999 END
1499 REM*****************************
1500 REM*****INITIALISE  S/R********
1501 REM*****************************
1520 D$=CHR$(147):PRINT D$,CHR$(8);CHR$(142)
1540 X$=",S,W":Y$=",S,R"
1600 RETURN
1999 REM*****************************
2000 REM*****WRITE A FILE S/R*******
2001 REM*****************************
2020 PRINT C$:INPUT"HOW MANY RECORDS";R
2040 OPEN 8,8,2,Z$
2060 IF R=0 THEN PRINT#8,"*":CLOSE8:RETURN
2080 FOR I=1 TO R
2100 PRINT C$:PRINT "RECORD #";I
2120 INPUT "TEXT.....";R$
2140 PRINT#8,R$
2160 NEXT I
2180 PRINT#8,"*":CLOSE8
2499 RETURN
2999 REM*****************************
3000 REM*****READ  A FILE S/R*******
3001 REM*****************************
3020 PRINT D$;"SEARCHING ";L$;" FOR ";N$
3040 OPEN 8,8,2,Z$
3060 FOR I=1 TO 100000
3080 INPUT#8,R$
3100 PRINT R$
3120 IF R$="*" THEN N=0:I=100000
3140 IF R$=N$ THEN N=I:I=100000
3160 NEXT I:CLOSE8:N$=""
3180 RETURN
5000 REM*****CLOSE    PROGRAM*******
5020 PRINT CHR$(9);"END OF PROGRAM":STOP
```

220: Initialisieren.
260–280: Dateien „A“ bis „Z“ einrichten.
420–540: Eingabe des zu suchenden Datensatzes, Anfangsbuchstaben feststellen, korrekte Datei identifizieren.
560: Datei durchsuchen.
580–600: Suchergebnis anzeigen.
1520: Bildschirm löschen, Großbuchstaben einschalten.
2040: Datei zum Schreiben eröffnen.
2060: Auf leere Datei überprüfen, „*“ schreiben, Datei schließen.
2120–2140: Test des Datensatzes eingeben und in der Datei speichern.
2180: „*“ als letzten Datensatz schreiben, Datei schließen.
3040: Datei zum Lesen eröffnen.
3120: Auf letzten Datensatz überprüfen, Suche beenden.
3140: Überprüfen, ob Datensatz gefunden, Suche beenden.
3160: Datei schließen.

Dieses Programm zeigt, wie mit sequentiellen Dateien ein einfaches alphabetisches Notizbuch aufgebaut werden kann. Das „Buch“ besteht aus 26 Dateien – je eine pro Buchstabe. In jede Datei können Sie Daten eingeben, die mit dem entsprechenden Buchstaben anfangen. Dateien dürfen aber auch leer sein. Mit dem Programm kann nach beliebigen Datensätzen gesucht werden. Dabei wird die entsprechende Datei so lange durchsucht, bis der Datensatz gefunden ist. Ist die Suche ergebnislos, erscheint die Nachricht "RECORD 0". Über das Betriebssystem erhält das Programm direkten Zugriff auf die zu durchsuchende Datei.



BASIC-Dialekte

Acorn B
Zeile 260 und 540 wie beim Spectrum verändern. PRINT#8, INPUT#8 und CLOSE8 durch PRINT#C8, INPUT#C8 und CLOSE#C8 ersetzen.

```
600 GT$=GET$
1520 *DISK
1530 MODE 7
1540 D$=CHR$(12):PRINT D$"USE UPPER CASE"
1550 PRINT"--HIT ANY KEY--":GT$=GET$
2040 C8=OPENOUT(Z$)
3040 C8=OPENIN(Z$)
5020 PRINT"END OF PROGRAM"
```

Spectrum Microdrive
LET einsetzen wo nötig.
PRINT D$;.. durch CLS PRINT.. ersetzen.
PRINT C$ durch CLS:PRINT:C$ ersetzen.
OPEN 8,8,2,Z$ durch OPEN #8;"m";1;Z$ ersetzen.
CLOSE8 durch CLOSE#8 ersetzen.
Zeile 1540 löschen.

```
260 Z$=CHR$(K):LET C$="WRITING"+Z$
480 LET L$=N$(1):IF L$="*" THEN GOTO 5000
540 LET Z$=L$
600 PAUSE 0
1520 CLS:LET F2=PEEK 23658:POKE 23658,8
3080 INPUT #8;R$
5020 POKE 23658,F2:PRINT "END OF PROGRAM":STOP
```

Funktionen und Kontroll-Strukturen

In diesem Artikel widmen wir uns den Funktionen VAL, GOSUB und GOTO sowie den Kontroll-Strukturen WHILE . . . WEND und REPEAT . . . UNTIL.

Vielleicht haben Sie bereits bemerkt, daß einige Funktionen beim Sinclair-BASIC keine Klammern für das Argument benötigen, wie es bei vielen anderen BASIC-Versionen der Fall ist. Somit kann LEN(X$) sowohl in der Form LEN X$ als auch LEN (X$) geschrieben werden.

Die Funktion CODE ist das Sinclair-Äquivalent von ASC() und funktioniert in exakt derselben Art und Weise. Der Zeichensatz des Sinclair dagegen entspricht nur im Bereich der Werte von 32 bis 122 dem Standard-ASCII-Zeichensatz. Wenn Sie also beispielsweise PRINT CHR$(7) eingeben, was bei den meisten anderen BASIC-Versionen einen Kontrollton erklingen läßt, erhalten Sie beim Sinclair eine Fehlermeldung.

Die Funktion VAL entspricht dem normalen BASIC, doch würde eine Anweisung wie VAL("a45") beim Sinclair-BASIC zu einem Programmabsturz führen, da das Argument der Funktion nicht numerisch ist. Bei den meisten anderen BASIC-Dialekten erhielte man als Ergebnis den Wert Null. Wenn diese Feinheit zu einem Problem wird, müssen Sie eventuell eine Unterroutine schreiben, um die VAL-Funktion zu ersetzen. Sie können aber auch vor Ausführung der Funktion zuerst mit CODE den Wert des ersten Zeichens der VAL-Funktion überprüfen. Ergibt CODE A$(1) einen Wert kleiner als 48 oder größer als 57, dann ist A$ nicht numerisch und kann nicht als Argument der VAL-Funktion verwendet werden.

Die VAL-Funktion des Sinclair kann aber auch zur Berechnung numerischer Ausdrücke eingesetzt werden.

```
LET A$="6*12":PRINT VAL A$
```

Diese Anweisungen ermitteln das Ergebnis 72, also den Wert des Ausdruckes "6*12". Die Berechnung von Ausdrücken läßt sich auf verschiedene Art und Weise nutzen. Ein einfaches Beispiel ist das folgende Programm zum Zeichnen eines Block-Diagramms:

```
100 DIM S$(31)
200 LET S$="*******************************"
300 INPUT "GEBEN SIE EINE FUNKTION VON
    X EIN ";F$
400 PRINT "Y= ";F$:PRINT
500 FOR X=1 TO 10
600 PRINT S$(TO INT(VAL F$))
700 NEXT X
800 PRINT "====================
    ============="
900 PRINT
    "00000000011111111122222222223"
950 PRINT
    "12345678901234567890123456789O1"
```

Wenn Sie dieses Programm starten, können Sie jeden beliebigen algebraischen Ausdruck mit der Variablen X (beispielsweise 2*X+3/X) eingeben, worauf Sie als Ergebnis ein Blockdiagramm dieser Funktion auf dem Bildschirm sehen können. Die Grafikpunkte haben jeweils die Größe eines Pixels.

Auch GOSUB und GOTO sind in der Lage, mathematische Ausdrücke zu berechnen, wogegen die meisten anderen BASIC-Versionen eine gültige Zeilennummer als Argument benötigen. Durch diese Tatsache ergeben sich einige Vorteile. Sie können Ihren Unterroutinen beispielsweise Namen geben, Variablen mit denselben Namen und entsprechenden Werten definieren, und dann die Unterroutinen mit ihren Namen aufrufen.

```
100 LET ANFANG=1000
200 LET ENDE=2000
300 LET BERECHNUNG=3000
400 GOSUB ANFANG
500 GOSUB BERECHNUNG
600 GOSUB ENDE
700 STOP
1000 REM*** ANFANG UNTERROUTINE ***
. . .
2000 REM*** ENDE UNTERROUTINE ***
. . .
3000 REM*** BERECHNUNG UNTER-
     ROUTINE ***
```

Mit einer solchen Programmiermethode erklärt sich der Ablauf nahezu von selbst. Wenn Sie GOSUB ANFANG durch GOSUB (VAR1+N*VAR2) oder einen anderen gültigen numerischen Ausdruck ersetzen, wird der Ausdruck berechnet, und das Ergebnis wird wie eine normale Zeilennummer gehandhabt. Ein weiterer Vorteil des Sinclair-BASIC ist, daß das Programm zur nächsten gültigen Zeilennummer verzweigt, wenn das Argument einer GOTO- oder GOSUB-Anweisung auf eine Zeilennummer verweist, die nicht existiert.

Das Spectrum-BASIC beinhaltet nicht die Kontroll-Strukturen REPEAT...UNTIL und WHILE...WEND. Diese Strukturen können jedoch auf verschiedene Art und Weise simuliert werden. Die REPEAT-Struktur ist eine Schleife, die bei dem Wort REPEAT beginnt und bei dem Wort UNTIL endet, gefolgt von einer weiteren bedingten Anweisung. Wenn die Bedingung erfüllt ist, bricht die Schleife ab, und die Programmkontrolle wird an die folgende Anweisung weitergegeben. Beim Spectrum kann dieses wie folgt simuliert werden:

