Einsteigen - Verstehen - Beherrschen

DM 3,80 öS 30 sfr 3,80

computer kurs

Heft 30

Ein wöchentliches Sammelwerk

Cassettentricks

Disketten-Formatierung

Die wichtigsten Kleinroboter

Sharp Mini-Portables

Algebraische Strukturen

Koppler und Modems

computer kurs

Heft 30

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.

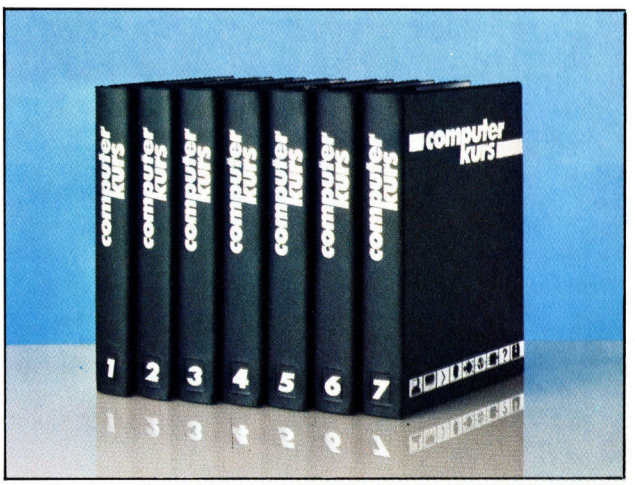
Schweiz: Der Sammelordner kostet sfr 15. Bitte wenden Sie sich an Ihren Kiosk; dort werden Sie jederzeit die gewünschten Exemplare erhalten.

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

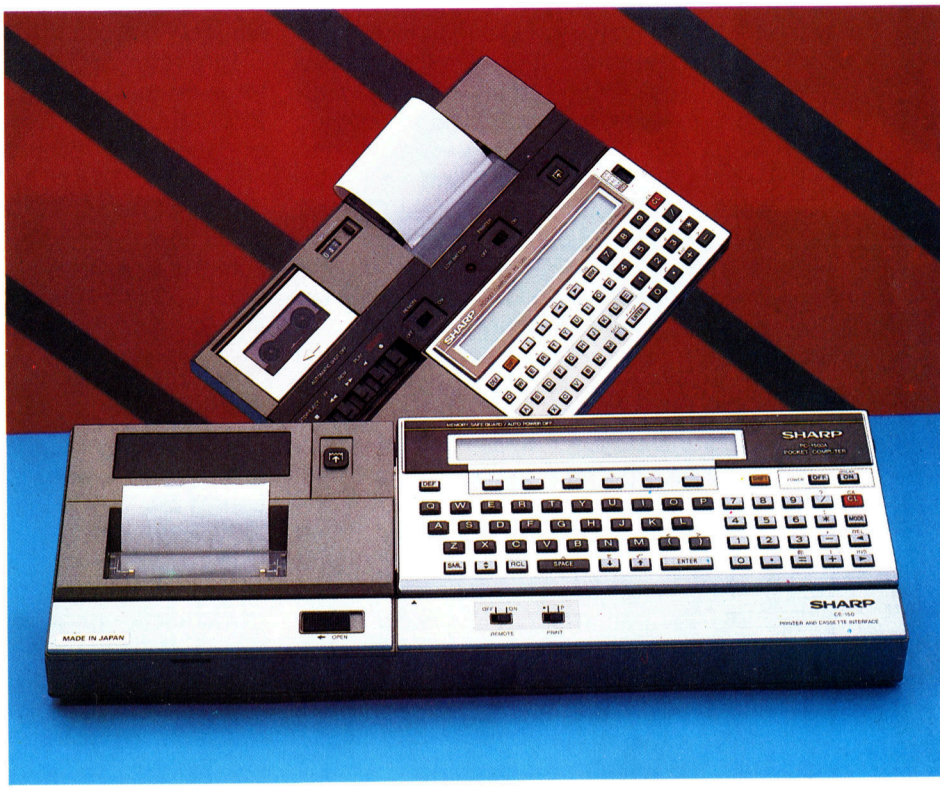
Vertrieb: Marshall Cavendish International Ltd., Heidenkampsweg 74, 2000 Hamburg 1, Tel.: 040/23 40 85



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

Im Taschenformat

Die beiden japanischen Firmen Casio und Sharp stellen Computer her, die in die Tasche passen. Trotz ihrer geringen Größe sind diese Maschinen echte Renner – wenn auch mit begrenzten Fähigkeiten. In diesem Artikel sehen wir uns die Geräte von Sharp genauer an.



Die Taschencomputer von Sharp sind mit einer CMOS 8-Bit-CPU, QWERTY-Tastatur und integriertem BASIC ausgestattet. Sie lassen sich durch zusätzliche Peripheriegeräte wie Drucker, Cassettenrecorder und RAM-Module erweitern.

Die Marketingstrategien von Sharp und Casio unterscheiden sich wesentlich. Sharp bietet zwei völlig verschiedene Geräte an, während Casio drei gleichartige Maschinen im Programm hat. Die beiden Sharp-Maschinen stehen nicht in Konkurrenz zueinander: Der PC-1251 ist ein kleiner Taschencomputer und kostet etwa 250 Mark. Der PC-1500A ist wesentlich größer, verfügt über mehr Möglichkeiten und kostet etwa 450 Mark.

Der Sharp PC-1251 wiegt 115 Gramm und hat die Größe von 135 × 70 × 10 mm. Seine Tasten sind vier Millimeter breit, und das Tastenfeld ist nur halb so groß wie eine normale Tastatur. Neben dem alphanumerischen Tastenfeld liegt ein Zehnerblock mit größeren Tasten, der sich als Taschenrechner einsetzen läßt.

Der PC-1251 besitzt eine Flüssigkristallanzeige für 24 Zeichen, die sich mit einem kleinen Rad an der Seite des Computers auf unterschiedliche Blickwinkel einstellen läßt. Rechts der Anzeige befindet sich ein Schalter für die drei Funktionsarten des Gerätes: Die Stellung RSV dient dazu, die Funktionen einzelner Tasten festzulegen, PRO schaltet die Programmierung ein, und mit RUN wird aus der Maschine ein Taschenrechner. In der vierten Stellung ist das Gerät abgeschaltet.

Eingeschränktes BASIC

Das BASIC kommt nicht an den Standard normaler Heimcomputer heran, ist für eine Maschine dieser Größe jedoch ausreichend. Einige Befehle wie ASC und CHR$ fehlen auf den Casio-Computern. Wie auf den kleineren Geräten von Casio setzt das BASIC des PC-1251 für Strings und Variablen die gleichen Buchstaben ein. Wenn also in der Variablen A eine Zahl gespeichert ist, kann die String-Variable A$ nicht verwandt werden.

Das BASIC unterstützt nur neun Fehlermeldungen, die jeweils durch einen Buchstaben gekennzeichnet werden. Dadurch lassen sich Programmfehler nur schwer spezifizieren. Über den Befehl PASS können Programme mit einem Kennwort geschützt werden. Die verfügbaren Zeilennummern sind von 1 bis 999 begrenzt. Mit Hilfe der Cursorsteuerungstasten lassen sich die Programmzeilen auf dem LCD hoch- und runterschieben. Bei allen Funktionsarten kann die Anzeige mit zwei weiteren Tasten auch seitwärts gerollt werden.

Da es für die Maschine nur wenig Software gibt, muß der Anwender seine Programme selbst schreiben. Hier hilft das Handbuch. Es enthält einen leichtverständlichen Führer durch das BASIC, neun kurze Programme zur Lösung mathematischer Probleme (beispielsweise Wurzelziehen, Statistik), Schreibmaschinenübungen und ein einfaches Spiel.

Der PC-1251 kann nur ein Programm speichern und nicht zehn wie die Geräte von Casio. Das Programm kann sich jedoch aus mehreren Unterprogrammen zusammensetzen, die durch END voneinander getrennt sind und über eine Unterroutine mit GOTO-Befehlen und den entsprechenden Zeilennummern gesteuert werden. Da der Arbeitsspeicher mit vier KByte sehr klein ist und sich nur kurze BASIC-Programme darin unterbringen lassen, müssen für jede ernsthaftere Anwendung der Drucker und das Cassettengerät dazugekauft werden.

Der zweite Taschencomputer von Sharp wurde für den kommerziellen Einsatz konzipiert. Der PC-1500A ist eine Weiterentwicklung des PC-1500. Der Preis für das Grundgerät liegt bei 450 Mark. Der PC-1500A mißt 195 × 85 × 25 mm und wiegt 375 Gramm.

Der PC-1500A hat eine bessere Tastatur und eine größere Anzeige als sein kleiner Bruder. Das LCD-Feld wurde auf 26 statt 24 Zeichen verbreitert, die Anzeigeregulierung des PC-1251 jedoch weggelassen.

Die BASIC-Version des PC-1500A ist weitaus besser als die des PC-1251. Variablennamen können von zwei Buchstaben dargestellt werden, und für numerische und String-Variablen lassen sich gleiche Variablennamen verwenden. Das BASIC unterstützt außerdem Klang- und Grafikeigenschaften. Der Befehl GPRINT kann auf dem LCD-Schirm Muster mit einer Auflösung von 7 × 156 Punkten darstellen, wobei jede Spalte mit sieben Punkten einzeln ansprechbar ist. Der Befehl BEEP steuert die Höhe und Dauer einzelner Töne.

Außer einem eingebauten Kalender besitzt der Computer auch eine Uhr, auf die mit der Variablen TIME zugegriffen werden kann. Die Zeit braucht nur einmal eingestellt zu werden, da sie auch nach dem Abschalten des Gerätes weiterläuft – eine Fähigkeit, die vielen Heimcomputern fehlt. Die Anweisung ON ERROR GOTO und Möglichkeiten, mit TRON und TROFF die Programmausführung schrittweise zu verfolgen, sind weitere zusätzliche Programmierhilfen. Das Standardgerät gibt 39 Fehlermeldungen aus. Weitere 16 stehen in Verbindung mit den Zusatzgeräten zur Verfügung. Die Meldungen werden jedoch wie auf dem PC-1251 nur als Zahlen dargestellt und sind somit nicht sehr hilfreich.

Folie mit Funktionsbezeichnungen

Einige Tasten sind mit den wichtigsten BASIC-Befehlen belegt. Eine Deckfolie gibt die Bezeichnungen der zehn Funktionen an. Unklar ist, warum Sharp diese Bezeichnungen nicht auf das Gehäuse aufgedruckt hat, da man die Folie leicht verlieren kann. Die sechs Tasten oberhalb der Haupttastatur lassen sich mit 18 Funktionen programmieren.

Auch die Besitzer des PC-1500A müssen ihre Programme im wesentlichen selbst entwickeln, da das Softwareangebot für dieses Gerät sehr klein ist. Im Lieferumfang ist ein Buch mit 53 Programmen enthalten. Sie bieten in fünf Hauptbereichen eine Vielfalt von Anwendungen an: Mathematik, Statistik, Elektrizität, Büroverwaltung und Spiele. Die Programme wurden ursprünglich für den PC-1500 geschrieben, laufen aber auch auf dem PC-1500A, da sich die beiden Maschinen nur durch die Größe ihres RAM-Speichers unterscheiden. Der PC-1500A verfügt über 8,5 KByte RAM, sein Vorgänger über 3,5 KByte.

Mit einer Cartridge läßt sich der Speicher vergrößern. Der Steckkontakt dafür befindet sich auf der Unterseite des Gerätes. Die vier angebotenen RAM-Cartridges sind jedoch alle recht teuer. Die beiden Standardpacks zu vier und acht KByte RAM verlieren ihren Speicherinhalt, sobald sie aus dem Steckschacht entfernt werden. Die beiden anderen Cartridges haben Batterien, die die gespeicherten Daten auch außerhalb des Computers erhalten. Diese beiden Cartridges haben eine Kapazität von acht bzw. 16 KByte.

Der Drucker/Plotter mit integrierter Cassettenrecorderschnittstelle bietet weitaus mehr fürs Geld. Mit der Schnittstelle lassen sich Programme auf normalen Cassettenrecordern speichern (Sharp bietet einen kompatiblen Recorder für ca. 150 Mark). Der Drucker/Plotter enthält vier verschiedenfarbige Kugelschreiberminen, mit denen sich Buchstaben und Grafik von guter Qualität zeichnen lassen. Die Papierbreite beträgt 57 Millimeter.

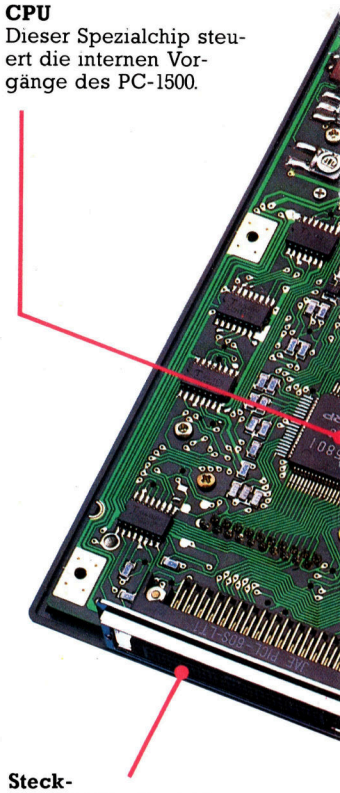
Druckersteuerung

Das BASIC enthält eine ganze Reihe von Steuerbefehlen für den Drucker/Plotter: CSIZE verändert die Zeichengröße, ROTATE druckt um 90, 180 oder 270 Grad gedrehte Zeichen, COLOR legt die Farbe fest, LF bewegt das Papier auf und ab, LPRINT druckt Text, LCURSOR und GLCURSOR bewegen den Zeichenstift im Text- oder Grafikmodus, SORGN bestimmt den Ausgangspunkt und LINE und RLINE zeichnen eine Gerade zwischen zwei Punkten, die sich durch absolute und relative Koordinaten angeben lassen. Der Drucker/Plotter befindet sich in einem separaten Gehäuse, und bei seinem Zubehör befindet sich ein Netzgerät, das auch Batterien auflädt.

Es gibt noch weitere Zusatzgeräte für den PC-1500A: Eine Centronics- und RS232-Schnittstelle für den Anschluß anderer Computer und Drucker. Weiterhin das „Software Board", ein großes, berührungsempfindliches Feld mit 140 programmierbaren Tasten, über die sich häufig eingesetzte Funktionen (zum Beispiel die Berechnung von Summen in einem Kalkulationssystem) abrufen lassen.

Wenn sich der PC-1500A auch zu einem interessanten System mit beeindruckenden Fähigkeiten ausbauen läßt, so gibt es in der gleichen Preisklasse doch viele Microsysteme, die flexibler sind und – als wichtigster Faktor – von einer breiteren Softwarepalette unterstützt werden.