```
100 DATA "A","B","C","D","E","*","F",
    "G","*"
200 FOR L=1 TO 1
300 READ X$
400 PRINT X$
500 IF X$ <> "*" THEN LET L=0
550 NEXT L
600 PRINT "ENDE DER DATEN"
```

Das Problem bei der REPEAT... UNTIL-Struktur ist, daß die Schleife mindestens einmal durchlaufen wird, da die Ausgangsbedingung erst am Ende der Schleife überprüft wird. Um das zu verhindern, können Sie die WHILE...WEND-Struktur verwenden.

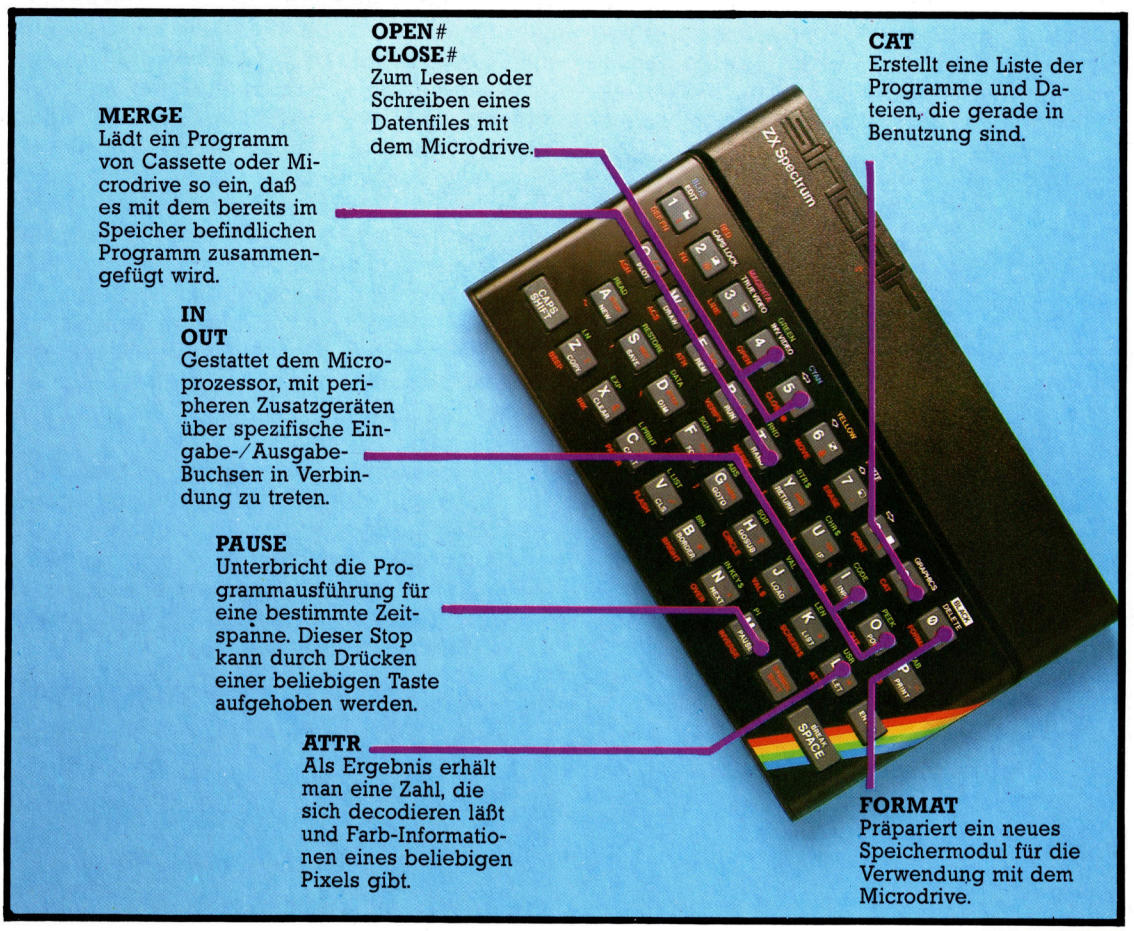
Diese Struktur ist eine Schleife, die bei der WHILE-Anweisung (der eine Kondition folgt) beginnt und bei der W(HILE)END-Anweisung endet. Solange die Bedingung nicht erfüllt wird, werden die Anweisungen zwischen WHILE und WEND immer wieder ausgeführt. Sobald die Bedingung wahr ist, wird der Programmlauf mit der Anweisung nach WEND fortgesetzt (die Anweisungen zwischen WHILE und WEND werden dann übersprungen). Ein entsprechendes Programm müßte so aussehen:

```
100 DATA "A","B","C","D","E","*","F",
    "G","*"
200 WHILE X$ <> "*" DO
300 READ X$
400 PRINT X$
500 WEND
600 PRINT "ENDE DER DATEN"
```

Beim Sinclair könnte dies durch eine IF...THEN...GOTO-Struktur ersetzt werden. Eine Alternative wäre jedoch:

```
 50 LET X$=""
100 DATA "A","B","C","D","E","*","F",
    "G","*"
200 LET TEST=(X$ <> "*")
250 FOR L=1 TO TEST
300 READ X$
400 PRINT X$
500 LET L=(X$="*")
550 NEXT L
600 PRINT "ENDE DER DATEN"
```

Wir haben uns jetzt mit den Hauptunterschieden des Sinclair-BASIC beschäftigt. Es gibt sicherlich noch einige kleinere Abweichungen, doch mit etwas Aufmerksamkeit und dem Handbuch können Sie diese allein bewältigen.



Geisterjäger

In den USA wie in Europa wurde der Film „Ghostbusters“ zu einem Kassenschlager. Der Software-Hersteller Activision, der das gleichnamige Computerspiel entwickelte, erwartet zweifellos einen ebenso großen Software-Verkaufserfolg.

Das Programm wurde in Zusammenarbeit mit der Filmgesellschaft Columbia Pictures entwickelt. Diese Kooperation ermöglichte den Programmierern, exakt nach Drehbuch zu arbeiten, ohne sich darum sorgen zu müssen, daß es Probleme mit den Urheberrechts-Inhabern geben könnte. Bestandteil des Programms ist auch der Titelsong des Films, der im Sommer 1984 zum Single-Plattenhit avancierte.

„Ghostbusters“ ist die Geschichte dreier Universitätsprofessoren, die in New York eine Geisterjäger-Agentur gründen und feststellen, daß in einem Apartmenthaus Poltergeister umgehen, um das Ende der Welt vorzubereiten. Das Spiel beginnt damit, daß der Spieler Geld bekommt, um ein Auto und eine „Geisterjäger“-Ausrüstung kaufen zu können. Dazu gehören ein „PK-Energie-Detektor“, der vor nahenden Geistern warnt, ein „Geisterstaubsauger“, mit dem die Geisterwesen aufgesaugt werden können, und ein tragbares Laserfangsystem, mit dem man Geister in eine Spezialfalle treiben kann. Nach Ausstattung mit diesen Waffen begibt sich der Spieler auf die Straßen New Yorks, um seinen Lebensunterhalt mühsam mit dem Fangen von Geistern zu verdienen.

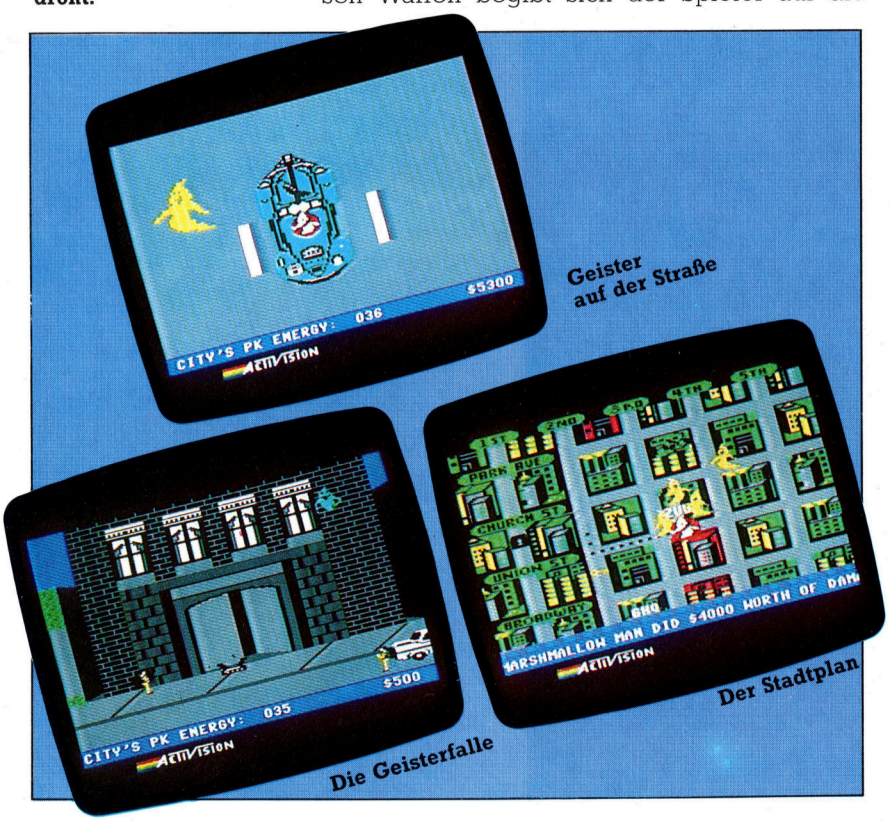
Der Bildschirm zeigt zunächst einen Stadtplan. Im Mittelpunkt der Karte ist das Gebäude zu sehen, in dem sich die Geister, angezogen vom bösen Zuul, zusammenfinden. Es spukt in verschiedenen Stadtteilen, und der Spieler muß sein Auto zu den jeweiligen Spukorten fahren. Während der Fahrt durch die Stadt tauchen „Roamer“ genannte Geister auf dem Bildschirm auf. Ist das Fahrzeug des Spielers mit einem Geisterstaubsauger ausgestattet, bringt der Fang der Roamer Extrapunkte.

Befindet sich der Geist bei Ankunft des Fahrzeuges noch vor Ort, muß die Geisterfangausrüstung aufgebaut werden. Zunächst stellt man eine Falle in die Bildschirmmitte. Dann wird der Geist mit Laserstrahlenhilfe hineingetrieben. War die Fangaktion erfolgreich, wird die Ghostbuster-Titelmelodie unterbrochen und man wird von einer Stimme mit dem Ausruf „Ghostbuster!“ beglückwünscht.

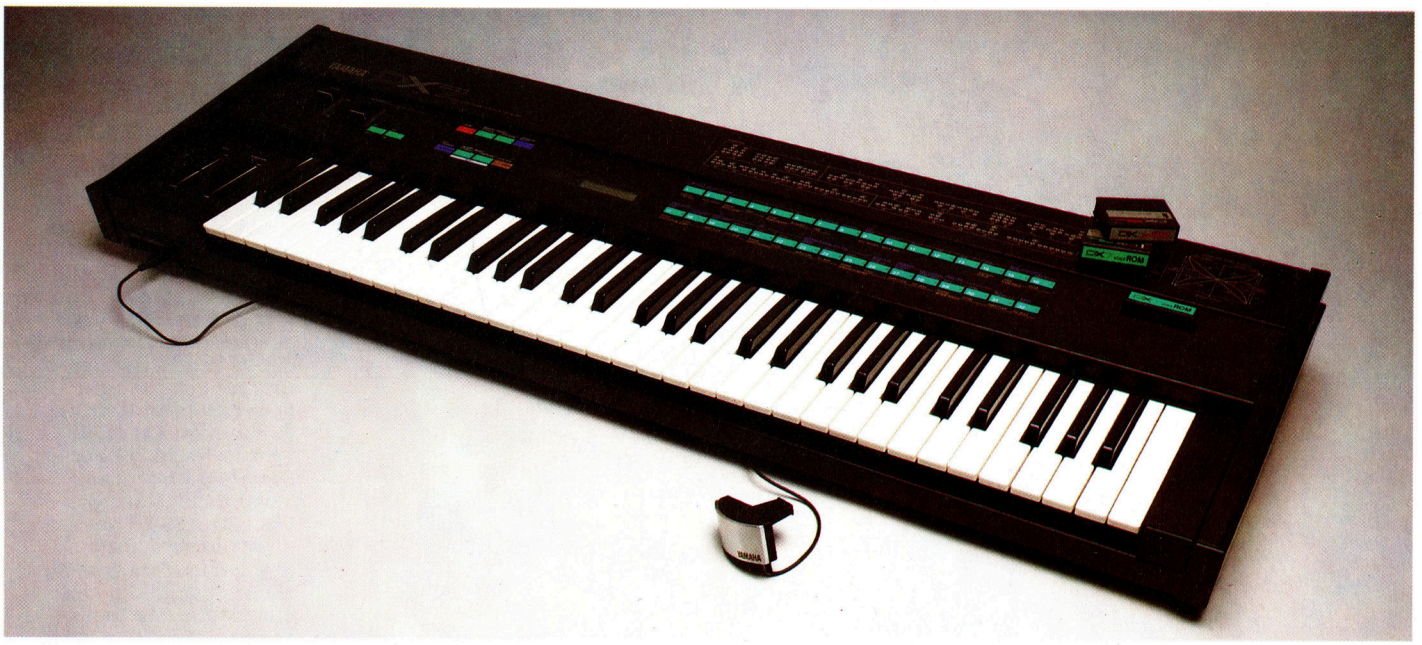
Während sich all dies ereignet, muß der Spieler stets vor dem „Marshmallow“-Alarm auf der Hut sein. Das bedeutet: Die Roamer vereinen sich und bilden den „Marshmallow Man“, der die Stadt zu zerstören droht, falls es nicht gelingt, sofort eine Geisterfalle aufzustellen. Zugleich hat der Spieler darauf zu achten, daß es dem Torhüter und dem Schlüsselmeister nicht gelingt, ihre Kräfte im Tempel des Zuul zu vereinen. Geschieht dies, bevor der Spieler genug Punkte gesammelt hat, um die Geisterjäger in den Tempel zu bringen, ist alles verloren.

Ghostbusters ist ein sehr gut programmiertes Spiel, mit detaillierter Grafik und ungewöhnlichem Sound. Als Dreingabe ist die Sprachausgabe zu bewerten. Das Spiel ist eine echte Herausforderung, und selbst erfahrenen Spielern beschert es langes Spielvergnügen. Eben eine richtige Geisterjagd!

Hier zeigen wir drei Bildschirm-Fotos des Activision-Spiels „Ghostbusters“. Während der Fahrt zu einem Spukhaus begegnen dem Spieler Geister auf der Straße, die man – entsprechende Fangvorrichtung vorausgesetzt – aufsaugen kann. Ist das Spukhaus erreicht, lassen sich die Geister mit Laserstrahlen in die Falle treiben. Der Stadtplan zeigt die Lage des Tempels von Zuul und weist zugleich die derzeitige Position des Marshmallow Man aus, der herumläuft und dabei Teile der Stadt bedroht.



Ghostbusters: Für Commodore 64, ZX Spectrum (48 K), Atari, Schneider CPL und MSX-Computer
Hersteller: Activision
Vertrieb: AriolaSoft Rushware
Autoren: David Crane, Adam Bellin, Hilary Mills
Joysticks: Erforderlich
Programm: Cassette oder Diskette



Synthi-Klänge

Von allen verfügbaren Microcomputer-gestützten Musiksystemen bietet MIDI die meisten Möglichkeiten. Musiker nutzen das Interface zur Verbesserung ihrer Life- und Studiodarbietungen.

Einer der neueren Synthesizer aus der DX-Reihe von Yamaha, der DX7, verfügt über eine Möglichkeit der Klangsynthese – als FM-Synthese bezeichnet –, die bisher ausschließlich sehr teuren Maschinen vorbehalten war. Statt vorhandene Töne aufzunehmen und diese durch Filter zu modifizieren bzw. durch Hüllkurvenkontrolle zu verändern, erzeugt der DX7 durch Kombination von sechs Wellenformen völlig eigene Klänge. Die mit dem DX7 erzeugten Klänge sind denen akustischer Instrumente ähnlicher als die herkömmlicher Synthesizer. Mit dem DX7 ist auch eine „Atemsteuerung" möglich. Ein Musiker kann also in einen Empfänger blasen und Saxophon- oder Trompetenklänge realistisch nachahmen. Dafür stehen ROMs mit gespeicherten Soundcharakteristika zur Verfügung. Außerdem kann man selbst Sounds erzeugen und diese auf RAMs speichern. Das Instrument kostet etwa 6000 Mark.

Bei der musikalischen Ausbildung liegen die Vorteile von MIDI auf der Hand, sowohl in der Schule als auch zu Hause. Denn das Lesen von Notenblättern ist schwierig, selbst wenn der Lernende bereits ein guter Spieler ist. Auch die Erarbeitung der Grundlagen wie Betonung, Punktierung und Taktmaß ist mühsam und zeitintensiv. Und es fällt recht schwer, eine einleuchtende Verbindung zwischen den Zeichen auf dem Papier und dem tatsächlich Gehörten herzustellen.

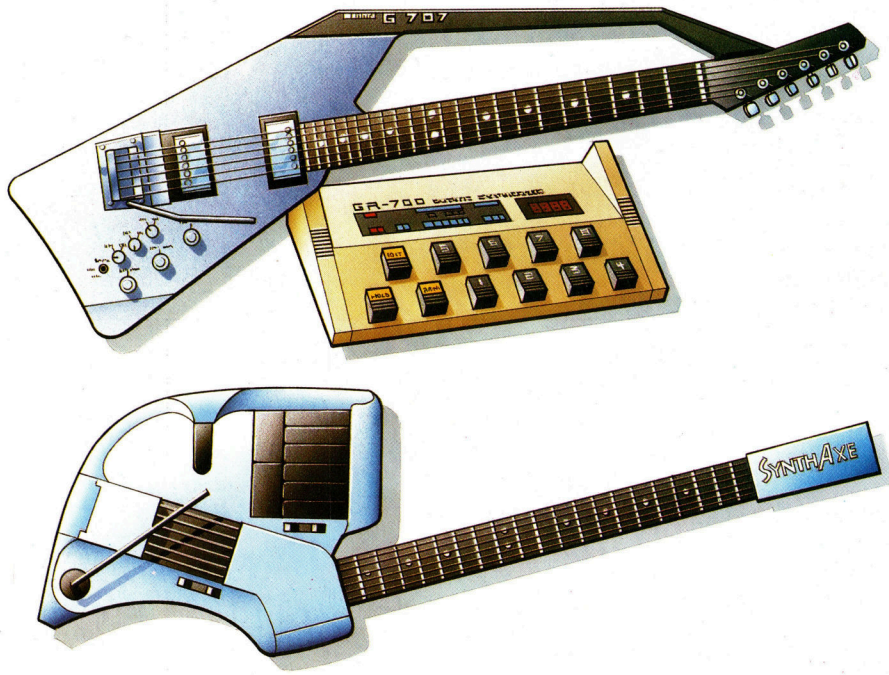
Eines der Kernprobleme liegt im Wesen der Musik: Damit sie einen Sinn gibt, müssen über einen bestimmten Zeitraum eine Reihe von Klangereignissen erfolgen. Hat der Anfänger nur dadurch die Möglichkeit, die visuelle Notation des betreffenden Stückes zu verfolgen, daß er den Ablauf unterbricht, werden alle Angaben für die Klangdauer sinnlos.

Mittels MIDI werden diese Hindernisse beseitigt. Es ermöglicht die Speicherung eines auf Synthesizer gespielten Stückes im Computer und, entsprechende Software vorausgesetzt, eine grafische Wiedergabe der gespielten Musik. Für den Musikstudenten ist dies eine unschätzbare Hilfe, da das bedeutet: Wird ein mittleres C auf dem Synthesizer-Keyboard gespielt, ist auch ein mittleres C auf den fünf Linien der Bildschirmdarstellung zu sehen. Wenn ein B-moll-Akkord für eine bestimmte Zeit gehalten wird, werden die harmonischen Komponenten (B, D und F) und zugleich die entsprechende Tondauer dargestellt.

Diese Idee läßt sich durch Software weiterentwickeln, die vorprogrammierte Musikstücke enthält: Der angeschlossene Synthesizer spielt die Musik, und gleichzeitig erfolgt eine komplette, rollende Notation auf dem Bildschirm. In dieser Situation können Musik wie Notation gleichzeitig angehalten und von einem bestimmten Takt an wieder gestartet werden. Ergänzend läßt sich das Gesamtklangbild der Musik durch Veränderung der Parameter über den Synthesizer modifizieren und bietet so dem Anwender die Möglichkeit, das Arrangieren zu erlernen.