Der vor kurzem vorgestellte Sharp PC-2500 wird gleich mit integrierter Software ausgeliefert. Diese umfaßt ein Telefonnummern-Verzeichnis sowie eine Tabellenkalkulation. Der PC-2500 wird mit einem 4-Farb-Printer/Plotter und Dokumentation zu einem Preis von circa 2000 Mark angeboten. Die RAM-Kapazität läßt sich mit Hilfe von zusätzlichen Karten von vier KByte bis auf 20 KByte aufstocken. Dank der beiden Schnittstellen können weitere Peripheriegeräte angeschlossen werden.



Batteriegehäuse
Hier befinden sich die vier Batterien des Gerätes.

CPU
Dieser Spezialchip steuert die internen Vorgänge des PC-1500.

Steckschacht für Cartridges
Über diese Schnittstelle können zusätzliche RAM-Packs an den PC-1500 angeschlossen werden, – auch die Einheit der Drucker / Cassettenrecorderschnittstelle.

Stromversorgung
Statt mit Batterien kann der PC-1500 auch über ein Netzgerät betrieben werden.

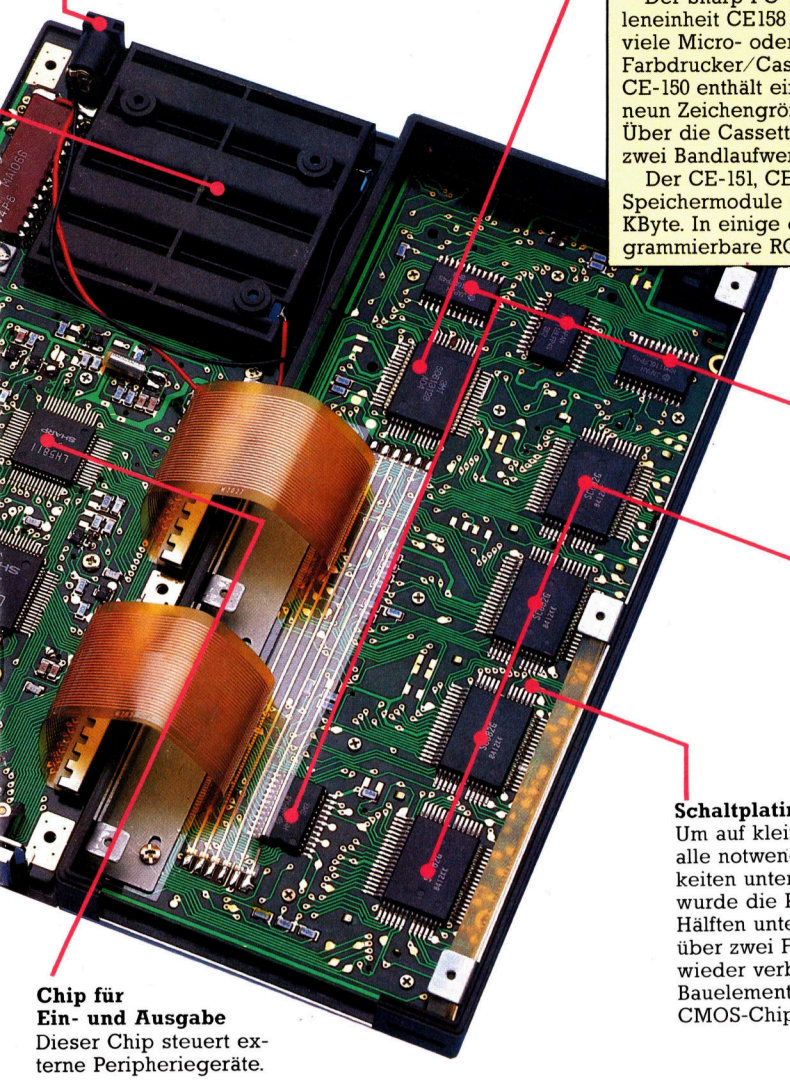
BASIC-ROM
Dieser Chip enthält das Microsoft-BASIC des Computers.

Zusatzgeräte

Der Sharp PC-1251 läßt sich auf die Drucker/Microcassetteneinheit CE-125 stecken. Das Zusatzgerät enthält einen Thermodrucker (24 Zeichen pro Zeile) und ein Laufwerk für Microcassetten.

Der Sharp PC-1500 läßt sich über die Schnittstelleneinheit CE158 per RS232C oder Centronics an viele Micro- oder Großsysteme anschließen. Die Farbdrucker/Cassettenrecorderinterface-Einheit CE-150 enthält einen Vierfarben-X-Y-Plotter mit neun Zeichengrößen und Grafikmöglichkeiten. Über die Cassettenrecorderschnittstelle können zwei Bandlaufwerke angeschlossen werden.

Der CE-151, CE-155, CE-159 und CE-161 sind Speichermodule mit CMOS-RAMs von zwei bis 16 KByte. In einige der Cartridges sind auch programmierbare ROMs eingebaut.



RAM-Chips
Diese Chips enthalten den Arbeitsspeicher von 8,5 KByte. Davon stehen dem Anwender 6,6 KByte zur Verfügung.

Steuerung der Anzeige
Diese vier Chips steuern die LCD-Anzeige.

Schaltplatine
Um auf kleinstem Raum alle notwendigen Fähigkeiten unterzubringen, wurde die Platine in zwei Hälften unterteilt und über zwei Flachkabel wieder verbunden. Viele Bauelemente sind im CMOS-Chip integriert.

Chip für Ein- und Ausgabe
Dieser Chip steuert externe Peripheriegeräte.

Sharp PC-1500A

ABMESSUNGEN
195 × 85 × 25 mm

GEWICHT
375 g

SPEICHERKAPAZITÄT
8,5K RAM

BILDSCHIRM-DARSTELLUNG
LCD-Anzeige, 26 Zeichen

WEITERE EIGENSCHAFTEN
Eingebauter Kalender mit Uhr, gutes BASIC, erweiterbarer Arbeitsspeicher.

Sharp PC-1251

ABMESSUNGEN
135 × 70 × 10 mm

GEWICHT
115 g

SPEICHERKAPAZITÄT
4K RAM

BILDSCHIRM-DARSTELLUNG
LCD-Anzeige, 24 Zeichen

WEITERE EIGENSCHAFTEN
Drei Modi, 18 programmierbare Funktionstasten; Recorder und Drucker sind verfügbar.

Sharp PC-1251

Dieser Taschenrechner ist das „Arbeitspferd" von Sharp. Da Texteingaben auf der winzigen QWERTY-Tastatur umständlich und ermüdend sind, ist er mehr für Ingenieure und Wissenschaftler geeignet als für den Büroeinsatz.

Sharp PC-1500A

Dieses Gerät ist ein außerordentlich vielseitiger Computer im Taschenrechnerformat. Obwohl er sich für viele Aufgaben eignet, wird er eher für ein teures Spielzeug oder einen beeindruckenden Taschenrechner gehalten.

Boolesche Algebra

Die Funktionsweise des Computers basiert auf der Abfolge von „High"- und „Low"-Zuständen in seinen Schaltkreisen. Damit der Rechner die Impulsfolgen weiterverarbeiten kann – also etwa eine Addition ausführen – sind spezielle Logikschaltungen nötig.

Theoretische Basis jedes Computers ist die Boolesche Algebra – eine mathematische Methode, die mit wenigen einfachen Regeln und Sätzen wahre von falschen Aussagen trennt.

Am Anfang dieses Kurses steht eine detaillierte Betrachtung der theoretischen und praktischen Merkmale logischer Schaltungen, die mit Beispielen aus der Technik Ihres eigenen Computers ergänzt werden. In der Booleschen Algebra gibt es drei Grundbegriffe: AND, OR und NOT (und, oder, nicht). Ihre Funktion kommt dem üblichen Sinn der Worte sehr nahe, wie dieses Beispiel verdeutlicht:

Wenn es Samstag ist AND die Sonne scheint, geht Thomas spazieren.

Ob Thomas spazierengeht, hängt also von zwei Dingen ab: Die Sonne muß scheinen, und es muß Samstag sein. Wenn er sich entscheidet, muß Thomas nur wissen, ob die Aussagen „die Sonne scheint" und „es ist Samstag" richtig oder falsch sind. Vier verschiedene Kombinationsmöglichkeiten dieser Eigenschaften gibt es, aber nur bei einer wird Thomas spazierengehen. Die verschiedenen Kombinationen von wahren und falschen Aussagen lassen sich auf einer Wahrheitstabelle darstellen:

DIE SONNE SCHEINT	ES IST SAMSTAG	THOMAS GEHT SPAZIEREN
FALSCH	FALSCH	FALSCH
FALSCH	WAHR	FALSCH
WAHR	FALSCH	FALSCH
WAHR	WAHR	WAHR

Die logische OR-Funktion läßt sich ähnlich darstellen. Betrachten Sie das Beispiel:

Udo geht essen, wenn Kai OR Eva mitkommt. Wieder haben wir zwei Bedingungen, von denen eine Handlung abhängt: Entweder begleitet ihn Kai oder Eva. Wie bei der AND-Funktion ergibt sich eine Wahrheitstabelle, die natürlich auch vier Kombinationsmöglichkeiten (zwei Bedingungen, jede entweder wahr oder falsch) darstellt:

KAI KOMMT MIT	EVA KOMMT MIT	UDO GEHT ESSEN
FALSCH	FALSCH	FALSCH
FALSCH	WAHR	WAHR
WAHR	FALSCH	WAHR
WAHR	WAHR	WAHR

Noch einfacher ist es bei NOT, der dritten logischen Funktion. Beispiel:

Ich gehe aus, wenn es NOT dunkel ist.

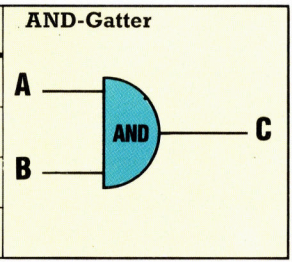
Hier ist nur eine Bedingung zu beachten, ob es dunkel ist oder nicht. Die Wahrheitstabelle zeigt daher bei diesem Beispiel nur zwei Kombinationsmöglichkeiten:

ES IST DUNKEL	ICH GEHE AUS
FALSCH	WAHR
WAHR	FALSCH

Die Schaltung eines Computers ist aus elektronischen Elementen aufgebaut, die den drei logischen Grundfunktionen AND, OR und NOT entsprechen. Eine wahre Bedingung wird dabei durch die Binärziffer 1, eine falsche Bedingung durch die Binärziffer 0 dargestellt. Man kann daher für jedes logische „Gatter" eine Wertetabelle aufstellen, die für eine Kombination von Eingangswerten den entsprechenden Ausgangswert verzeichnet. Die Namen der logischen Gatter entsprechen der Bezeichnung ihrer Funktion in der Booleschen Algebra.

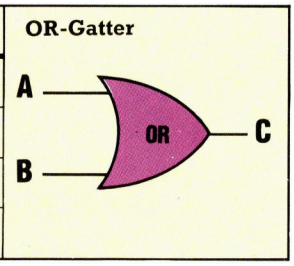
Diagramm und Wertetabelle eines AND-Gatters mit Eingängen A, B und Ausgang C:

A	B	C
0	0	0
0	1	0
1	0	0
1	1	1



In Worten ist die Funktion des AND-Gatters so zu beschreiben: Der Ausgang ist 1, wenn beide Eingänge 1 sind, sonst ist er 0. Die Boolesche Schreibweise für die Ausgabe des Gatters: A.B. Wahrheitstabelle und Diagramm des OR-Gatters:

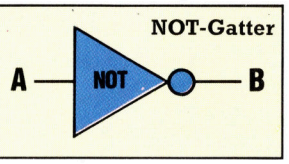
A	B	C
0	0	0
0	1	1
1	0	1
1	1	1



Die Funktion des OR-Gatters läßt sich durch den Satz beschreiben: Der Ausgang ist 1, wenn einer oder beide Eingänge 1 sind. Der Boolesche Ausdruck dafür lautet A+B.

Das NOT-Gatter hat, anders als das AND- bzw. OR-Gatter, nur einen Ein- und einen Ausgang.

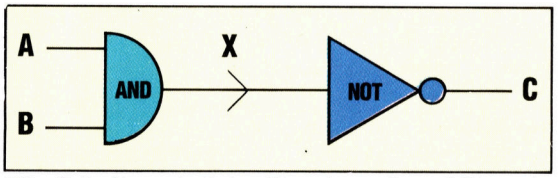
A	B
0	1
1	0



Beim NOT-Gatter zeigt der Ausgang immer den umgekehrten Wert des Einganges. Die Boolesche Schreibweise für die Ausgabe ist $\overline{A}$.

Verknüpfung logischer Gatter

In einem Computer sind eine Vielzahl logischer Gatter zu einem komplexen Schaltkreis verbunden oder „vernetzt". Auch die Funktion vernetzter logischer Gatter ist durch eine Wahrheitstabelle zu beschreiben, die jeder möglichen Eingabe die entsprechende Ausgabe zuordnet. Ein einfaches Beispiel:

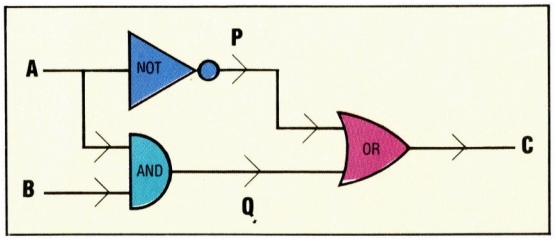


Die Schaltung hat die Eingänge A und B sowie den Ausgang C. Den Ausgang des ersten Gatters haben wir mit X bezeichnet, damit sich die Wertetabelle leichter aufstellen laßt. Durch die zwei Eingänge sind vier Kombinationen der Eingangsvariablen möglich

A	B	X	C
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

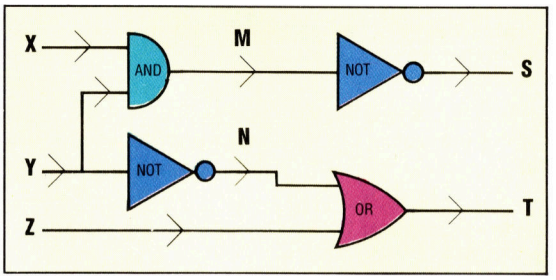
Der Ausgang des AND-Gatters führt über das NOT-Gatter zum „Ergebnis": Ausgang C.

Die nächste Schaltung ist schwieriger, obwohl es wieder nur zwei Eingänge und damit vier Zeilen in der Tabelle gibt. Rechts in der Wahrheitstabelle (P, Q und C) findet sich die umgestellte OR-Tabelle wieder.