Music Composition Language

Ist eine bestimmte Sicherheit beim Notenlesen erreicht, fällt es leichter, Musik mit Hilfe der Computertastatur zu schreiben. Dazu gehört manchmal auch die Eingabe von Aufführungsanweisungen, ohne daß eine gleichzeitiger Vergleich zum Synthesizerklang erforderlich ist. Das Ergebnis ließe sich dann mit dem beabsichtigten Klang vergleichen. Wenn man so weit gekommen ist, könnte man auf die Fünf-Linien-Notation verzichten und sich anderen musikalischen Notationssystemen zuwenden wie etwa der MCL (Music Composition Language). Für die elektronische Musik ist MCL weitaus geeigneter bei der Dateneingabe, da



Künstlicher Klang

Einige Hersteller haben Synthesizer entwickelt, die nicht mehr durch Tasten, sondern durch Gitarrensaiten aktiviert werden. Beim Roland GR 700 verwendet man normale Gitarrensaiten. Komplexe Klanginformationen werden hexaphonisch eingegeben und an den Synthesizer geleitet, wo andere Parameter hinzugefügt werden. Dieselbe Technik findet bei der SynthAxe Anwendung, einem Synthesizer, der von einem 6809-Prozessor gesteuert wird. Dieser Synthesizer-Prototyp entnimmt Sounddaten einer elektrischen Verbindung zwischen Saite und Bund. Sensoren am Hals greifen die entsprechenden Variationen wie Beugen und Gleiten ab.

sie Spezifikationen beinhaltet, die ausschließlich für die Erzeugung elektronischer Klänge bedeutsam sind.

Für viele Microcomputer-Besitzer ist das Verständnis der Liniennotation oder von MCL eine wichtige Voraussetzung, um optimal mit MIDI arbeiten zu können. Eines der wesentlichen Hindernisse für Auszubildende an Schulen und Universitäten ist jedoch die gegenwärtige Form der Musikerziehung. Hauptstudienfach der meisten dieser Studenten bleibt unverändert die klassische europäische Musik. Häufig wird dabei elektronische Musik mit avantgardistischen oder gar radikalen Kompositionen gleichgesetzt, wie man sie aus den letzten beiden Jahrzehnten kennt, oder auch mit Pop-Musik. Keiner der genannten Bereiche wird als der klassischen Musik ebenbürtig angesehen. Einige Klassik-Enthusiasten stellen sogar in Frage, daß es sich dabei überhaupt um „Musik" handelt.

So wird offensichtlich, daß die Ausbildungsmöglichkeiten, die MIDI bietet, in den musikalischen Hauptströmungen bislang nicht stark genug berücksichtigt werden, besonders unter dem Gesichtspunkt, daß neben musikalischen auch Programmierfähigkeiten erforderlich sind. Auf der anderen Seite kann es kaum Computerkurse geben, in denen Soundmöglichkeiten so vermittelt werden, wie in einer Klasse für MIDI-Synthesizerspieler. Sollte MIDI in diesem Bereich populärer werden, müßten beim Komponieren Kopfhörer eingesetzt werden. Da die meisten MIDI-Einheiten jedoch so konstruiert sind, daß sie als Schnittstelle zu einem oder mehreren Synthesizern fungieren, wären hierfür andere Signale erforderlich.

Bei Bühnendarbietungen nutzt man MIDI hauptsächlich dazu, eine Reihe von Synthesizern, Sequenzern und Rhythmusmaschinen zu einem einzigen kontrollierbaren System zu vereinen. Musiker, die bei ihren Darbietungen alle Möglichkeiten der Sequenzierung benutzen, haben jedoch ungeheure Angst davor, daß die Synchronisation der einzelnen Einheiten gestört wird, was einen Zusammenbruch der Musik zur Folge hätte. Künstler wie die Thompson Twins und Howard Jones sind dafür bekannt, daß sie lieber Bänder ihrer Studioaufnahmen als Hintergrundmusik verwenden und dazu „live agieren", als Risiken der vorgenannten Art einzugehen. Mit MIDI sind allerdings zumindest theoretisch störungsfreie Konzerte sicher.

Konzentration auf das Wesentliche

Seit Anfang der siebziger Jahre galt ein Tasteninstrument als Synthesizer, das mit einer Reihe von Knöpfen und Schiebern ausgestattet war, mit denen die Eigenschaften der Klänge verändert wurden. Wird aber ein Keyboard-Synthesizer an MIDI angeschlossen, um so einen zweiten oder dritten zu steuern, sind weiter Keyboards entbehrlich. Je stärker sich MIDI durchsetzt, um so mehr „Synthesizer-Module" gelangen auf den Markt. Dabei handelt es sich lediglich um klangerzeugende und sequenzierende Einheiten, die zuvor Teile von Keyboard-Instrumenten waren.

Es gibt jedoch noch eine andere Entwicklung, die vor MIDI stattfand: die Möglichkeit elektronischer Klangerzeugung, gesteuert durch auf Saiten gespielte Töne (bei der Gitarre) und durch Atem- bzw. Mundkontrolle bei anderen Instrumenten. Verglichen mit dem

rein mechanischen Vorgang des Anschlagens einer Taste auf einem Keyboard, erlaubt das Zupfen einer Saite oder das Schwingen eines Blättchens Töne mit zahlreichen Varianten, die auf das betreffende Instrument übertragen werden. Codiert man diese Information digital und überträgt sie mittels MIDI auf ein hinter der Bühne befindliches Synthesizer-Modul, kann der Saxophonist einer Gruppe zugleich auch die Rhythmus-Section oder sogar die gesamte Elektronik kontrollieren. Yamaha hat in seinen DX7-Synthesizer eine Atem-Steuerung integriert, und die unlängst entwickelte digitale Gitarre mit der Bezeichnung SynthAxe bietet die Möglichkeit, MIDI zur Steuerung der Ausgabe eines Fairlight einzusetzen.

Das bedeutet, daß der Saitenklang auf dem Yamaha durch ein Saxophon produziert werden kann und der Anschlag einer Gitarre auf einem Fairlight einen Trompetenklang zur Folge hätte. Noch sind diese Entwicklungen nicht ausgereift und zu teuer. Die SynthAxe, Kostenpunkt rund 30 000 Mark, ist eine sehr kostspielige „Gitarre", doch daraus werden Tendenzen für künftige Live-Darbietungen deutlich. Die Tastaturen auf der Bühne werden zahlenmäßig abnehmen. Saiten-, Blas- und stimmbare Percussion-Instrumente, wie etwa Vibraphone, gewinnen an Bedeutung. Akustische Klänge könnten dominieren, weil das Sound-Sampling preiswerter wird.

Viele Gruppen, die über beachtliche Konzerterfahrung verfügen, lassen sich bei der ersten Begegnung mit einem modernen Aufnahmestudio einschüchtern. Sie werden mit Betriebssystemen konfrontiert, die sie noch nie zuvor gesehen haben. Und doch erwartet ihre Plattengesellschaft die Produktion von „größeren und besseren" Versionen ihrer auf der Bühne so erfolgreichen Titel. Das wäre ungefähr so, als würde man eine Laienspielgruppe in eine perfekte Filmkulisse stecken und erwartete gleich einen Kassenhit als Filmergebnis. Zuweilen gelingt es, und die Sorgen erweisen sich als gegenstandslos. Oft aber gehen im technischen Labyrinth die ursprünglichen künstlerischen Ideen verloren.

Sehr häufig kommt es dadurch zu Fehlschlägen, daß Gruppen auf ihr vertrautes Instrumentarium verzichten – und damit ja eigentlich auf ihren originären „Sound" – und die scheinbar so begehrenswerten Studioinstrumente benutzen. Doch eine musikalische Idee, die auf einem Mini-Synthesizer stimmt, zeigt auf einem Fairlight ein völlig anderes Ergebnis.

Sind die Musiker einer Gruppe aber mit MIDI vertraut, und wurde zuvor ein Computer zur Aufzeichnung von Sequenzen und anderen musikalischen Steuerdaten benutzt, steht einer sinnvollen Studioarbeit nichts im Wege. Es sollte problemlos möglich sein, die Ideen auf dem Studioinstrumentarium auszuführen, da bekannt ist, daß alle Ideen und Sequenzen ohne großen Aufwand durch einfachen Synthesizer-Tausch realisierbar sind.