A	B	P	Q	C
0	0	1	0	1
0	1	1	0	1
1	0	0	0	0
1	1	0	1	1

Eine Wahrheitstabelle läßt sich auch für Schaltungen mit mehr als zwei Eingängen und einem Ausgang aufstellen. Unten ein Beispiel für drei Ein- und zwei Ausgänge:

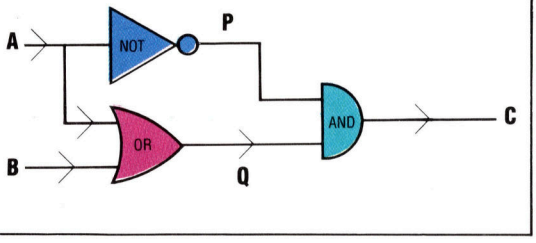


Bei drei Eingängen gibt es 2*2*2=8 mögliche Kombinationen für die Eingabe:

X	Y	Z	M	N	S	T
0	0	0	0	1	1	1
0	0	1	0	1	1	1
0	1	0	0	0	1	0
0	1	1	0	0	1	1
1	0	0	0	1	1	1
1	0	1	0	1	1	1
1	1	0	1	0	0	0
1	1	1	1	0	0	1

ÜBUNG 1

1. Schreiben Sie die Tabelle zu dieser Aussage: „Wenn er den Führerschein hat OR wenn ein Fahrlehrer dabei ist, darf Max fahren."
2. Welche Wahrheitstabelle gehört zu dieser Aussage: „Ein Programm kann geladen werden, wenn ein Cassettenrecorder OR ein Diskettenlaufwerk vorhanden ist AND das Programm NOT für einen anderen Rechner geschrieben wurde"?
3. Ermitteln Sie die Wahrheitstabelle für die unten abgebildete Schaltung:



Tal der Trolle

Abenteuer-Programme erlauben den Spielern, heldenhafte Taten in phantastischer Umgebung zu vollbringen. Spieler verbringen oft Wochen damit, die Lösung zu finden. Hier wird Bug-Bytes „Twin Kingdom Valley" als Prototyp eines Abenteuerspiels vorgestellt.

Die Computer-Abenteuerspiele entwickelten sich aus den „Dungeons and Dragons"-Rollenspielen. Zu Beginn der sechziger Jahre entwickelten Programmierer auf Großrechnern erste Versionen für Computer und nutzten dabei die großen Speicherkapazitäten aus, um alle nur vorstellbaren Einzelheiten dieser phantastischen Welten voller Zauberer und Monster, Zwerge und Trolle zu speichern. Die heutigen Abenteuerspiele für Computer stammen von diesen Prototypen ab, doch die Handlungsorte finden sich nun überall – das reicht von verlassenen Raumschiffen bis hin zu den Straßen Chicagos in der Gangster-Blütezeit der dreißiger Jahre.

Das erste, sehr gut verkaufte Grafik-Adventure war „The Hobbit" nach J. R. R. Tolkiens gleichnamigem Bestseller. Das hier vorgestellte Abenteuer „Twin Kingdom Valley" verwendet interessante Grafiken und spielt in einer Umgebung aus mittelalterlichen Burgen und tiefen Wäldern.

Wenngleich Grafikelemente einem Programm „mehr" geben als reine Text-Adventures, so können sie doch nicht die Einfallslosigkeit eines Programmierers kaschieren. – Man sieht es an diesem Spiel. Die Geschichte ist sehr einfach. Der Spieler übernimmt die Rolle eines Wanderers. Er gelangt in ein Tal, das von zwei Königen beherrscht wird, die einander bekriegen. Es handelt sich dabei um den „Wüsten-König" und den „Waldland-König". Im Tal befinden sich mehrere Flüsse. Sie münden in den magischen See „Watersmeet". Beim Durchstreifen des Königreiches muß der Spieler – ein gewinnsüchtiger Held – so viele Schätze wie möglich einsammeln. Ist genug zusammengekommen, und hat der Spieler 1024 Punkte erreicht, geschieht etwas Überraschendes (das wir aber, um die Spielspannung nicht zu nehmen, nicht preisgeben).

Spielsteuerung

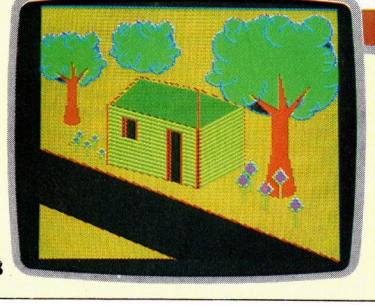
Bewegungen und Handlungen werden durch Eingabe entsprechender Anweisungen gesteuert. Das Programm erkennt 23 Verben, die mit spielbezogenen Gegenständen in Verbindung stehen. Bewegungsrichtungen werden durch Eingabe der Kompaßzahlen geändert, ergänzt um die Worte „up" oder „down". Auch andere Wesen bevölkern das Königreich. Will man deren Besitztümer erlangen, muß „ask" eingegeben werden. Allerdings stößt man in den meisten Fällen schon beim Versuch zu fragen auf heftigen Widerstand.

175 der Spielstätten sind grafisch dargestellt. Besonders der Acorn B braucht bei diesem Programm sehr viel Speicherplatz zur Darstellung der hochauflösenden Bilder. Deshalb sind die meisten Bilder Kombinationen derselben Grund-Konturen. So setzt sich zum Beispiel ein Wald aus zehn oder zwölf Baumkonturen zusammen, die in verschiedenen Größen wiederholt werden. Die größere Speicherkapazität des Commodore und die Sprite-Grafik-Möglichkeit sind der Grund dafür, daß in einigen Darstellungen Animation erfolgen kann. So klettern Eichhörnchen in den Bäumen und Wasser tropft von Stalaktiten. Die Grafik kann abgeschaltet werden, belegt aber dennoch Speicherplatz, der besser zur aufregenderen Gestaltung dieses Abenteuers hätte genutzt werden können.

Königreiche der Erfahrung

Die grafische Darstellung beim Acorn B besteht aus der Kombination verschiedener Grundkonturen, wogegen bei der Commodore-64-Version durch die Verwendung von Sprites Bewegung ins Spiel kommt.

Commodore 64



Acorn B

Twin Kingdom Valley: Für Acorn B und Commodore 64

Hersteller: Bug-Byte-Software, Mulberry House, Canning Place, Liverpool L1 8JB

Autor: Trevor Hall

Joysticks: Nicht erforderlich

Format: Cassette

Buffer für den User Port

Diese Folge geht näher auf die Datenausgabe via User Port ein. Mit einem vielseitigen Zusatzgerät können Sie die Anschlußmöglichkeiten Ihres Rechners erweitern.

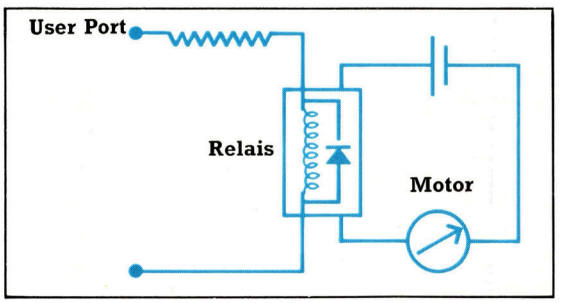
Im Computer-Bereich versteht man unter einem Buffer einen Kurzzeitspeicher, der den Datenaustausch innerhalb des Rechner-Systems unterstützt. Anders in der klassischen Elektrotechnik: Hier ist ein Buffer eine Sicherheitseinrichtung, die ein Gerät vor schädlichen Einflüssen durch ein anderes Gerät schützen soll. Soll der User Port Ihres Computers etwa einen Motor ansteuern, muß darauf geachtet werden, daß die empfindlichen Bauteile des Rechners dabei nicht überlastet werden.

Das Steuer-IC des User Port ist für Spannungen von 0 bis 5 Volt und Ströme von nur wenigen Milliampere geeignet. Sie müssen also sicherstellen, daß an keinem Anschluß eine höhere Spannung auftritt und der Stromfluß 20 mA nicht überschreitet.

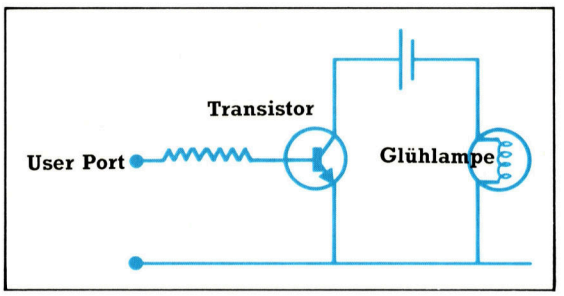
Im letzten Artikel wurde demonstriert, daß sich der Inhalt der Datenregister durch Verbinden der User-Port-Anschlüsse verändern läßt. Wird eine Speicherzelle nur mit Masse (0 Volt) verbunden, treten weder gefährliche Spannungen noch große Stromstärken auf – besondere Sicherheitsvorkehrungen waren überflüssig. Beim Anschluß externer Geräte ist das anders. Dabei sollte die Schaltung über ein Relais erfolgen, das über einen Schutzwiderstand vom User Port gesteuert wird.

So können größere Verbraucher gesteuert werden, ohne daß der betreffende Anschlußpunkt des User Ports mehr als 20 mA liefern muß. Mit dem Ohmschen Gesetz (Spannung = Strom*Widerstand) läßt sich der Wert des Schutzwiderstandes leicht ausrechnen:

U = I×R
R = U / I
R = 5 Volt / 0,02 Ampere
R = 250 Ohm

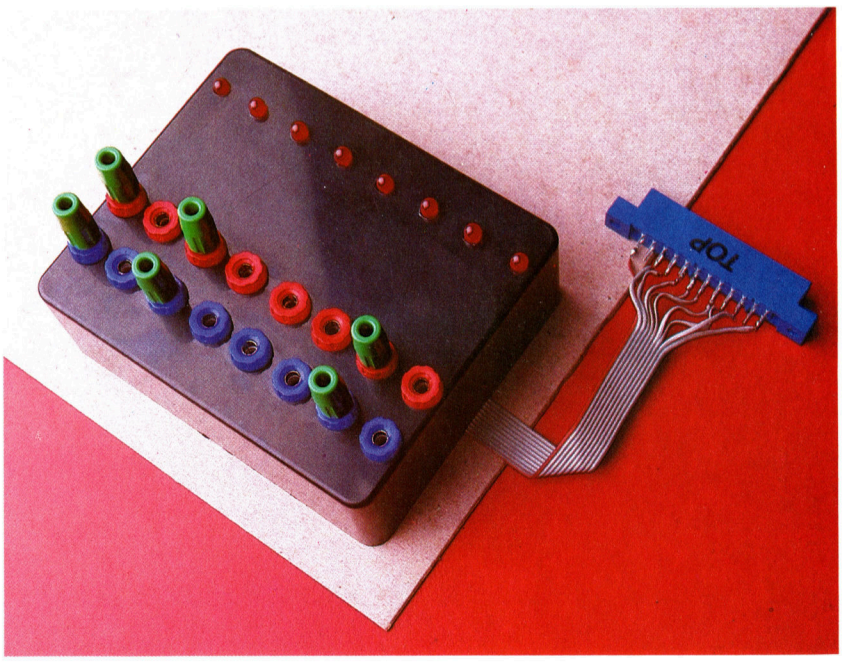


Eine zweite Möglichkeit ist es, den geringen Ausgangsstrom des User Ports durch einen nachgeschalteten Transistor zu verstärken:



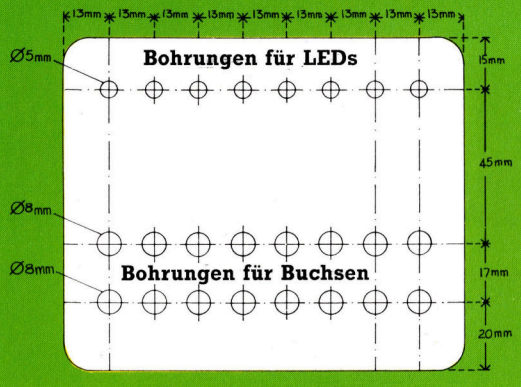
Wir haben uns bei unserer Bauanleitung für den User-Port-Buffer für die Transistorversion entschieden. Sie ist preisgünstiger und bequemer als eine Relaisschaltung. Inzwischen sind bereits Chips auf dem Markt, auf denen je acht integrierte Transistor-Schaltungen untergebracht sind.

Mit dem gebufferten User Port können Sie durch einige Zusatzschaltkreise eine Vielzahl von Geräten steuern. Das reicht von der einfachen Leuchtdiode über den Schwachstrommotor bis zu Haushaltsgeräten wie dem Fernse-



Der Buffer-Kasten wird mit dem im letzten Artikel beschriebenen Kabel am User Port des Acorn B oder Commodore 64 angeschlossen. Der Buffer schützt Ihren Computer vor Überlastung durch zu hohe Ströme oder Spannungen. Die LEDs zeigen den Zustand der Ausgangsleitungen des Ports an. Mit den Buchsen und Steckern können die Eingangsleitungen ein- und ausgeschaltet werden.

Bohren des Gehäuses



Die Leuchtdioden und die farbigen Buchsen sollten nach dieser Schablone gebohrt werden. Das Kunststoffmaterial des Gehäuses läßt sich gut bearbeiten. Damit es keine Schrammen gibt und Sie beim Bohren nicht abrutschen, sollten Sie die Oberfläche mit Klebestreifen abdecken und dann erst die Markierung anzeichnen. Mit einem dünnen Bohrer oder Vorstecher etwas einsenken, dann erst bohren: für die LEDs 4-mm-Löcher, für die Buchsen Lochdurchmesser von 8 mm.

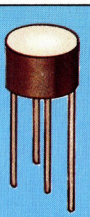
Bauteilliste

Diese Einzelteile brauchen Sie zum Bau des Buffer-Interfaces. Das Gehäuse darf auch andere Maße haben – allerdings müssen Sie dann sämtliche Werte entsprechend ändern.

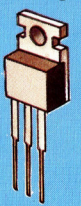
Anzahl	Bauteil
8	Widerstände, 4,7 kOhm, 0,5 Watt
8	Widerstände, 240 Ohm, 0,5 Watt
1	1 µF Elektrolyt-Kondensator
1	0,1 µF Kondensator
8	Leuchtdioden, rot
8	Dioden 1N4148
1	Brückengleichrichter, Typ W005
3	Sechsfach-TTL-Buffer, Typ 7407
1	5-Volt-Spannungsregler, Typ µA 7805
1	Lochplatine (Veroboard), 36 Reihen à 50 Löcher
3	DIL IC-Sockel, 14-polig
1	Rolle Elektronik-Lötzinn
1	1 Meter 12-poliges Flachbandkabel
8	schwarze 4-mm-Buchsen (Telefonbuchsen)
8	rote 4-mm-Buchsen (Telefonbuchsen)
8	schwarze 4-mm-Stecker (Bananenstekker)
8	rote 4-mm-Stecker (Bananenstecker)
1	Netzteilbuchse, 2,1 mm
1	12-poliger Minicon-Kontakt
1	Kunststoffgehäuse, 115 × 95 × 37 mm

Neue Teile

In der Liste tauchen viele bereits bekannte Einzelteile auf. Manche kennen Sie vielleicht noch nicht. Die Netzteilbuchse, der Gleichrichter und der Spannungsregler gehören zur Stromversorgung des Geräts. Als Trafo können Sie einen beliebigen Netztransformator mit einer Sekundärspannung zwischen 7 und 25 Volt verwenden. Der 12-polige Minicon-Kontakt dient zum Anschluß externer Geräte.

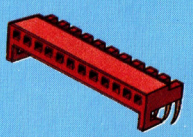


Brückengleichrichter
Der Gleichrichter wandelt die eingehende Wechselspannung in eine Gleichspannung um.



Festspannungsregler
Begrenzt und glättet die vom Brückengleichrichter abgegebene Gleichspannung auf eine konstante Spannung von genau 5 Volt.

Netzteilbuchse
Paßt zu den 2-mm-Netzteilsteckern vieler Computer (etwa Sinclair Spectrum). Läßt sich direkt auf die Platine auflöten.



Minicon-Kontakt
Läßt sich direkt auf die Platine löten. Der Anschluß externer Leitungen ist an Minicon-Sokkeln recht einfach.

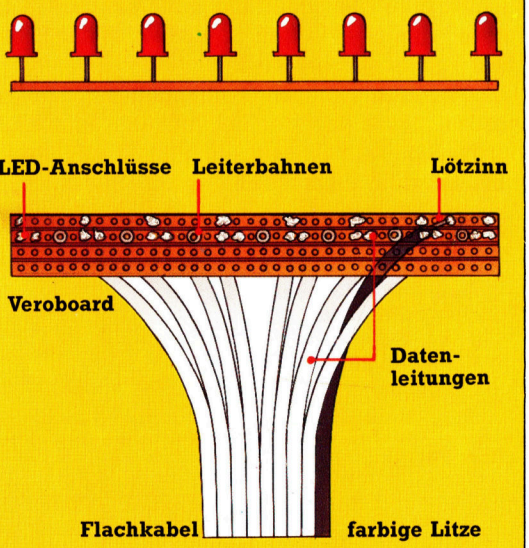
her oder gar der Heizungsanlage. Wegen der geringen Belastbarkeit des User Port ist für die Schaltung eine externe Spannungsquelle von neun Volt erforderlich. Die Schalteinheiten für einzelne Verbraucher werden vom Buffer über eine gemeinsame Busleitung gesteuert, welche die acht Datenleitungen, Masse und die 9-Volt-Spannungsversorgung vereinigt. Dadurch lassen sich die einzelnen Schalteinheiten auch gekoppelt betreiben. In der nächsten Folge sollen der Buffer getestet und die Reaktionszeit mit Hilfe von speziellen Programmen festgestellt werden.

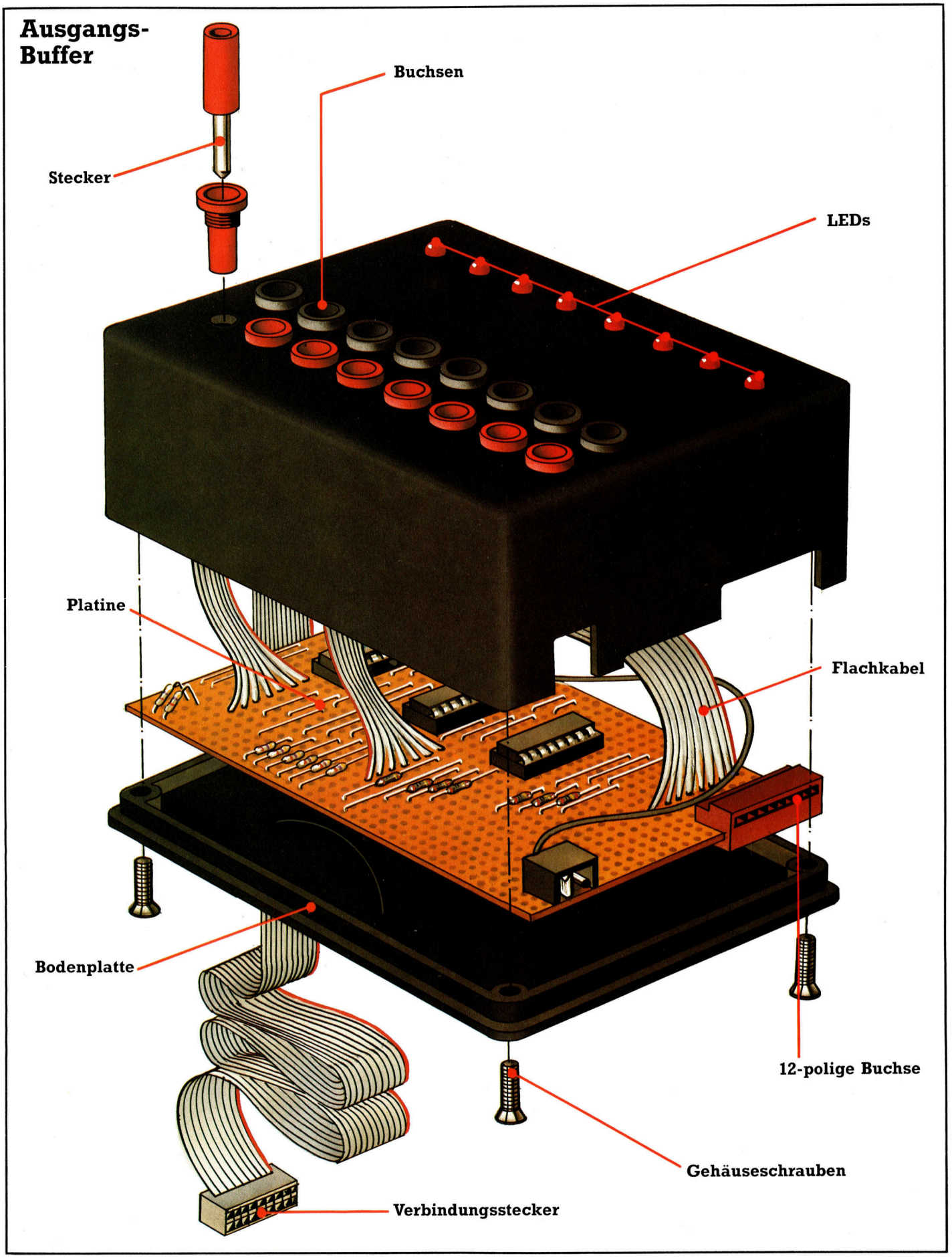
Montage der LED-Anzeige

Für die LEDs brauchen Sie ein vier Leiterbahnen breites Stück Veroboard, mit 36 Löchern pro Streifen. Stecken Sie die LEDs mit den längeren Anschlüssen an der Kante in die Leiterbahn, wobei der Abstand jeweils vier Löcher beträgt. So sollten sie in die vorgebohrten Gehäuselöcher passen. Wenn nicht, müssen die LEDs noch einmal versetzt werden. Löten Sie die Anschlüsse der LEDs auf den Leiterbahnen fest – achten Sie darauf, daß das Lötzinn nicht verläuft. Prüfen Sie den Widerstand zwischen zwei benachbarten Leiterbahnen mit dem Multimeter. Sollte er 0 sein, ist irgendwo Lötzinn zwischen zwei Bahnen geraten.

Von einem 20 cm langen, 12-poligen Kabel werden drei Litzen abgetrennt – jedoch nicht die farbig gekennzeichnete Litze, da diese an der allen LEDs gemeinsamen Leiterbahn festgelötet wird. Die anderen Drähte werden – jeweils einer neben einem LED-Beinchen – auf die andere Leiterbahn gelötet, die Sie nachher an sieben Stellen durchschneiden müssen, so daß jede Leitung nur Verbindung zu einem LED hat.

Die kleine Platine mit den LEDs kann nun vorsichtig an ihren Platz im Gehäuse geschoben und dort befestigt werden. Geschafft? Dann haben Sie es verdient, bis zum nächsten Teil unseres Kurses auszuruhen – als nächstes kommt nämlich die Montage der Hauptplatine dran!




Ausgangs-
Buffer
Buchsen
Stecker
LEDs
Platine
Flachkabel
Bodenplatte
12-polige Buchse
Gehäuseschrauben
Verbindungsstecker

Fakultäten

In dieser Folge wird demonstriert, wie man LOGO zur Berechnung von Fakultäten verwendet und wie Ergebnisse in „Produkt-Bäume" umgewandelt werden.

Wieviele Möglichkeiten gibt es, vier Personen auf vier Stühlen um einen Tisch zu plazieren? Die erste Person kann auf jedem der vier Stühle sitzen. Sobald sie aber Platz genommen hat, bleiben für die zweite Person nur noch drei Möglichkeiten, folglich nur noch zwei für die dritte und für die letzte nur noch eine. Die Gesamtzahl der verschiedenen Anordnungen beträgt deshalb: $4 \times 3 \times 2 \times 1$. Das wird gewöhnlich als 4! geschrieben und als „Fakultät 4" bezeichnet. Man findet bei mathematischen Problemen, die Reihen, Kombinationen und Wahrscheinlichkeiten darstellen, häufig Fakultäten.

Eine recursive Definition von Fakultäten zu schreiben ist einfach. Zunächst ist zu beachten, daß die Fakultät von 0 als 1 definiert ist. Die Fakultät jeder positiven Zahl, die nicht gleich Null ist – zum Beispiel X – ist die Fakultät aus X – 1 multipliziert mit X. In einem Programm sieht das so aus:

```
TO FAKULTAET :X
  IF :X = 0 THEN OUTPUT 1
  OUTPUT (FAKULTAET :X — 1) * :X
END
```

Zur Kontrolle können Sie PRINT FAKULTAET 6 eingeben. Das Ergebnis muß 720 sein.

Diese Prozedur arbeitet bis zur Zahl 12 korrekt. Größere Zahlen aber werden vom Computer nicht mehr ganzzahlig ausgegeben. Auf dem C 64 würde PRINT FAKULTAET 13 zum Beispiel 6.22702E9 ergeben, was 6.22702 mal 10^9 entspricht. Das ist unbefriedigend, da die letzten vier Ziffern abgeschnitten werden. Um diese trotzdem darstellen zu können, ist es erforderlich, die arithmetischen Fähigkeiten so zu erweitern, daß mit mehr als siebenstelliger Genauigkeit gerechnet werden kann.

Um das zu vereinfachen, betrachten wir nur positive ganze Zahlen. Diese werden als Listen dargestellt; so entspräche 1 234 567 der Darstellung [1 2 3 4 5 6 7]. Die beiden folgenden Routinen addieren solche Zahlen. Nach Eingabe von PRINT LANGADD [1 2 3] [5 6 9] sollte das Ergebnis [6 9 2] lauten:

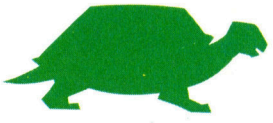
```
TO LANGADD :X :Y
  OUTPUT :LANGADD1 :X :Y 0
END
TO LANGADD :X :Y :UEBERTRAG
  IF (ALLOF (EMPTY? :X) (EMPTY?) :Y
    (:UEBERTRAG = 0)) THEN OUTPUT []
  TEST EMPTY ? :Y
  IFTRUE IF :UEBERTRAG = 0 THEN
    OUTPUT :X ELSE OUTPUT
    LANGADD1 :X [1] 0
  TEST EMPTY? :X
  IFTRUE IF :UEBERTRAG = 0 THEN
    OUTPUT :Y ELSE OUTPUT
    LANGADD1 [1] :Y 0
  MAKE "SUM (LAST :X) + (LAST :Y) +
    :UEBERTRAG
  OUTPUT LPUT REMAINDER :SUM 10
    LANGADD1
  BUTLAST :X BUTLAST :Y QUOTIENT:
    SUM 10
END
```

Dieses Programm entspricht einer normalen Addition, das heißt, es wird von links ausgehend addiert und jede Zahl der vorhergehenden Spalte eingerechnet.

Ähnlich ist es bei der Subtraktion, wobei jedoch eine Routine eingefügt wurde, mit der Nullen am Anfang eines Ergebnisses gelöscht werden. Resultate wie [0 0 0 7 8] gibt es also nicht.

```
TO LANGSUB :X :Y
  OUTPUT LOESCHNULL LANGSUB1
    :X :Y 0
END
TO LANGSUB1 :X :Y :NEHMEN
  IF (ALLOF (EMPTY? :X) (EMPTY? :Y)
    (:NEHMEN = 0)) THEN OUTPUT [0]
  TEST EMPTY? :Y
  IFTRUE IF :NEHMEN = 0 THEN OUTPUT
    :X ELSE OUTPUT LANGSUB1 :X [1] 0
  IF EMPTY? :X THEN PRINT [TUT MIR
    LEID, NEGATIVE ERGEBNISSE NICHT
    MOEGLICH] TOPLEVEL
  MAKE "DIFF (LAST :X) — (LAST :Y) —
    :NEHMEN
    IF :DIFF < 0 THEN
  OUTPUT LPUT (10 + :DIFF)
  LANGSUB1 BUTLAST :X BUTLAST :Y 1
  OUTPUT LPUT :DIFF LANGSUB1
    BUTLAST :X BUTLAST :Y 0
END
TO LOESCHNULL :X
  IF EMPTY? :X THEN OUTPUT [0]
  IF NOT ((FIRST :X) = 0) THEN
    OUTPUT :X
  OUTPUT LOESCHNULL BUTFIRST :X
END
```

Multiplikation in „Langform" ist etwas komplizierter. Angenommen, wir wollen 123 mit 338 malnehmen. Dafür wird die Aufgabe in drei Teile zerlegt: Zuerst multipliziert man 123 mit 8,



dann 123 mit 330 und addiert abschließend die beiden Ergebnisse. Man kann die zweite Rechnung in zwei Schritte zerlegen: Zunächst multipliziert man 123 mit 33 und setzt dann eine Null ans Ende des Resultats. Für die Multiplikation einer zweistelligen Zahl wird eine Recursion verwendet. Dies erfolgt mit der Prozedur LANGMULT:

```
TO LANGMULT :X :Y
  IF EMPTY? BUTLAST :Y THEN OUTPUT
    LANGMULT1 :X LAST :Y 0
  OUTPUT LANGADD (LANGMULT1 :X
    (LAST :Y) 0) (LPUT "0 LANGMULT
    :X BUTLAST :Y)
END
```

Die Multiplikation einer Zeile mit einer einzelnen Zahl wird von LANGMULT1 ausgeführt:

```
TO LANGMULT1 :X :NR :UEBERTRAG
  TEST EMPTY? :X
  IF TRUE IF :UEBERTRAG = 0 THEN
    OUTPUT [] ELSE OUTPUT (LIST
    :UEBERTRAG)
  MAKE "PROD (LAST :X) * :NR +
    :UEBERTRAG
OUTPUT LPUT REMAINDER :PROD 10
LANGMULT1 BUTLAST :X :NR
    QUOTIENT :PROD 10
END
```

Mit diesen Prozeduren sind die Grundlagen für präzise arithmetische Berechnungen geschaffen. Die einzige Beschränkung für die Zahlengröße ist der verfügbare Speicherplatz.