Der Einsatz von MIDI ist ebenso wichtig, wenn es um die Nutzung der Studioeinrichtung geht, die mit der Klangerzeugung selbst nichts zu tun hat. Ein logisches Mischpult beispielsweise ist auf einem komplizierten Computer aufgebaut, der den Mischvorgang steuert. Sind alle 24 Spuren eines Bandes gefüllt und die Musiken der einzelnen Instrumente einzeln aufgezeichnet, beginnt die schwierige Aufgabe des Ausgleichens und Mischens der Einzelelemente. Das ist die Endphase jeder Aufnahme. In diesem Stadium erfolgt üblicherweise die „Effekt"-Behandlung der Teile. Das heißt, sie werden in die Mischung eingebracht oder herausgenommen. Ein einzelner Trompetenton soll zum Beispiel nur an einer Stelle verstärkt werden. Das kann leicht danebengehen, da gleichzeitig 23 andere Klangereignisse stattfinden. Die Verwendung eines Computers entspricht da dem Sequenzieren durch MIDI.

Ein andere Technik, die ursprünglich für Videoschnitt- und Synchronisation entwickelt wurde, inzwischen aber auch Einzug in die Musikproduktion gefunden hat, ist die Anwendung von Zeit-Codes. Dieser, im Englischen als „Time-Code" bezeichnet, hat die Funktion einer digitalen Uhr und gibt ein auslösendes Signal, wird aber direkt aufs Band gebracht. Werden Musik und Videosequenzen synchronisiert, erfolgt die Übermittlung von Daten durch 80-Bit-Worte. Musikalische Ereignisse und nur Sekunden dauernde Videoszenen können so exakt zusammengeschnitten werden (beispielsweise für Video-Clips). Ohne den Time-Code wäre dies kaum möglich.

Laurie Anderson, die New Yorker Dichterin und Künstlerin, kombiniert eine ungewöhnliche Mischung von Klängen mit Film, Tonband und Video-Technik und schafft so einen unverwechselbaren Stil. In ihren Songs „O Superman" oder „Mr. Heartbreak" verwendet sie alle Instrumente – von afrikanischen Glocken bis zu hochentwickelten elektronischen Instrumenten wie Vocoder und Synclavier – bzw. spielt sie als Backup ein.

Prestel-Modem

Das in Deutschland im Aufbau befindliche Bildschirmtextsystem (Btx) orientiert sich in vielem an dem englischen Prestel-System. Auch bei uns könnten Geräte wie das hier vorgestellte Modem VTX5000 der Firma Prism (für Sinclair Spectrum) dem Heimcomputer-Benutzer das interessante Gebiet der Telekommunikation zugänglich machen.

Die Engländer stießen mit ihrem Bildschirmtextsystem Anfang der 70er Jahre zunächst auf wenig Gegenliebe. Es kostete zu viel, und außerdem war die Datenbank so wenig ergiebig, daß verglichen mit einer Zeitung kein wesentlicher Vorteil deutlich wurde. Diese Anfangsschwierigkeiten sind längst überwunden – das Informationsangebot ist geradezu explodiert und wird entsprechend genutzt. Beim deutschen Btx-System hat man daraus gelernt und ist gleich mit einem großzügigen Konzept an den Start gegangen.

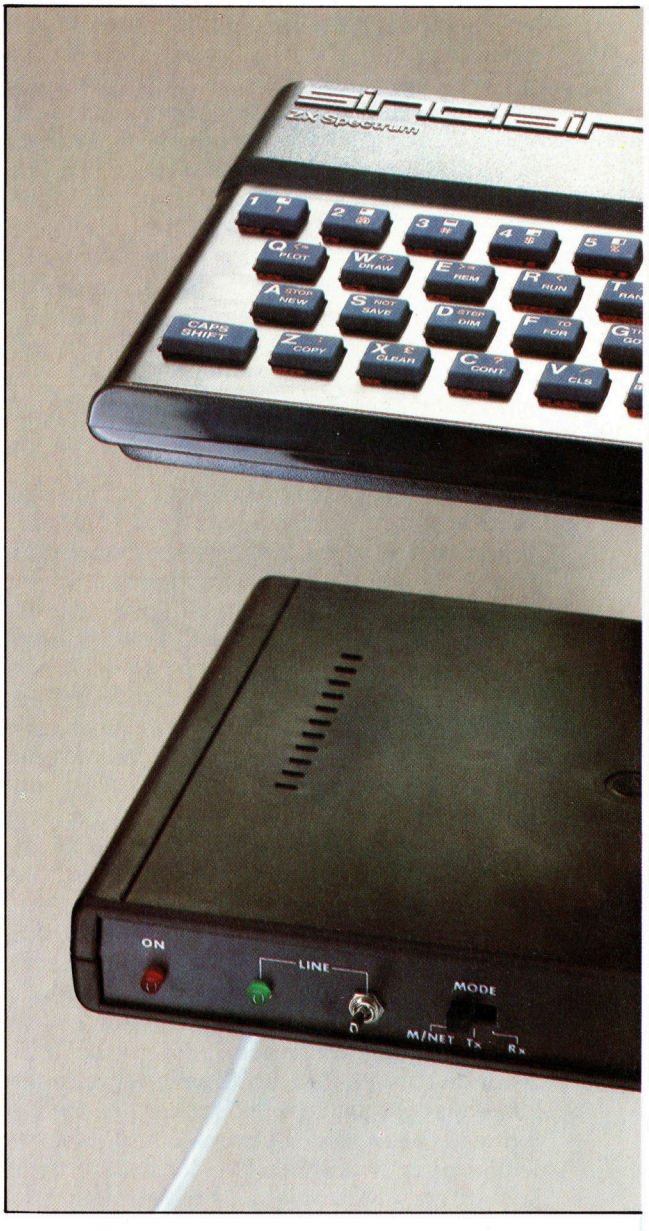
In England ist man inzwischen schon ein Stück weiter und hat sich die Tatsache zunutze gemacht, daß das Bildschirmtextsystem über ein geeignetes Interface und entsprechende Software für die meisten Heimcomputer erreichbar ist.

Um diese Chance wahrzunehmen, wurde von Prism das „Micronet" eingeführt – ein spezieller Sektor von Prestel, der ausschließlich Neuigkeiten und Informationen über Computer bringt. Modems und Software dafür gab es auch bald, und das Projekt erwies sich als äußerst erfolgreich.

Die Entwicklung eines Micronet-Modems für den Spectrum war nicht einfach: Es fehlt eine serielle Schnittstelle, etwa nach RS232-Norm (weshalb auch keins der üblichen Modems verwendbar ist). Außerdem beträgt die Bildschirmdarstellung beim Spectrum 24 Zeilen zu je 32 Zeichen, während Prestel mit 24 Zeilen à 40 Zeichen arbeitet und mit einer komplizierten Grafik.

Das Prism-System beinhaltet daher eine Spectrum-Schnittstelle, ein 1200/75-Baud-Modem sowie spezielle Software: Standardprozeduren für das Einloggen und einen Umsetzer, der über den Grafikmodus des Spectrum eine Zeilenlänge von 40 Zeichen simuliert.

Das VTX5000 wird über die Erweiterungssteckleiste an der Unterseite des Computers



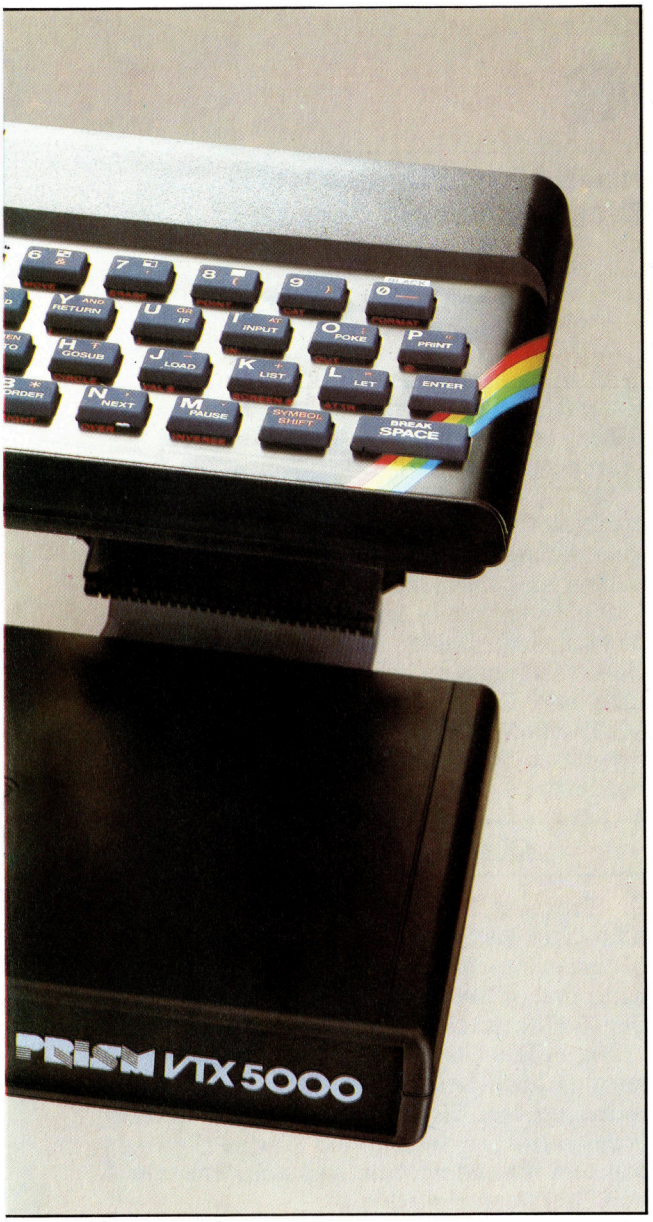
angeschlossen. Das Verbindungskabel ist mit einer zusätzlichen Steckerleiste für andere Peripheriegeräte wie Drucker oder Microdrive ausgestattet. Das Modem wird wie üblich direkt mit dem Telefonnetz verbunden.

Verbindung aufnehmen

Nach dem Einschalten des Rechners wird automatisch die Micronet-Software aufgerufen, die innerhalb des Modems in einem ROM gespeichert ist. Der Umgang mit den Programmen ist leicht erlernbar, da sie durchweg menügesteuert sind.

Der Benutzer muß sich zunächst „einloggen", das heißt seine Prestel-Teilnehmernummer eintippen. Diese Zahl „merkt" sich der Rechner, solange er eingeschaltet ist. Sie muß bei jeder „Sitzung" nur einmal eingegeben werden, auch wenn ein mehrmaliger Aufruf des Systems erfolgt.

Danach kann der Benutzer einen der vier



Das Modem VTX5000 für das englische „Prestel"-Bildschirmtextsystem wird direkt mit dem Spectrum verbunden. Der Telefonapparat ist am Modem einzustecken, das seinerseits an die Telefon-Wandsteckdose angeschlossen wird. Ein entsprechendes Modem für das deutsche Btx-System ist im Gespräch.

Prestel-Datenbank-Computer anwählen. Wenn als Antwort ein hoher Piepton kommt, betätigt man den „Line"-Schalter am Modem und legt den Telefonhörer auf – jetzt hat der Rechner Verbindung mit dem Prestel-Computer. Anschließend ist ein Paßwort einzugeben.

Nachdem der Anwender Zugang zur Datenbank hat, geht es wie bei jedem anderen Prestel-Adapter weiter. Zum „Umblättern" dienen statt der Tasten „*" und „#" die Spectrumtasten „Enter" und „Symbol Shift". Zusätzlich kann man jederzeit andere Micronet-Softwarefunktionen aufrufen und beispielsweise Bildschirmseiten über den Drucker ausgeben oder auf Cassette speichern, ohne die Verbindung zu unterbrechen. Außerdem besteht die Möglichkeit, Programme aus der Micronet-Bibliothek in den Rechner zu laden. In der „Telesoftware" von Micronet finden Sie kostenlose Programme und andere, für deren Benutzung eine Gebühr zu entrichten ist.

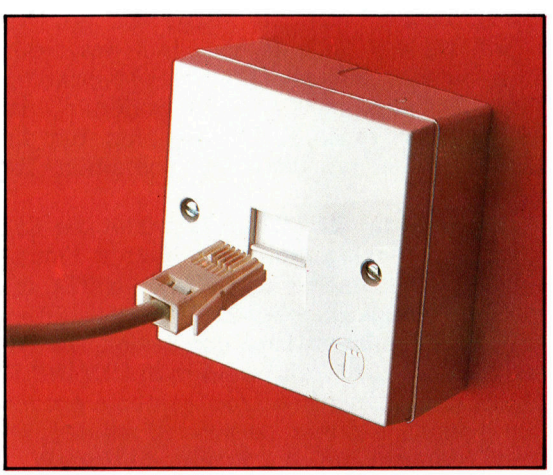
Mit Prestel und dem Micronet steht ein umfassendes Informationsangebot zur Verfügung – Nachrichten und Berichte, technische Beratung, Witze und Spiele, Kontaktanzeigen usw. Ferner ist Telekommunikation mit anderen Benutzern nach dem Prinzip des elektronischen Briefkastens möglich.

Wem die Nachrichten nicht aktuell genug sind oder wem es einfach zu langweilig wird, der kann versuchen, über das VTX5000 direkt mit anderen Spectrum-Benutzern Verbindung aufzunehmen. Das Modem ist so ausgelegt, daß damit auch ein telefonischer Datenaustausch zwischen zwei Spectrum-Computern bzw. VTX5000 möglich ist, und zwar mit der Übertragungsrate von 1200 Baud. Die ersten Modems wurden noch ohne die dazu erforderlichen Programme verkauft, aber inzwischen gehört die Software für diesen Datenverkehr zum Lieferumfang.

Spezielle Software