Unser ursprüngliches Fakultät-Programm kann nun in folgende Form gebracht werden:

```
TO FAKT :X
  IF FIRST :X = 0 THEN OUTPUT [1]
  OUTPUT LANGMULT (FAKT LANGSUB
  :X [1]) :X
END
```

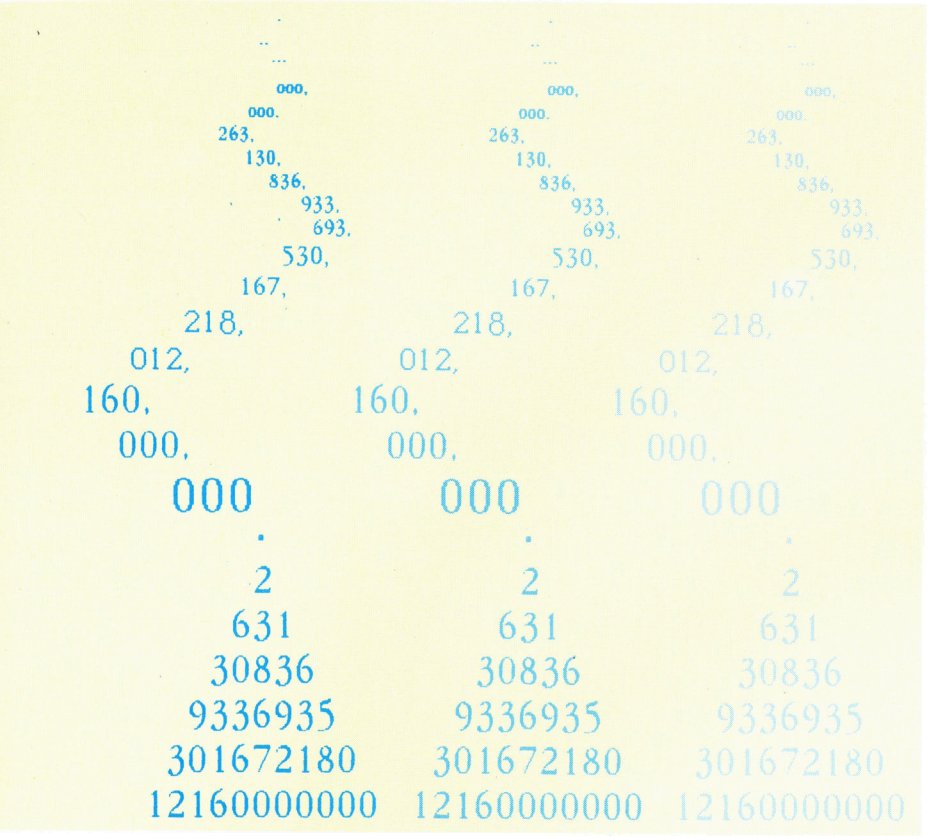
Bei der Eingabe FAKT [1 3] sollte das Ergebnis [6 2 2 7 0 2 0 8 0 0] lauten. Wie Sie sehen, dauert die Berechnung lange. Auf dem C 64 ist die größte Fakultät, die verarbeitet werden kann, 34!. Das Resultat hat 39 Stellen.

Die Darstellung großer Zahlen in Listenform sieht zwar ungewöhnlich aus, aber wir können das Programm so modifizieren, daß es die gewöhnliche Schreibweise in Listenform und umgekehrt übersetzt. Dazu verwenden wir die Prozeduren EXPLODE und IMPLODE.
EXPLODE 123 ergibt [1 2 3] und IMPLODE [1 2 3] ergibt 123.

```
TO EXPLODE :X
  IF EMPTY? :X THEN OUTPUT []
  OUTPUT (SENTENCE FIRST :X EXPLODE
    BUTFIRST :X)
END
TO IMPLODE :X
  IF EMPTY? :X THEN OUTPUT"
  OUTPUT (WORD FIRST :X IMPLODE
    BUTFIRST :X)
END
```

```
0!  1
1!  1
2!  2
3!  6
4!  24
5!  120
6!  720
7!  5.040
8!  40.320
9!  362.880
10! 3.628.800
11! 39.916.800
12! 479.001.600
13! 6.227.020.800
14! 87.178.291.200
15! 1.307.674.368.000
16! 20.922.789.888.000
17! 355.687.428.096.000
18! 6.402.373.705.728.000
19! 121.645.100.408.832.000
20! 2.432.902.008.176.640.000
16! 20.922.789.888.000
17! 355.687.428.096.000
18! 6.402.373.705.728.000
19! 121.645.100.408.832.000
20! 2.432.902.008.176.640.000
```

Fakultätenwerte wachsen erstaunlich schnell, wie man hier sehen kann. Da die Werte sehr lang werden, stellen die meisten Computer und Taschenrechner Fakultäten, die mehr als zwölf Stellen haben, in exponenter Schreibweise dar. Deshalb erfolgt die Darstellung der Fakultät von 12 als 4,79E8 oder $4{,}79 \times 10^8$.



Fakultätenbäume werden erzeugt, indem man die ganz links stehende Ziffer eines Fakultätenwertes an die Spitze des Baumes setzt. Nachfolgende Ziffern werden aus dem tatsächlichen Wert von links nach rechts in langsam wachsenden Gruppen herausgenommen. Die Gruppen werden darunter plaziert und symmetrisch geordnet. So bilden die Eckwerte eine Baumkontur. Das Diagramm zeigt die Fakultät von 32.

Hier wird deutlich, daß Zahlen in LOGO wie Worte behandelt werden. Wir können jetzt die Prozedur F definieren:

```
TO F :X
  PRINT IMPLODE FAKT EXPLODE :X
END
```

Damit wird die Fakultät von 13 nach Eingabe von F 13 berechnet.

Das Ergebnis (6227020800) ist in dieser Form etwas schwer zu lesen. Deshalb fügt man der besseren Lesbarkeit halber Kommas ein (6,227,020,800). Die folgenden Prozeduren gliedern das Wort in Gruppen von je drei Zahlen und fügen Kommas ein.

```
TO ADDKOMMAS :X
  IF ((COUNT :X) < 4) THEN OUTPUT :X
  OUTPUT (WORD ADDKOMMAS
    BUTTHREE :X ", LASTTHREE :X)
END
TO BUTTHREE :X
  OUTPUT BUTLAST BUTLAST BUTLAST :X
END
TO LASTTHREE :X
  OUTPUT (WORD (LAST BUTLAST
    BUTLAST :X) (LAST BUTLAST :X)
      (LAST: X))
END
```

Um diese Prozeduren einbeziehen zu können, muß auch F modifiziert werden:

```
TO F :X
  PRINT ADDKOMMAS IMPLODE FAKT
    EXPLODE :X
END
```

Verwendet man F, um die ersten zwanzig Fakultäten darzustellen, bekommt man eine Vorstellung, wie schnell diese an Länge zunehmen (die Resultate werden rechts dargestellt).

Nachdem wir die Fakultäten einiger Zahlen ermittelt haben, können wir beginnen, damit zu „spielen". Ein amerikanischer Mathematiker hatte einmal die Idee, große Fakultäten als Bäume auf Weihnachtskarten auszudrucken. Es gibt nicht viele Fakultäten, die die richtige Anzahl von Ziffern haben, um in Form von Bäumen ausgedruckt zu werden, doch die folgenden Prozeduren ermöglichen das:

Zahlenbäume

```
TO BAUM :L
  BAUM1 1 :L
END
TO BAUM1 :NR :L
  IF EMPTY? :L THEN STOP
  REPEAT ROUND (20 — :NR / 2) [PRINT1
    LEERZ] LINEPRINT :NR :L
  BAUM1 :NR + 2 SCHNEID :NR :L
END
TO LEERZ
  OUTPUT CHAR 32
END
TO LINEPRINT :NR :L
  IF :NR = 0 THEN PRINT "STOP
  PRINT1 FIRST :L
  LINEPRINT :NR — 1 BUTFIRST :L
END
TO SCHNEID :NR :L
  IF :NR = 0 THEN OUTPUT :L
  OUTPUT SCHNEID :NR — 1 BUTFIRST :L
END
```

Unsere Steuerprozedur muß wieder modifiziert werden:

```
TO F :X
  BAUM IMPLODE FAKT EXPLODE :X
END
```

Das Diagramm zeigt 32! als Baum geschrieben. Sind Sie daran interessiert, sich weiter mit diesen Fakultätenbäumen zu beschäftigen, sollten Sie wissen, daß es nur drei Zahlen unter 32 gibt, deren Fakultäten als Bäume dargestellt werden können. Die nächste größere passende Fakultätenreihe wäre 59!.

LOGO-Dialekte

Bei einigen LOGO-Versionen muß EMPTY? durch EMPTYP, ALLOF durch AND und PRINT1 durch TYPE ersetzt werden. Achten Sie auf die unterschiedliche Schreibweise der IF-Abfrage:

```
IF UEBERTRAG = 0 [OUTPUT [] ] [OUTPUT
  (LIST :UEBERTRAG)]
```

Statt QUOTIENT :X :Y verwendet man DIV :X :Y beim Spectrum und ROUND (:X / :Y) beim Atari.
Für SENTENCE wird beim Atari SE gesetzt.

Kleinroboter

Diesmal setzen wir uns mit Produkten auseinander, die unter der Bezeichnung „Roboter" im Angebot sind, und bewerten, ob sie die Erwartungen und die festgelegten Definitionen tatsächlich erfüllen.

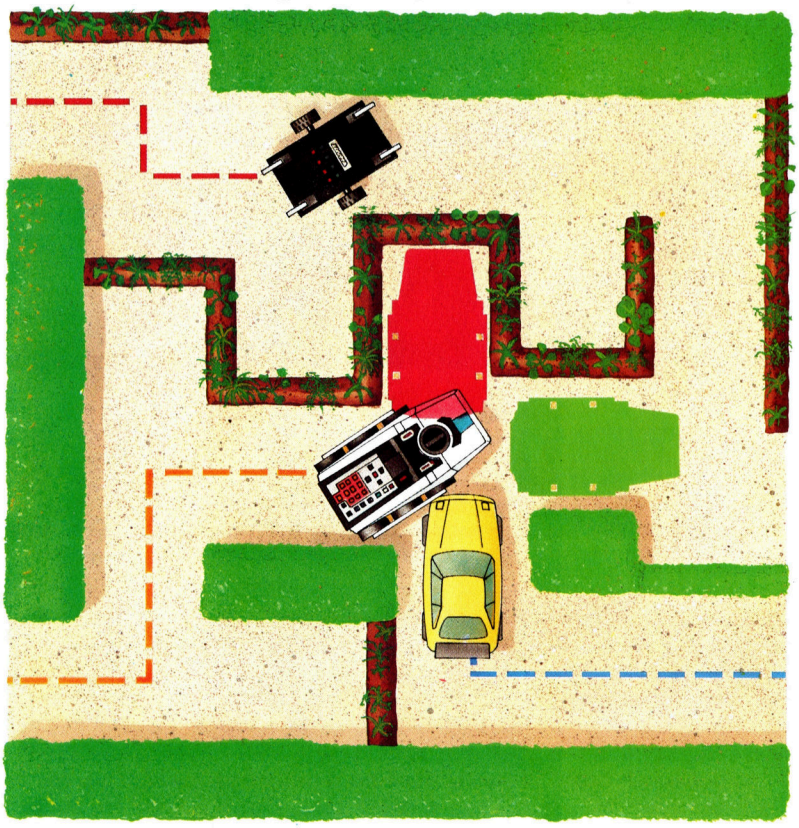
In der Praxis sind viele der bisher dargelegten Konzepte bei der Konstruktion von Robotern noch nicht realisiert worden oder konnten mangels geeigneter mechanischer Teile und/oder entsprechender intelligenter Software nicht verwirklicht werden. Derzeit vorhandene Roboter, ob nun für den Heim- oder Industriegebrauch gedacht, erfüllen bei weitem nicht die Anforderungen, die seit Jahren mit dem Begriff verbunden werden. Zwar gibt es Sensoren, mit deren Hilfe ein Roboter sehen, hören oder fühlen kann – doch alle diese Eindrücke haben für den Roboter keine Bedeutung und können nicht dahingehend umgesetzt werden, daß sie den Roboter zu eigenständigem, nichtprogrammiertem Verhalten bringen.

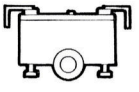
Dennoch gibt es eine Reihe von Produkten, die als „Roboter" verkauft werden. Die Bandbreite umfaßt kleine Spielzeuge, die knapp vier Mark kosten und reicht bis hin zu teuren R2D2-ähnlichen Konstruktionen und Industrieroboter. Nachdem die Komponenten der Roboterkonstruktion und die damit verbundenen Theorien dargelegt wurden, bleibt nun abzuwägen, was einen richtigen Roboter ausmacht.

Die erste Frage, die zugleich viele der niedrigpreisigen „Roboter" ausschließt, ist die nach Bewegung: Kann sich ein Roboter selbständig in einem bestimmten Raum bewegen? Man darf nicht erwarten, daß der Roboter sich selbst programmiert oder ohne menschliche Hilfe einen eigenen Bewegungskurs festlegt. Realistisch ist jedoch die Erwartung, daß sich ein Roboter – ist er erst einmal in Bewegung gesetzt – unabhängig von menschlicher Steuerung oder Kontrolle weiterbewegen und agieren kann.

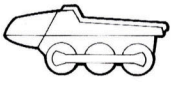
Nach diesem ersten Test sollte der Roboter nun dahingehend begutachtet werden, wie die Bewegung abläuft. Ein kleines Spielzeugauto kann zum Beispiel mit einem Motor und Batterien ausgestattet werden, die ihm Bewegung in gerader Richtung erlauben. Fügt man Stopper hinzu, ist das Fahrzeug imstande, Hindernissen

Roboter oder Spielzeug?





Our Robot
Angetrieben und gesteuert über einen großen Computer ist dieser Roboter mit berührungs- und lichtempfindlichen Sensoren ausgestattet.



Big Trak
LOGO-ähnliche Entfernungs- und Richtungsanweisungen können in das microprozessorgesteuerte Gerät mittels eigener Tastatur programmiert werden.

Bumper Car
Dieses batteriegetriebene Spielzeug fährt solange in einer Richtung, bis es gegen ein Hindernis stößt.