Es gibt noch eine Reihe anderer Übertragungsstandards für die Telekommunikation, die das VTX5000 aber nicht verarbeiten kann. Die wichtigste Einschränkung besteht darin, daß das Modem nicht auf die Übertragungsrate von 300 Baud einstellbar ist, die von vielen Computer-Freaks für den Informationsaustausch verwendet wird.

Wer alle Möglichkeiten der Telekommunikation ausschöpfen will, ist wohl mit einer RS232-Schnittstelle (speziell mit dem „Interface 1" von Sinclair) und einem universellen Modem besser bedient. Bereits seit einigen Monaten ist der Akustikkoppler „dataphon s 21 d" (Übertragungsrate 300 Baud) auf dem Markt, der auch mit dem Sinclair Spectrum betrieben werden kann. Neueren Datums ist das Datenübertragungsprogramm „Tekos" für den Spectrum. Tekos ist menügesteuert, ermöglicht eine Bildschirmausgabe von 64 Zeichen pro Zeile und ist für die Übertragung von Texten, die auf einem Wordprozessor erstellt wurden, geeignet. Das Paket enthält die Software, das Verbindungskabel vom Interface 1 zum Koppler sowie ein Handbuch.



Diese Steckdosen werden in England bereits bei der Neueinrichtung von Anschlüssen verwendet. Sie sind so geschaltet, daß die Leitung vor Störimpulsen geschützt ist.

Diagramme

In dieser Folge des LOGO-Kurses werden Möglichkeiten aufgezeigt, wie sich Datengruppen in Form von Diagrammen auf dem Bildschirm darstellen lassen.

Anhand von Diagrammen kann man numerische Daten auf einfache Weise veranschaulichen. Der Sinn dieser Grafiken besteht darin, dem Leser Zahlenmaterial und bestimmte Verhältnismäßigkeiten schnell zu vermitteln. Bei kommerziellen Anwendungen findet man häufig Tabellenkalkulationen, die mit Diagrammprogrammen gekoppelt sind und die eingegebenen, berechneten Werte anschließend in Diagrammform auf dem Bildschirm darstellen.

Derart komplizierte Vorgänge lassen sich mit diesem für den Commodore 64 entwickelten Programm natürlich nicht erledigen. Anhand der Prozeduren wird jedoch aufgezeigt, wie Sie mit LOGO Zahlen grafisch ausgeben können. Beachten Sie bitte die LOGO-Dialekte, da die nachstehenden Programme einige Anweisungen enthalten, die auf anderen Rechnern nicht verfügbar sind.

Der gesamte Arbeitsvorgang läßt sich in drei Schritte unterteilen:

a) Eingabe,
b) Suchen des größten Wertes (das ist zur Berechnung der Balkenlänge notwendig),
c) Ausfüllen des Diagramms, Wertzuweisung und Beschriftung.

Die Hauptprozedur heißt BARCHART. INIT bestimmt die Anzahl der im Programm benötigten numerischen Konstanten. COLORS enthält eine Farbliste zum Zeichnen von fünfzehn Balken. Sollte Ihnen die gewählte Farbzusammenstellung nicht zusagen, können Sie diese recht einfach ändern.

Die Eingabe für jedes der Diagrammelemente besteht aus zwei Angaben: der Beschriftung des Balkens und dem jeweiligen Wert, aus dem sich wiederum der Maßstab ergibt. Die Eingaberoutinen überprüfen, ob die eingetippten Werte auch gültig sind. Um die Dateneingabe zu beenden, schreiben Sie END.

Die Prozedur GET.TITLE verwaltet die Balkenbeschriftungen, wobei ein „leerer" Eintrag ignoriert wird. Anders als GET.QUANTITY, die nur numerische Eingaben akzeptiert, verarbeitet GET.TITLE Buchstaben und Zahlen. Die Daten (Beschriftung/Zahl) werden in GET.INPUT einer Liste zugefügt, und die Elemente werden von links nach rechts, beginnend mit dem ersten Datenpaar, auf dem Schirm dargestellt.

CALCULATE ruft GET.MAX auf, um den größten Wert zu ermitteln und die maximale Höhe der Balken sowie – je nachdem wieviele Balken zu zeichnen sind – die Breite des Diagramms auszurichten.

Mit Hilfe der Prozedur DRAW.CHART wird die Breite der einzelnen Balken berechnet und eine Achse gezeichnet. Danach werden die Balken eingefärbt und beschriftet.

Die Balkenbreite wird mit dem Faktor acht multipliziert. Diese Berechnung verhindert Überschneidungen der verschiedenfarbigen Elemente. Der C 64 kann nämlich im normalen LOGO-Grafikmode in einem acht mal acht umfassenden Pixelblock nur jeweils eine Farbe darstellen.

Diagrammerstellung

Die Prozedur DRAW.AXIS zeichnet eine vertikale Linie als Achse und markiert darauf den höchsten der eingegebenen Datenwerte. Anhand dieser Markierung lassen sich die übrigen Zahlen mit dem Maximalwert vergleichen.

Der Befehl COUNT ermittelt die Ziffernstellen des höchsten Wertes. Anhand dieser Berechnung legt die Prozedur fest, an welcher Position die Darstellung des Wertes beginnen soll, und verhindert somit, daß sich Achsenlinie und Zahl überschneiden.

WRITE wird aufgerufen, um den Text auf den Grafikschirm zu schreiben. Die Abstände zwischen den einzelnen Zeichen werden mit

LOGO-Dialekte

Viele LOGO-Versionen verfügen nicht über den Befehl STAMPCHART, mit dem sich Zeichen und Grafik kombinieren lassen. Die Farben sowie der Maßstab des Diagramms müssen je nach verwendetem Computer geändert werden. Diese Anweisungen enthält die Prozedur INIT.
Bei einigen LOGO-Versionen ersetzen Sie: PRINT1 durch TYPE und EMPTY? durch EMPTYP. Ferner können Sie bei einigen Dialekten EQUALP statt der Gleichheitszeichen verwenden.
Nach SETXY muß eine Liste folgen. Beachten Sie die unterschiedlichen Schreibweisen der IF-Abfragen:
IF EMPTYP :DATALIST [STOP]

Hilfe der Variablen XINC und YINC berechnet. Die Prozedur ruft den Befehl STAMPCHART auf, der an der gegenwärtigen Position der Turtle ein Zeichen darstellt. Leider existiert diese Anweisung nicht in allen LOGO-Versionen.

DRAW.CHART1 zeichnet nun die einzelnen Balken. Die Prozedur ruft jedes Element einzeln auf, wählt die nächste Farbe, berechnet die Höhe und verzweigt zu FILL, die die Balken schließlich mit der entsprechenden Farbe ausfüllt. Die Routinen LABEL und WRITE setzen die Feldbeschriftungen ein.

Eine weitere gebräuchliche Zahlendarstellungsart ist das Tortendiagramm. Hier sind einige Tips, die bei der Erstellung eines solchen Programms helfen:

* Die Daten werden in der gleichen Form eingegeben wie beim Balkendiagramm.
* Die Berechnung geht von der Gesamtsumme der Werte aus und ermittelt, welchen Anteil die einzelnen „Tortenelemente" – ausgehend von 360 Grad – belegen.
* Das Zeichnen der Elemente dürfte keine Schwierigkeiten bereiten. Jedoch könnten beim Einsatz der Farben beim Commodore 64 Probleme entstehen. Um zu verhindern, daß die verschiedenen Töne ineinanderlaufen, verwenden Sie den Befehl DOUBLECOLOR statt DRAW. Eine weitere Möglichkeit ist, in der Mitte des Diagramms ein „Loch" von zehn Einheiten frei zu lassen.
* Die Beschriftung der Elemente erfolgt in der gleichen Weise wie beim Balkendiagramm.

Versuchen Sie zur Übung, ein Balkendiagramm und ein Tortendiagramm mit den gleichen Daten nebeneinander auf dem Bildschirm darzustellen.

Diagramme

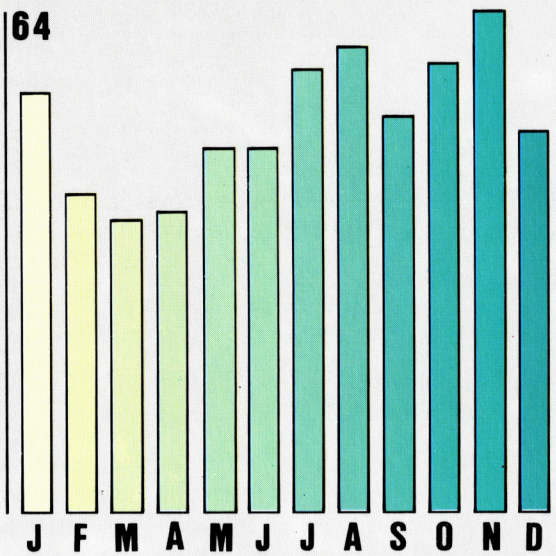
Mit Diagrammen lassen sich Zahlen grafisch darstellen. Aufgrund der langsamen Verarbeitungsgeschwindigkeit ist LOGO für diese Aufgabe jedoch nicht besonders geeignet.

Dateneingabe

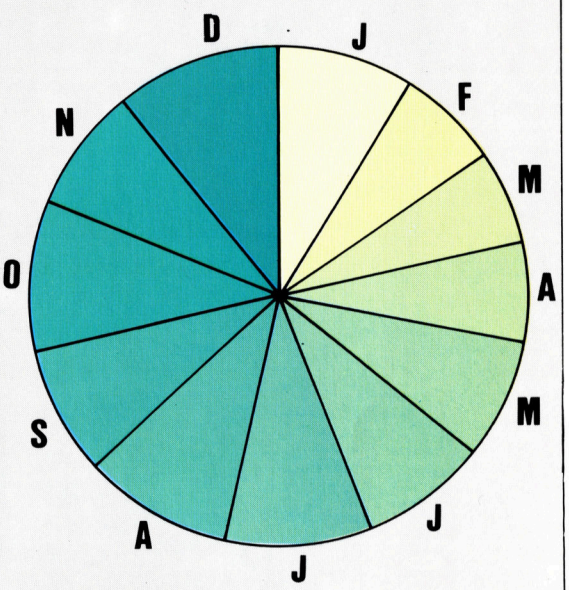
	London	New York
JAN	53	94
FEB	40	97
MAR	37	91
APR	38	81
MAY	46	81
JUN	46	84
JUL	56	107
AUG	59	109
SEP	50	86
OCT	57	89
NOV	64	76
DEC	48	91

Niederschlag (in Millimetern)

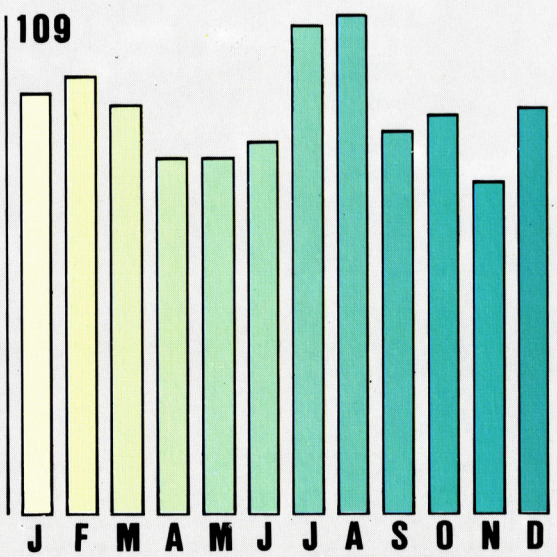
Balkendiagramm – London



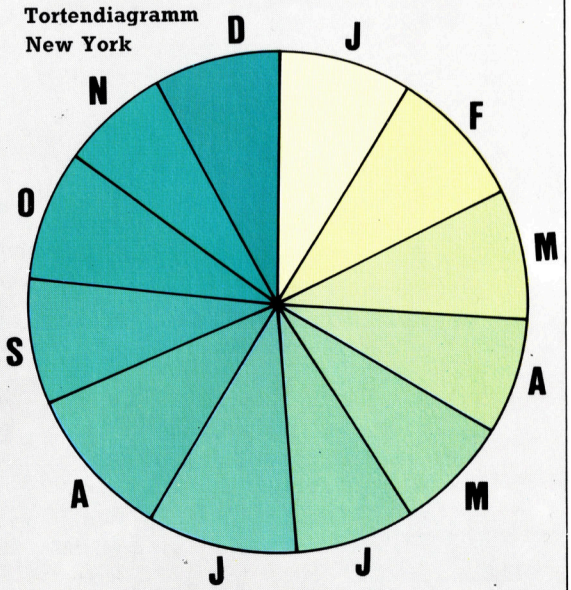
Tortendiagramm – London



Balkendiagramm – New York

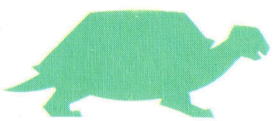


Tortendiagramm New York



Balkendiagramm-Programm

```
TO BARCHART
  INIT NODRAW
  GET.INPUT CALCULATE
  DRAW.CHART
END

TO INIT
  MAKE "COLORS [0 1 2 5 6 0 1 2 5 6 0 1 2 5 6]
  MAKE "BACKG 11
  MAKE "DATA []
  MAKE "HEIGHT 160
  MAKE "WIDE 190
  MAKE "XSTART ( – 104 )
  MAKE "YSTART ( – 40 )
END

TO GET.INPUT
  GET.TITLE
  IF FIRST :TITLE = "END THEN STOP
  GET.QUANTITY
  MAKE "DATA LPUT SENTENCE :TITLE :QUANTITY
    :DATA
  GET.INPUT
END

TO GET.TITLE
  PRINT "
  PRINT [ (TYPE END TO STOP) ]
  ( PRINT1 "TITLE: "' )
  MAKE "TITLE REQUEST
  IF EMPTY? :TITLE THEN GET.TITLE
END

TO GET.QUANTITY
  ( PRINT1 "QUANTITY: "' )
  MAKE "QUANTITY REQUEST
  IF EMPTY? :QUANTITY THEN GET.QUANTITY
    ELSE IF NOT NUMBER?
  FIRST :QUANTITY THEN GET.QUANTITY
END

TO CALCULATE
  MAKE "MAX 0
  GET.MAX :DATA
  MAKE "SCALE :HEIGHT / :MAX
END

TO GET.MAX :DATALIST
  IF EMPTY? :DATALIST THEN STOP
  IF LAST FIRST :DATALIST > :MAX THEN MAKE
    "MAX LAST FIRST
    :DATALIST
  GET.MAX BUTFIRST :DATALIST
END

TO DRAW.CHART
  DRAW BACKGROUND :BACKG
  HIDETURTLE FULLSCREEN
  MAKE "WIDTH ( ( ROUND ( ( :WIDE / COUNT
    :DATA ) / 8 ) ) * 8 ) — 8
  PENUP SETXY :XSTART :YSTART
  DRAW.AXIS
  DRAW.CHART1 :COLORS :DATA
  LABEL :DATA
END

TO DRAW.AXIS
  PENDOWN
  SETY YCOR + :HEIGHT + 10
  SETY YCOR — 10
  SETX XCOR + 4
  SETX XCOR — 8
  PENUP
  SETX XCOR — ( 8 * COUNT :MAX )
  WRITE 8 0 :MAX
  SETX XCOR + 4
  SETY :START
END

TO WRITE :XINC :YINC :NAME
  IF EMPTY? :NAME THEN STOP
  STAMPCHAR FIRST :NAME
  SETXY XCOR + :XINC YCOR + :YINC
  WRITE :XINC :YINC BUTFIRST :NAME
END

TO DRAW.CHART1 :PENCOLORS :DATALIST
  IF EMPTY? :DATALIST THEN STOP
  SETX XCOR + 8
  PENCOLOR FIRST :PENCOLORS
  PENDOWN
  FILL ( ( LAST FIRST :DATALIST ) * :SCALE )
    :WIDTH
  PENUP
  DRAW.CHART1 BUTFIRST :PENCOLORS
    BUTFIRST :DATALIST
END

TO FILL :LEN :WID
  IF :WID = 0 THEN STOP
  FORWARD :LEN RIGHT 90
  FORWARD 1 LEFT 90
  BACK :LEN RIGHT 90
  FORWARD 1 LEFT 90
  FILL :LEN :WID — 2
END