Die drei Geräte versuchen durch ein Labyrinth zu fahren: Das Spielzeug-Auto arbeitet sich von Wand zu Wand vorwärts. „Big Trak" folgt den Anweisungen des vom Menschen eingegebenen Programms, wogegen „Our Robot" das Labyrinth mittels seiner Software und Sensoren kennenlernt. Man darf sicher sein, daß Our Robot, gleich was geschieht, das Labyrinth bewältigt. Big Trak folgt seinem Programm und kann, richtige Anweisungen vorausgesetzt, die gestellte Aufgabe ebenfalls lösen. Das Spielzeugauto könnte nur „rechtweisende" Labyrinthe durchfahren. Es bleibt dennoch fraglich, ob es die Aufgabe lösen könnte.

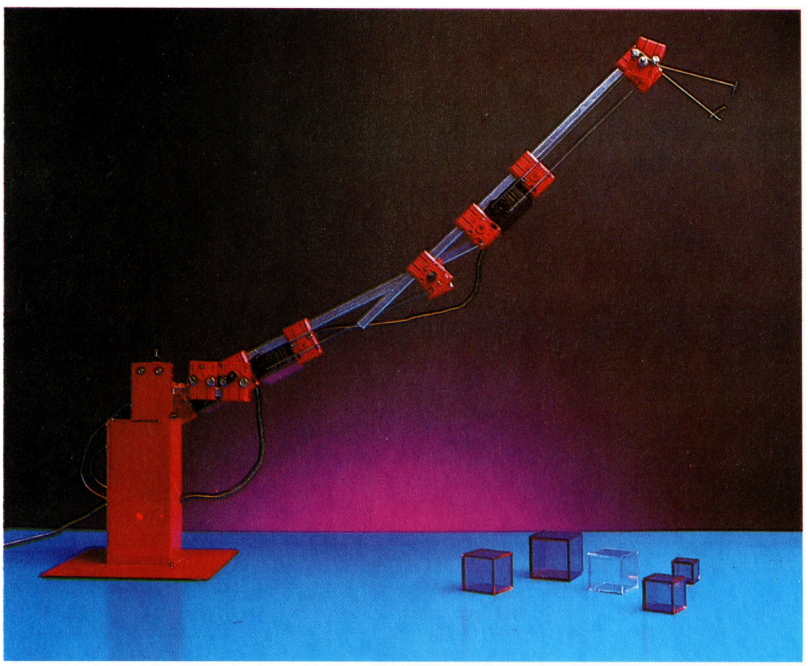
Stößt das Auto mit Big Trak zusammen, hat das auf das Fahrzeug keinen Einfluß. Big Trak dagegen wird um 90 Grad von seinem Weg (in Grün dargestellt), abgebracht, wendet weiter und fährt dann so (in Rot dargestellt), als befände er sich noch auf der ursprünglichen Strecke. Beide Geräte reagieren „unintelligent" auf dieses unverhergesehene Ereignis, wogegen der Roboter es als zusätzliche Information in einer unbekannten Umgebung betrachten würde.

wie Wänden oder Tischen auszuweichen. Durch sinnvolle Anordnung des Schwerpunkts und Ausstattung mit Gummireifen kann das Auto sogar eine Wand hochfahren, sich nach dem Umkippen wieder aufrichten und weiterfahren. Das Auto bewegt sich selbständig und unabhängig von menschlicher Kontrolle. Aber kann man es deshalb als „Roboter" bezeichnen? Die Antwort lautet eindeutig „Nein".

Steuerung durch Lochkarten

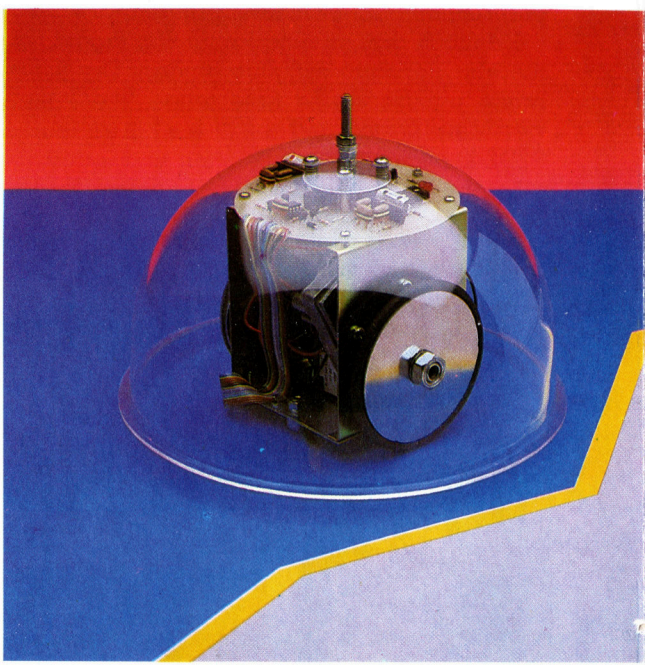
Wie dargelegt, läßt sich die Bewegung von Robotern in zwei Kategorien einteilen: in die einfache, programmgesteuerte und die intelligente. Das motorbetriebene Auto kann kein Roboter sein, weil es nicht für die Bewegung in verschiedene Richtungen programmiert wurde. Sein Aktionsraster ist zwar mechanisch integriert, doch es reagiert nicht auf menschliche Befehle, und verfügt über keine Möglichkeit, einen steuernden Menschen die Bewegung verändern zu lassen. Eine interessante Variante ist hierbei das motorgetriebene Fahrzeug, dessen Weg durch eine Lochkarte programmierbar wäre. Die Karte wird mechanisch gelesen. Das Auto folgt dem Muster, wendet nach links oder fährt geradeaus – entsprechend den vorgestanzten Anweisungen. Unserer Definition zufolge wäre dieses Auto ein Roboter, da es mittels Lochkarten programmierbar ist. Die Forderung nach programmierbarer Bewegung schließt zahlreiche kleine preiswerte Produkte von der allgemeinen Bezeichnung „Roboter" aus.

Natürlich müssen auch andere Gesichtspunkte berücksichtigt werden. Verfügt das Gerät über Sensoren, um eine Eingabe der „Außenwelt" zu ermöglichen? Kann es auf seine Umgebung reagieren und ein internes Kartenmodell entsprechend den Veränderungen erstellen? Spielt es gut Schach? All diese Kriterien wären bei der Entscheidung für einen Roboter zu bedenken. In der endgültigen Analyse aber ist die programmierbare Bewegung immer der entscheidende Faktor. Die folgenden Produkte, alle als „Roboter" bezeichnet, sind zu Preisen unter 1000 Mark erhältlich.



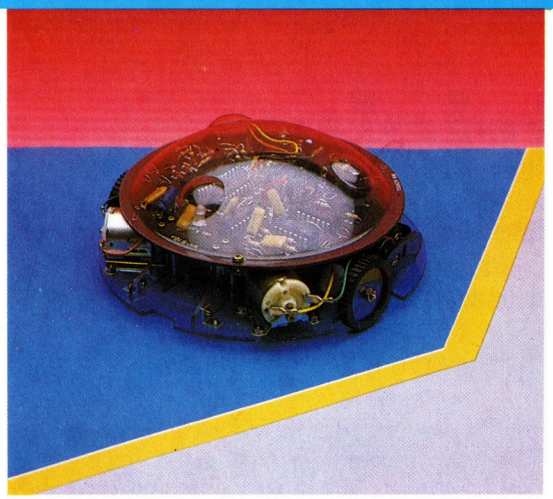
Name: Beasty
Art: Roboterarm
Programmierbar: Ja
Software-Unterstützung: Ja
Sensorische Rückmeldung? Position
Vertrieb: Commotion Ltd., 241 Green Street, Enfield, Middlesex EN3 7SJ

Beasty empfängt die Befehle über den User Port des Acorn B und wird über Hilfsstrom betrieben. Er wird als Bausatz geliefert, einschließlich dreier Servo-Motoren. Dazu gehören zwei Handbücher, eine Bauanleitung und eine Bedienungsanleitung. Beasty ist mit einem eigenen Betriebssystem ausgestattet – Robol –, mit dem der Benutzer jeden der drei Servomotoren unabhängig voneinander bewegen kann.



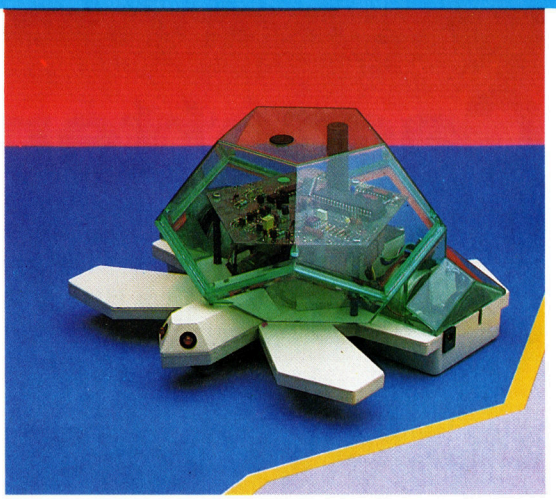
Name: Hebot II
Typ: Bodenroboter
Programmierbar: Ja
Software-Unterstützung: Ja
Sensorische Rückmeldung: Taster
Hersteller: Powertran Cybernetics Ltd., West Portway Industrial Estate, Andover, Hampshire, SP 10 3NN

Hebot II ist eine Roboter-Turtle, die durch zwei Gleichstrom-Motoren angetrieben wird, die mit den beiden Rädern verbunden sind. Über ein Verbindungskabel kann die Turtle mit dem Sinclair ZX 81 verbunden werden. Der Hersteller verweist aber darauf, daß durch andere Pin-Belegung der Hebot mit jedem Heimcomputer betrieben werden kann. Hebot II wird als Bausatz mit Handbuch geliefert.



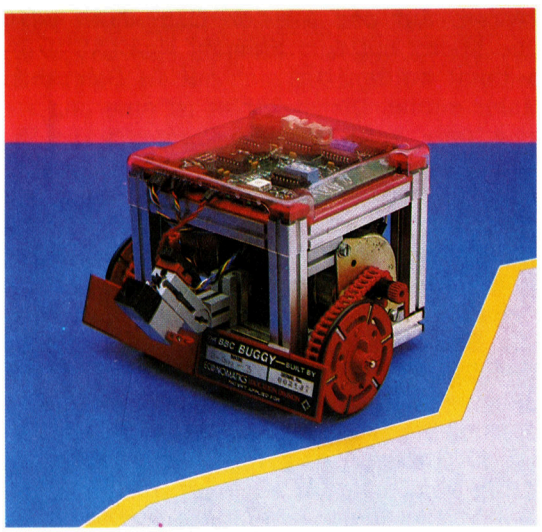
Name: Memocon
Typ: Bodenroboter
Programmierbar: Ja
Software-Unterstützung: Nein
Sensorische Rückmeldung: Nein
Vertrieb: Fachhandel

Der Memocon Crawler ist der vielseitigste Roboter aus der Movit-Reihe. Zum Antrieb der Gleichstrommotoren werden 1,5-Volt-Batterien verwendet. Die Steuerung erfolgt über eine Kommandotastatur, die über fünf Eingabetasten verfügt. Sie ist mit dem Crawler durch ein Kabel verbunden. Ferner ist das Gerät mit einem Summer und einer LED ausgestattet, die ebenfalls über Keypad aktiviert werden können.



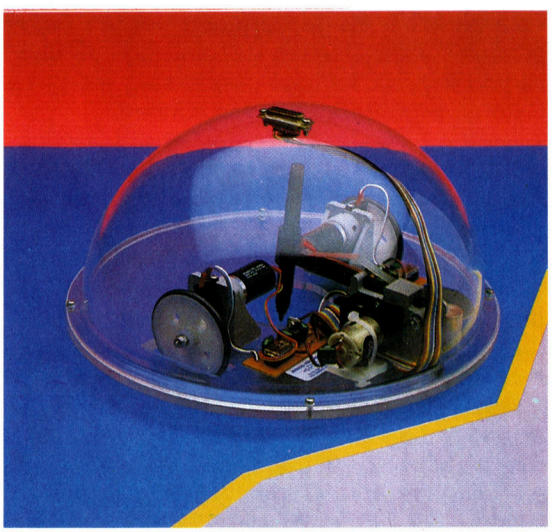
Name: Valiant Turtle
Typ: Bodenroboter
Programmierbar: Ja
Software-Unterstützung: Ja
Sensorische Rückmeldung: Position
Hersteller: Valiant Designs.

Die Valiant Turtle wird durch ein Motorenpaar angetrieben, von denen jeweils einer ein Rad bewegt. Das Gerät wird durch Infrarot über einen Computer gesteuert. Die dazugehörige Software läuft unter LOGO, kann allerdings auch ohne diese Kommandosprache betrieben werden. Die Stromversorgung erfolgt über einen wiederaufladbaren, ins Gerät integrierten Akkumulator.



Name: BBC Buggy
Typ: Bodenroboter
Programmierbar: Ja
Software-Unterstützung: Ja
Sensorische Rückmeldung: Lichtsensor, Drucksensoren
Hersteller: Economatics Education Ltd., Epic House, 9 Orgreave Road, Handsworth, Sheffield

BBC Buggy ist eine Software-gesteuerte Turtle. Buggy wird als Bausatz geliefert und über den User-Port des Acorn B gesteuert. Die Stromversorgung erfolgt über den Hilfsstromanschluß des Rechners. Buggy wird durch Schrittmotoren angetrieben. Der BBC Buggy reagiert auf und sucht nach Lichtquellen. Er kann ferner mit einem Zeichenstift und einem Strichcode-Leser ausgerüstet werden.



Name: Edinburgh Turtle
Typ: Bodenroboter
Programmierbar: Ja
Software-Unterstützung: Ja
Sensorische Rückmeldung: Position
Hersteller: Jessop Acoustics, Unit 5, 7 Long Street, London, E2

Die Edinburgh-Turtle wurde nach der Stadt benannt, in der man sie entwickelte. Sie ist durch ein Spiralkabel mit dem Computer verbunden, der sie auch mit Strom versorgt. Der Antrieb erfolgt durch zwei Gleichstrommotoren, die jeweils ein Rad antreiben. Sie ist außerdem mit einem einziehbaren Stifthalter und einem Lautsprecher ausgestattet. Unlängst brachte Jessop auch eine ferngesteuerte Version dieser Turtle auf den Markt.

Cassettentricks

Da die Einsatzmöglichkeiten verschiedener Dateistrukturen von dem jeweils verwendeten Computersystem abhängen, gehen wir in dieser Folge darauf ein, wie sich die betreffenden Techniken auf Geräten mit Cassettenbetrieb simulieren lassen.

Standardprogramme müssen oft geändert werden, wenn sie auf einer bestimmten Maschine funktionieren sollen. Der Grund dafür liegt in der verwirrenden Vielfalt von Dateisytemen. Prüfen Sie daher genau, wie sich die Möglichkeiten und Befehlsabläufe dieser Serie über den Umgang mit Dateien auf Ihrem Gerät einsetzen lassen. Am Beispiel eines Rechners mit Cassettenspeicher wollen wir untersuchen, wie sich die beschriebenen Dateistrukturen realisieren lassen.

Cassettensysteme können keine Dateien im Random-Access-Format verarbeiten, sondern müssen die Daten in der Reihenfolge abrufen, in der sie gespeichert wurden, das heißt, sequentiell. Sequentielle Dateiverwaltung bedeutet, daß Informationen, die aus einer Datei gelesen wurden, nach ihrer Bearbeitung in eine zweite Datei geschrieben werden. Dabei müssen beide Dateien gleichzeitig „eröffnet" sein, das heißt, es muß gleichzeitig auf beide Dateien zugegriffen werden können.

Automatische Speicher

Die meisten Heimcomputer mit ausschließlich sequentiellen Speichermöglichkeiten sind weiteren Einschränkungen unterworfen. So müssen Dateien vor ihrer Bearbeitung vollständig in den Speicher geladen werden. Sie lassen sich auch nur als Ganzes wieder auf die Cassette zurückschreiben, wobei der Speichervorgang entweder mehrmals während der Bearbeitung stattfinden kann oder nach Abschluß aller Änderungen. Weiter dürfen die Dateien die Größe des RAM-Bereiches nicht überschreiten, der nach dem Laden des Programms noch zur Verfügung steht.

Es gibt drei Methoden, Informationen auf Cassetten unterzubringen. Das einfachste System legt dabei keine separaten Dateien an, sondern speichert bei Eingabe von SAVE das Programm mit allen Variablen. Bei Anlage einer neuen Datei wird außer den Daten automatisch auch das Programm gespeichert, während beim Aufruf eines Programms die Variablen wieder mit den entsprechenden Daten versehen werden. Von Vorteil ist die Einfachheit dieser Methode – der Anwender sollte jedoch sicherheitshalber überprüfen, ob das komplette Programm korrekt geladen und gespeichert wurde.

Differenziertere Systeme verwenden ein BASIC, das einzelne Datentabellen ansprechen kann. Auf dem Oric Atmos schreibt der Befehl STORE A$,"NAME" die Matrix A$ auf die Cassette, während RECALL A$, "NAME" sie wieder in den Arbeitsspeicher lädt. Dabei wird die gesamte Matrix (A$ (1)), A$ (2) etc.) gespeichert, ohne daß STORE und RECALL die Größe der Matrix angeben müssen. Für diese Angaben wird automatisch der entsprechende DIM-Befehl des Programms gelesen.

Datensatzzähler

Dieses System gibt jedoch keine Auskunft über die Anzahl der Matrixeinträge. Daher muß die entsprechende Information vor dem Speichern in der Tabelle selbst untergebracht werden. Auf den meisten Computern kann eine Matrix mit dem Element Null (A$ (0,0)) definiert werden, das sich gut als Datensatzzähler eignet. Dieser Zähler ist eine numerische Variable (in unserem Programm die Variable R), die bei einer String-Matrix mit Befehlen wie A$ (0,0)=STR$(R) in einen String umgewandelt werden muß. Mit R = VAL(A$(0,0)) wird R beim Rückladen in den Arbeitsspeicher wieder auf Null gesetzt.

Nehmen wir als Beispiel eine zweidimensionale String-Matrix, die mit dem Befehl DIM A$(100,3) angelegt wurde. Die erste Zahl der Klammer bezeichnet die Datensatznummer, während die zweite Zahl auf eines der vier Felder des Datensatzes zeigt. Auf die Daten der Matrix kann nun wie auf eine Datei im Random-Access-Format zugegriffen werden.

Auf einigen Maschinen lassen sich Daten mit dem Befehl APPEND an das Ende einer sequentiellen Datei schreiben, ohne daß erst alle Daten gelesen und in eine neue Datei kopiert werden müssen. Einige Computer bieten auch die Möglichkeit, in sequentiellen Dateien eine Anzahl Felder zu überspringen und damit eine Art wahlfreien Zugriff aufzubauen.

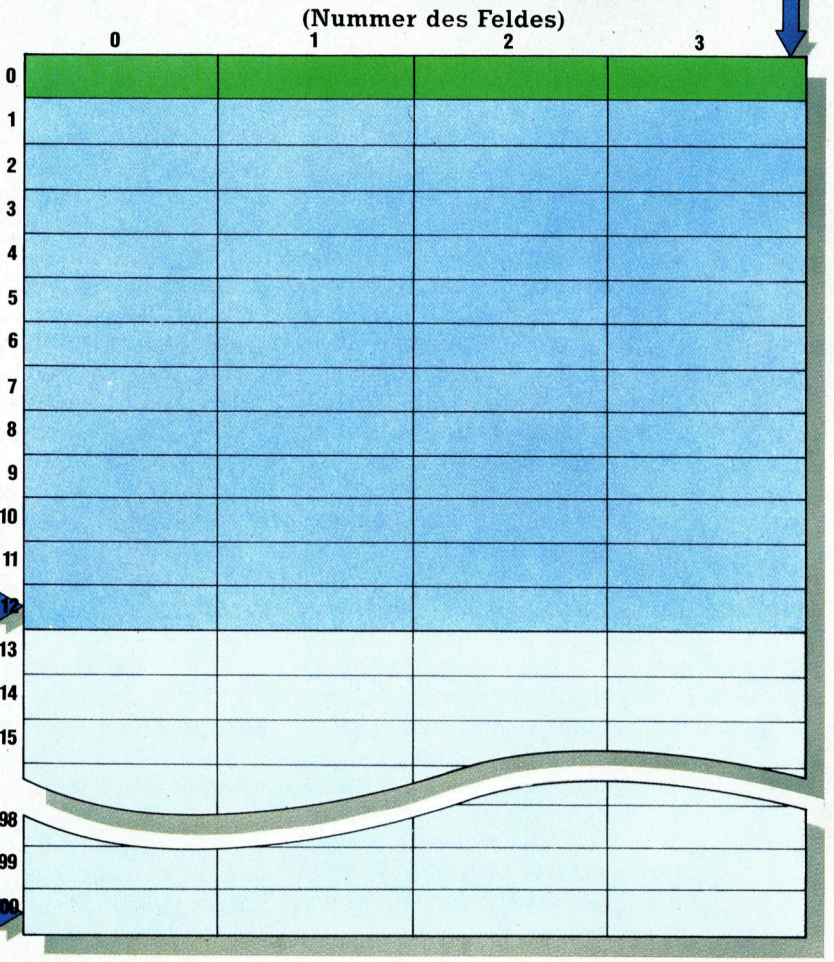
In dieser Serie haben wir versucht, die Grundlagen dieses komplexen Gebietes zu erläutern. Die erwähnten Strukturen lassen sich auf fast jeder Maschine verwirklichen, selbst wenn die Dateiverwaltung von Maschine zu Maschine unterschiedlich gehandhabt wird. Mit ein wenig Übung werden Sie dies mühelos in den Griff bekommen.

Datensätze und Felder in einer BASIC-Matrix

Mit einer zweidimensionalen String-Matrix läßt sich eine Random-Access-Datei simulieren. Dabei enthält das erste Teilelement der Klammer die Nummer des Datensatzes, während das zweite Element ein bestimmtes Datenfeld anzeigt.

Anzahl der Felder
Am Anfang des Programms sollte einer Variablen die Anzahl der Felder zugeordnet werden, da somit bei einer späteren Umstellung der Feldanzahl nur ein Befehl geändert werden muß.

Der Datensatz „Null"
In A$(0,0) wird die Anzahl der Datensätze untergebracht. Beim Speichern der Matrix auf Cassette wird dieser Wert automatisch mit abgelegt.



Anzahl der Datensätze
In einer Variablen wird die aktuelle Anzahl der Datensätze gespeichert, die in der Datei enthalten sind.

Maximale Zahl der Datensätze
Die Variable sollte die maximale Anzahl der Datensätze enthalten, die in der Datei vorkommen dürfen. Sie entspricht der im DIM-Befehl enthaltenen Zahl.

Löschen eines Datensatzes
Dieser Programmteil löscht den Datensatz N aus der Matrix. Alle Datensätze nach N werden dabei um eine Position vorbewegt, wobei die in N gespeicherten Daten überschrieben werden.

```
100  LET R = R + 1
110  IF R > M THEN  PRINT "ARRAY FULL": RETURN
120  FOR I = R TO N + 1 STEP  - 1
130  FOR J = 0 TO F
140  LET A$(I,J) = A$(I - 1,J)
150  NEXT J: NEXT I
170  LET A$(N,0) = N$: LET A$(N,1) = C$
180  LET A$(N,2) = D$: LET A$(N,3) = E$
190  RETURN
```

Einen Datensatz einfügen
Dieser Programmteil fügt an der Position N einen neuen Datensatz ein. Alle Datensätze hinter dieser Position werden um eine Position nach hinten bewegt. Dadurch entsteht eine Lücke für den neuen Datensatz.

```
200  FOR I = N TO R - 1
210  FOR J = 0 TO F
220  LET A$(I,J) = A$(I + 1,J)
230  NEXT J: NEXT I
240  LET R = R - 1
250  RETURN
```

Laden und Speichern einer Datenmatrix

Speichern: Programm mit Variablen
Der Spectrum speichert alle Variablen zusammen mit dem Programm (obwohl er eine Matrix auch einzeln ablegen kann). Unser Beispielprogramm füllt die Matrix mit Daten. Der SAVE-Befehl speichert Daten und Programm. Beim Laden des Programms beginnt die Ausführung automatisch bei Zeile 700, da der SAVE-Befehl dorthin verweist. Der Befehl RUN kann hier nicht eingesetzt werden, da er alle Variablen freisetzt.

Spectrum

```
100 REM***SPECTRUM DEMO***
200 DIM A$(100,20)
300 FOR K=1 TO 100
400 LET A$(K)="RecNo."+STR$(K)
500 NEXT K
600 STOP
700 PRINT "ARRAY CONTAINS..."
800 FOR k=1 TO 300
900 PRINT A$(K),
1000 NEXT K
1100 STOP
SAVE "DEMOPROG" LINE 700
LOAD "DEMOPROG"
```

Speichern: Einzelmatrix mit Bezeichnung
Mit den Befehlen STORE und RECALL kann der Oric einzelne Matrizen auf der Cassette ansprechen und vereinfacht den Speichervorgang.

Oric Atmos

```
100 REM Save array to tape
110 A$(0,0)=STR$(R)
120 PRINT "Please position tape,
press PLAY and  RECORD then hit RETURN"
130 A$=KEY$:IF A$="" THEN 130
140 STORE A$,"AFILE",5
150 PRINT "FINISHED"
160 RETURN

200 REM Load array from tape
210 PRINT "Please position tape
 and press PLAY then   RETURN"
220 A$=KEY$:IF A$="" THEN 220
230 RECALL A$,"AFILE",5
240 R=VAL(A$(0,0))
250 PRINT "FINISHED"
260 RETURN
```

Speichern: sequentielle Dateien
Der Acorn B ist einer der wenigen Heimcomputer, die die Cassettenspeicherung sequentieller Dateien unterstützen. Diese beiden Unterroutinen speichern und laden

Acorn B

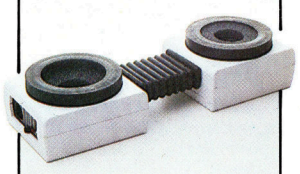
```
100 REM Save array to tape
105 PRINT "Please position tape"
110 X=OPENOUT("AFILE")
120 PRINT#X,R
130 FOR I= 1 TO R
140 FOR J= 1 TO F
150 PRINT#X,A$(I,J)
160 NEXT J: NEXT I
170 CLOSE#X
180 PRINT "FINISHED"
190 RETURN

200 REM Load array from tape
205 PRINT "Please position tape
and press PLAY then RETURN"
210 IF INKEY(0)=-1 THEN 210
215 X=OPENIN("AFILE")
220 INPUT#X,R
230 FOR I= 1 TO R
240 FOR J= 1 TO F
250 INPUT#X,A$(I,J)
260 NEXT J: NEXT I
270 CLOSE#X
280 PRINT "FINISHED"
290 RETURN
```

eine Matrix, indem sie eine serielle Datei aufbauen. Das erste Feld enthält die Anzahl der Datensätze.

Telekommunikation

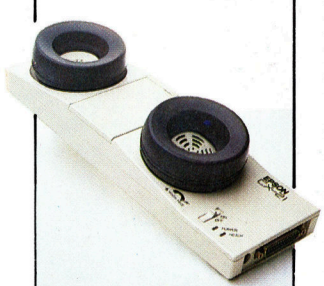
Nachdem wir uns mit der Kommunikation von Computern untereinander befaßt haben, wenden wir unsere Aufmerksamkeit nun der Wahl eines geeigneten Modems und entsprechender Software zu.



Protek 1200
Typ: Akustik-Koppler
Baud-Rate: 1200/75 und 1200/1200

Prism 1000
Typ: Fest installiertes Modem für Videotext
Baud-Rate: 1200/75
Auf Rechner abgestimmtes Komplettpaket

Commodore Communications System
Typ: Fest installiertes Modem für Videotext
Baud-Rate: 1200/75 und 1200/1200



Epson CX-21
Typ: Akustikkoppler
Baud-Rate: 300

Als Sie überlegten, für welches Computersystem Sie sich entscheiden sollten, hat man Ihnen vielleicht geraten, den Kauf von der Software, die darauf laufen soll, abhängig zu machen. Erst dann sollte die Hardware-Entscheidung fallen. Bis zu einem gewissen Grad hat diese Empfehlung auch bei der Auswahl eines Telekommunikationssystems Gültigkeit.

Zunächst ist zu überlegen, wofür das System verwendet werden soll. Für die Benutzung eines Videotext-Systems wie beispielsweise „Prestel" ist ein 1200/75-Baud-Modem erforderlich. Die meisten Datenbanken und elektronischen Mailboxen können mit einem 300-Baud-Modem angesprochen werden. Für Anwender-Anwender-Verbindungen ist wiederum eine Übertragungsrate von 1200 Baud empfehlenswert. Will man Compunet nutzen, benötigt man ein spezielles Compunet-Modem, da ein Teil der Software im ROM des Modems gespeichert ist. Außerdem sollte die Frequenz berücksichtigt werden. In England arbeitet man mit CCITT-Frequenzen, in den USA gilt der Bell-Standard. Will man also transatlantische Verbindungen nutzen, muß das Modem beide Frequenzen verarbeiten können.

Für die Anwender-Anwender-Verbindung ist empfehlenswert, ein umschaltbares Modem zu erwerben, das zwischen „originate"-(Übertragung bzw. Sendung) und „answer"-(Empfang)-Frequenzen im Modus verändert werden kann. Arbeitet das Modem nur im „originate"-Modus, muß das empfangende umschaltbar sein.