TO LABEL :DATALIST
  LABEL1 ( :XSTART + 8 + :WIDTH / 2 ) ( :YSTART
    — 10 )
    :DATALIST
END

TO LABEL1 :X :Y :DATALIST
  IF EMPTY? :DATALIST THEN STOP
  SETXY :X :Y
  WRITE 0 ( — 10 ) FIRST FIRST :DATALIST
  LABEL1 ( :X + :WIDTH + 8 ) :Y BUTFIRST
    :DATALIST
END
```



Fachwörter von A bis Z

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

Chain = Kette
Eine Kette ist eine bestimmte Ordnungsstruktur für Datensätze. Bei der Verkettung enthält jeder Datensatz einen Zeiger mit der Adresse des nächsten Satzes. Wie bei jeder anderen Kette hat aber auch hier der Bruch eines einzigen Gliedes (etwa ein Speicherfehler) entscheidende Folgen: den Verlust der gesamten Datei, weshalb meist auch noch alternative Zugriffsmöglichkeiten vorgesehen werden.

Die meisten Rechner können auch vollständige Programme verketten: Der Rechner lädt ein erstes Programm, läßt es ablaufen, lädt automatisch das nächste usw. Dies wirkt nach außen wie der Ablauf eines einzigen langen Programms. Bei Computerspielen wird beispielsweise oft zuerst nur die Spielanweisung geladen, die anschließend mit dem Hauptprogramm überschrieben werden kann. Auf diese Weise steht für das Hauptprogramm der ganze Speicher zur Verfügung.

Der Begriff „Daisy-Chain" kennzeichnet dagegen eine Hardware-Konfiguration, und zwar meist die Art des Anschlusses der Peripherie an den Computer. Eine Möglichkeit besteht beispielsweise in der Verwendung einer speziellen Rechner-Schnittstelle für jedes Gerät, eine andere in der Verbindung aller Komponenten durch ein gemeinsames Bus-System. Bei der Daisy-Chain wird an die Peripheriebuchse des Rechners ein Gerät angeschlossen, das seinerseits wieder über ein entsprechendes Interface verfügt, so daß dort ein zweites Gerät eingesteckt werden kann usw.

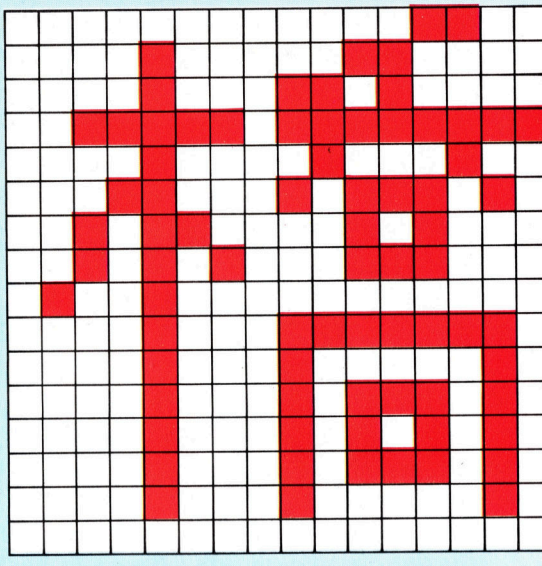
Channel = Kanal
Ein Kanal ist ein Transportmedium für Daten. Bei Computern ist damit nicht die Hardware (etwa eine Schnittstelle oder Leitung) gemeint, sondern das zugehörige Programm für die Datenflußsteuerung. Bei den meisten Betriebssystemen ist eine Ausgabe über den Bildschirm, den Drucker oder auf Diskette erst möglich, wenn der richtige Kanal geöffnet ist. Anschließend werden die Daten an den zugeordneten Kanal gesendet, der durch einen Namen oder eine Nummer gekennzeichnet ist, und das Betriebssystem leitet sie zu dem entsprechenden Gerät.

Die softwaremäßige Organisation von Kanälen hat auch den Vorteil, daß der Benutzer die Geräte nur indirekt spezifizieren muß. Wenn in den Ausgabeanweisungen eines Programms etwa über die Kanalnummer 5 der Drucker angesprochen wird und die Ausgabe aber auf den Plotter umgeleitet werden soll, brauchen Sie beim Programmstart nur anzugeben, daß über Kanal 5 jetzt der Plotter bedient wird, und müssen nicht die entsprechenden PRINT- und WRITE-Anweisungen einzeln ändern.

Character Generator = Zeichengenerator
Der Zeichengenerator ist für die Erzeugung der einzelnen Symbole zuständig. Üblicherweise ist im ROM für jedes Zeichen das zugehörige Punktmuster in Form einer Folge von Einsen und Nullen abgelegt.

Bei den meisten Heimcomputern läßt sich die Speicherung der Zeichenmuster aber auch in einen bestimmten RAM-Bereich übertragen. Dabei können Sie die Gestalt jedes alphanumerischen Zeichens oder Grafiksymbols beliebig verändern und sich so einen maßgeschneiderten Zeichensatz schaffen.

Check Digit/Check Bit = Prüfziffer/Prüfbit
Zur Fehlererkennung bei der Übertragung und Speicherung von Informationen dienen zusätzliche Prüfziffern oder Prüfbits. Solche Prüfzeichen werden vor allem bei der Datenfernübertragung oder beim Speichern auf Magnetträgern verwendet, weil dabei sehr leicht Fehler auftreten können.



Ein Zeichengenerator erzeugt Textzeichen und andere Symbole in Form eines Punktrasters. Bei einigen Rechnern können Sie das Punktmuster selbst definieren, wie am Beispiel des japanischen Schriftzeichens für „Brücke" gezeigt.

Prüfziffern sind Zusatzinformationen, die aus den Daten nach einer bestimmten mathematischen Vorgabe erzeugt und mit übertragen werden. Beim Datenempfang (oder Lesen des Speichermediums) wird nach der gleichen Regel verfahren. – Entspricht das Ergebnis nicht dem ursprünglichen Prüfzeichen, hat sich ein Fehler eingeschlichen.

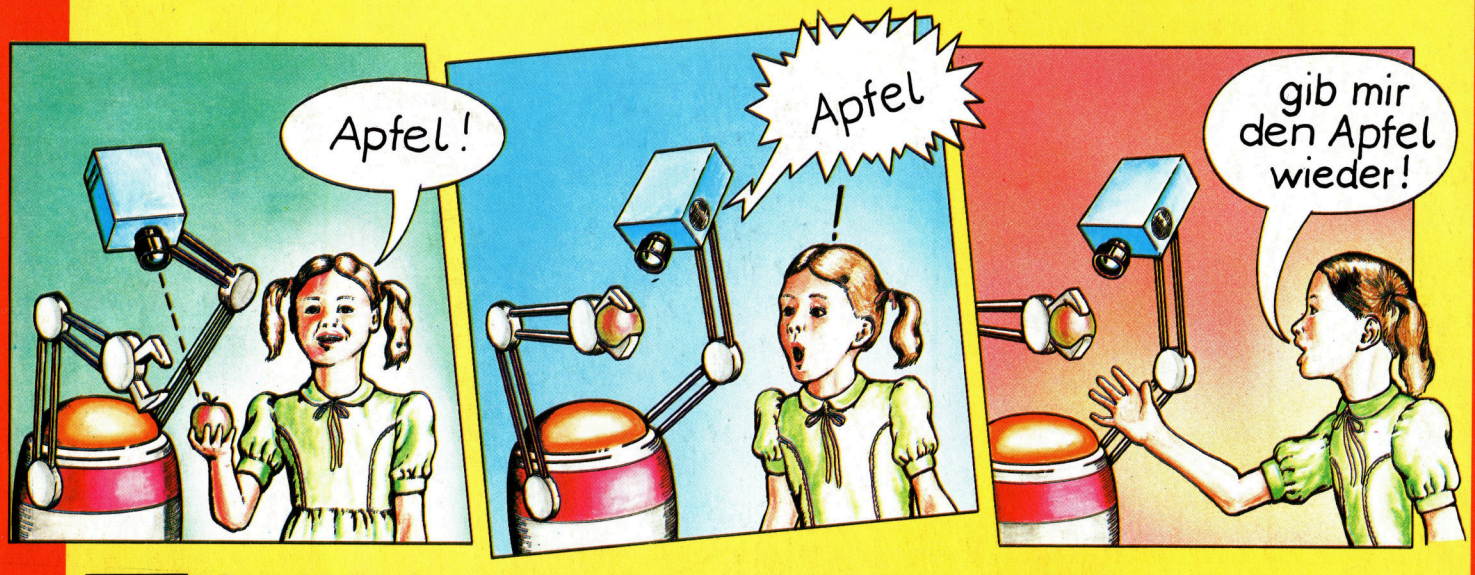
Die Erzeugung des Prüfzeichens kann mehr oder weniger aufwendig gestaltet werden. Sie kann beispielsweise so erfolgen, daß man alle Bytes in einem Datenblock addiert und aus dem Divisionsrest beim Teilen durch 256 ein Prüfbit bildet (Prüfsummen-Verfahren).

Bildnachweis

701: Steve Cross
702, 703, 707, 722: Kevin Jones
705: The Science Museum
711, 719, 720: Ian McKinnell
713, 714, 715, 724, 725: Chris Stevens
716: Bob Hall
717: Roy Ingrams
721: Marcus Wilson-Smith
U3: Liz Dixon

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

computer kurs Heft 27



Spracherkennung ist für Roboter eine schwierige Aufgabe.

Hardware: Mephisto PHC 64

Im Gegensatz zu anderen Microcomputern verfügt dieses Gerät über ein eingebautes Textsystem. BASIC wird als Cartridge geliefert und nur bei Bedarf eingesetzt.

Peripherie: Die Drucker

Gute Matrixdrucker können oft zwei- bis dreimal soviel kosten wie der Rechner selbst. Eine Alternative sind die „anschlagfreien" Drucker.

Sequentielle Dateien

Nachteil dieser Dateien ist die Zugriffmethode. Wir zeigen Verbesserungen.

Addieren mit Übertrag

In der Assemblersprache ist ADC ein wichtiger Befehl: Er bedeutet "Add with Carry" oder "Addiere mit Übertrag".

Verbesserungen und Reparaturen

Wer unserem Reparatur-Kurs folgt, wird um die Anschaffung spezieller Werkzeuge und Materialien nicht herum kommen. Wichtig ist aber auch das sorgfältige Arbeiten mit den empfindlichen Teilen.