Wer häufig ein sogenanntes „Bulletin Board" (eine Art Datenbank) anruft, sollte ein Selbstwahl-Modem erwerben. Die meisten „Bulletin Boards" verfügen nur über eine einzige Telefonleitung und können jeweils nur mit einem Benutzer kommunizieren. Folglich sind sie häufig besetzt, und es ist sinnvoll, ein Modem zu verwenden, das das Wählen der Nummer automatisch wiederholt. Ein Selbstwähl-Modem sollte überdies mit Software ausgestattet sein, die Nummern empfangen – über Keyboard oder via Datenbank – und sie im richtige Format ans Modem weiterleiten kann. Verschiedene Selbstwähl-Modems senden die Nummern in unterschiedlichen Formaten. Daraus folgt, daß Modem und Software kompatibel sein müssen. Ein weiteres Argument dafür, Modem und Software als Paket zu kaufen.

Wer eine eigene Mailbox betreibt und auf der gleichen Leitung „normale" Gespräche führen möchte, sollte sich für ein automatisch antwortendes Modem (Auto-answer) entscheiden. Dabei sollten jedoch eventuelle Anrufer, die kein Modem haben, Bescheid wissen; denn das Modem pfeift ihnen einfach zehn Sekunden etwas vor und „hängt ein".

Auch für das automatisch antwortende Modem ist entsprechende Software nötig. Das Angebot reicht von Programmen, die bei jedem Anruf ein neues File eröffnen und es auf Diskette abspeichern, bis zu sehr ausgefeilten Programmen wie etwa das TBBS. Interessant für CP/M-Rechner ist die „auto-answer"-Software mit dem Namen „Remote CP/M". Damit ist es möglich, den CP/M-Rechner anzuwählen und jedes beliebige CP/M-Programm mittels Telefon laufen zu lassen. Das ist ideal für Benutzer, die einen normalen Rechner und ein Portable haben.

Bei der Suche nach geeigneter Software kommt man an Programmen, die ASCII-Daten senden und empfangen können, nicht vorbei. Natürlich ist es wichtig, daß das Programm mit Ihrem Rechner auch läuft. BASIC-Programme können im ASCII-Format übertragen werden, doch bei der Übertragung von Binär-Datenblöcken (-Files) (zum Beispiel CP/M .COM Files) ist ein binäres Übertragungsprotokoll erforderlich. XModem ist das verbreitetste.

Selbstwählsysteme

Komfortabel ist die Möglichkeit des „auto-log on"-Aufrufes für verschiedene Systeme. Ist man mit einem bestimmten System verbunden, muß lediglich das entsprechende File geladen werden, das die ID (Identifikationsnummer), das Paßwort und eventuell zusätzliche Informationen enthält. Einige Selbstwählsysteme verbinden diese Art File mit einer Datenbank, in der Telefonnummern gespeichert sind. Danach ist lediglich der Name des gewünschten Dienstes einzugeben. Die Software sucht nach der richtigen Rufnummer, wählt sie an und stellt automatisch die Verbindung her.

Haben Sie sich erst einmal entschieden, was Sie benötigen, müssen ein geeignetes Modem und entsprechende Software gefunden werden, die diese Leistung erbringen. Man kann zwar Modem und Software separat kaufen, besser ist es, einem Fachhändler eine Liste mit den gewünschten Besonderheiten zu geben, dazu Details über den entsprechenden Computer, und ihm die Zusammenstellung des Gesamtpakets zu überlassen. Sollte das System

dann nicht wie erwartet funktionieren, können Sie sich wiederum an den Händler wenden.

Wer bereits einen Computer besitzt, wird ihn wahrscheinlich als Terminal benutzen wollen. Das ist mit jedem Rechner möglich – selbst ein ZX 81 kann dazu verwendet werden. Allerdings ist die Eignung der Heimcomputer unterschiedlich. Nachstehend bringen wir eine kurze Zusammenfassung der Fähigkeiten von vier Rechnern.

Am einfachsten ist der Acorn B in ein Terminal zu „verwandeln". Mit wenigen BASIC-Zeilen kann man sein eigenes Terminal-Programm schreiben:

```
100 REM Acorn B Terminal-Programm
110 *FX2,2
120 *FX3,1
130 REPEAT: GET A$: IF$=CHR$(13)
    THEN PRINT
140 PRINT A$; :UNTIL FALSE
```

Für den Acorn B werden mehrere gute Terminal-Programme angeboten, von denen einige in ROM-Form geliefert werden. Diese sind aber relativ teuer. Da das Betriebssystem des Acorn B fast die ganze Arbeit erledigt und die Software sehr umfangreich ist, braucht der Programmierer nur wenig selbst zu tun.

Weit schwerer ist der Einsatz des Spectrums als Terminal. In BASIC ist eine Telekommunikation nicht möglich. Mit einem BASIC-Programm kann der Spectrum lediglich auf 10 Baud gebracht werden. Somit ist beispielsweise das Speichern der Zeichen im RAM unmöglich. Bis vor kurzem war auch das Interface 1 nicht anwendbar. Da es sich dabei um keine RS232-Schnittstelle handelt, konnte man es für Telekommunikation nicht verwenden. In Deutschland wird jetzt allerdings von Hansesoft in Hamburg das TEKOS-System angeboten, das mit dem Interface 1 arbeitet.

Auch der Commodore 64 ist mit einem nicht standardisierten seriellen Interface ausgestattet. Die meisten Modems lassen sich an den User-Port des Rechners anschließen. In BASIC läßt sich auch hier nur wenig Nützliches machen. Da der C 64 über einen nicht standardisierten ASCII-Zeichensatz verfügt, muß die Software den C-64-ASCII-Code in Standard-ASCII übersetzen. Ein empfehlenswertes Tele-Kommunikationsprogramm für den C-64 ist „Terminal 64".

Tandy bietet Terminal-Software für seine mit Diskette arbeitenden TRS-80-Rechner, die mit fast allen Betriebssystemen lauffähig sind. Sie ist eigentlich für die Kommunikation Maschine-Maschine angelegt, kann aber auch mit Modems betrieben werden. Tandy-Microcomputer waren übrigens die ersten Rechner, für die es Mailboxen gab. Deshalb gibt es eine ganze Reihe von Programmen auf Disketten und Cassetten. Allerdings kann Videotext wegen der Übertragungsrate von 1200/75 Baud auf den Rechnern der TRS-80-Reihe von Tandy-Computer nicht betrieben werden.

Das breite Angebot von Hardware und Software macht den Kauf eines Modems recht kompliziert. Die hier gezeigte Einheit vereint die Vorzüge mehrerer Modems. Der Anfänger sollte überlegen, was er bei der Telekommunikation eigentlich braucht, und es dann dem Händler überlassen, ein entsprechendes Paket zusammenzustellen.

Größere Personalcomputer können dagegen fast ohne Ausnahme für die Datenfernübertragung eingesetzt werden. Einige wichtige Punkte sind dabei allerdings zu berücksichtigen. Es gibt eine wachsende Tendenz in Richtung eingebauter Modems. Bei Geräten, die mit Telekommunikations-Software auf ROM-Basis ausgestattet sind, ist die Benutzung denkbar einfach. Sie müssen lediglich ans Telefonnetz angeschlossen werden.

Die zweitbeste Möglichkeit ist ein Rechner, der über mindestens zwei RS232-Ports verfügt, womit gleichzeitig der Betrieb eines Modems und eines seriellen Druckers möglich ist. Einige Rechner sind mit separaten, nicht standardisierten Modem-Ports ausgestattet. Ein passendes Kabel vorausgesetzt, läßt sich das ausgesuchte Modem anschließen.

Bei der Software gibt es keine Probleme, wenn man einen Rechner mit CP/M, MS-DOS oder PC-DOS als Betriebssystem hat. Nicht standardisierte Betriebssysteme können jedoch, wie jede andere Software auch, mit Übertragungsprogrammen lauffähig gemacht werden.

Universell einsetzbar

Maschinencodemodule lassen sich für die unterschiedlichsten Programme einsetzen, wenn sie statt absoluter Adressen und fester Werte Symbole und Lables enthalten und dadurch „frei verschiebbar" (relocatable) werden. In dieser Folge stellen wir weitere Assemblerbefehle vor und werfen einen ersten Blick auf die Methoden, mit denen die Assemblersprache Unterroutinen aufruft.

Viele Grundfunktionen, die im Befehlssatz einer Hochsprache als selbstverständlich angesehen werden, sind im Assembler nicht von vornherein vorhanden. Da die Assemblersprache hauptsächlich „primitive" Anweisungen enthält, die die „einfach denkende" CPU verstehen kann, müssen Routinen für grundlegende Aufgaben, wie Ein- und Ausgabesteuerung oder Zwei-Byte-Arithmetik, bei jeder Programmierung neu angelegt werden. Es bietet sich an, die am häufigsten eingesetzten Routinen auf Papier, Band oder Diskette zu sammeln, um sie bei Bedarf in neue Programme einbinden zu können.

Hier gibt es jedoch zwei grundlegende Probleme. Zunächst müssen wichtige Routinen so aufgebaut sein, daß sie ohne Änderung in den unterschiedlichsten Programmen funktionieren können. Diese Module dürfen auch nicht an bestimmte Speicheradressen gebunden sein, sondern müssen bei einer Neuassemblierung unter einer anderen ORG-Adresse genau die gleichen Aufgaben erfüllen wie an ihrer ursprünglichen Speicheradresse.

Beide Probleme sind auch dem BASIC-Programmierer vertraut, der seine Programme auf mehreren Maschinentypen einsetzen will. Die Lösung der Assemblersprache sieht der BASIC-Lösung ähnlich: Variablen übertragen die Werte des Hauptprogramms auf die Unterroutinen, die durch die Verwendung eigener (lokaler) Variablen unabhängig sind. Außerdem wird auf die Angabe von festen Werten (numerische und String-Konstanten) und auf Zeilennummern verzichtet.

Im bisherigen Verlauf des Kurses haben wir die Inhalte der Speicherstellen ähnlich wie BASIC-Variablen behandelt. Aus Gründen der Einfachheit wurden die Bytes dabei fast ausschließlich mit absoluten Adressen eingesetzt. Soll eine Routine jedoch universell einsetzbar sein, müssen statt absoluter Adressen und fester Werte Symbole verwandt werden, die die Pseudo-Op-codes des Assemblers als Variablen und als Zeilennummern des Programms anbieten. Hier ein Beispiel für beide Anwendungen:

6502			Z80		
DATA1	EQU	$12	DATA1	EQU	$12
DATA2	EQU	$79	DATA2	EQU	$79
	LDA	DATA1	LD	A,(DATA1)	
LOOP	ADC	DATA2	ADC	A,(DATA2)	
	BNE	LOOP	JR	NZ,LOOP	
	RTS		RET		

Die Tabelle enthält zwei Symboltypen: zwei Werte und ein Label. Alle Symbole sind Operanden von Assemblerbefehlen. Das Programm kann somit leicht modifiziert werden. Die einzigen absoluten Werte sind DATA1 und DATA2.

Außer den bereits erwähnten Pseudo-Opcodes gibt es noch DB, DW und DS (die Bezeichnungen können wie bei ORG und EQU von Assembler zu Assembler verschieden sein). Diese drei Anweisungen bedeuten „Definiere Byte", „Definiere Wort" und „Definiere Speicher".

			ORG	$D3A0
D3A0	5F	LABL1	DB	$5F
D3A1	CE98	LABL2	DW	$98CE
D3B3		LABL3	DS	$10
D3B3		DATA1	EQU	LABL3

Symboltabelle:
LABL1=D3A0 : LABL2 = D3A1 : LABL3 = D3A3
DATA1 = D3A3
ASSEMBLIERUNG BEENDET — KEINE FEHLER

In diesem vollständigen Assemblerlisting (die Anzeige eines Assemblerprogramms) taucht zum ersten Mal eine Symboltabelle auf, die die im Programm eingesetzten Symbole und ihre entsprechenden Werte enthält. Die neuen Abläufe haben folgende Aufgaben: Die Zeile, die mit LABL1 anfängt, enthält den Pseudo-Op DB. Aus dem Programmlisting läßt sich ersehen, daß LABL1 durch die ORG-Anweisung die Adresse $D3A0 zugeordnet wird. DB setzt den Wert $5F auf das von LABL1 adressierte Byte. Die Op-codespalte zeigt, daß die Speicherstelle $D3A0 mit $5F initialisiert wurde.

LABL2 stellt die Adresse $D3A1 dar. DW initialisiert jedoch ein „Wort" (zwei aufeinander folgende Bytes) und bringt damit den Wert $98CE in den Speicheradressen $D3A1 und $D3A2 im lo-hi-Format unter. Da DW seine Operanden automatisch in das lo-hi-Format umwandelt, wird er oft für die Initialisierung

von Zeigeradressen eingesetzt. LABL2 oder die Speicherstelle $D3A1 könnte eine derartige Adresse sein, die auf $98CE zeigt.

Der Befehl DS $10 addiert $10 auf den Programmzähler. Dies ist in der Symboltabelle leichter zu erkennen als im Programmlisting: LABL3 stellt die Adresse $D3A3 (die Adresse, die dem vorherigen Befehl folgt) dar, obwohl dieser Wert aufgrund der Liste $D3B3 zu sein scheint. $D3B3 ist jedoch die Speicheradresse des nächsten Befehls. Damit wird deutlich, daß DS $10 einen Block von 16 Bytes (von $D3A3 bis einschließlich $D3B2) zwischen sich und dem nächsten Befehl reserviert.

Bei dem letzten Befehl des Listings setzt EQU ein Symbol dem Wert eines anderen gleich. DATA1 entspricht daher $D3A3 (dem Wert von LABL3). Auch dieser Ablauf kann zunächst etwas verwirren. LABL3 ist die symbolische Darstellung der Adresse $D3A3. DATA1 EQU LABL3 bedeutet also „das Symbol DATA1 soll die gleiche Bedeutung und den gleichen Wert wie das Symbol LABL3 erhalten". Die Tatsache, daß DB den Inhalt von $D3A3 auf $5F gesetzt hat, hat keinen Einfluß auf die Bedeutung der Symbole LABL3 und DATA1.

Pseudo-Befehle

Auf den ersten Blick scheint DB der Anweisung EQU zu entsprechen. Dies ist jedoch nicht der Fall. LABL1 bedeutet: „die Speicheradresse $D3A0", während DB $5F dieses Byte mit dem Wert $5F initialisiert. Obwohl der Wert von LABL1 festgelegt ist, kann der Inhalt der Adresse, die LABL1 darstellt, jederzeit geändert werden (zum Beispiel durch Speichern des Akkumulatorinhalts später im Programm). DATA1 dagegen ist ein Symbol, dessen Wert von der Anweisung EQU festgelegt wurde und durch das Programm nicht mehr zu ändern ist. LABL3 zeigt nun wiederum auf den Anfang eines Datenbereiches mit 16 Bytes, deren Inhalt das Programm ändern kann. Die Adresse von LABL3 ist jedoch fixiert.

Damit sind die Möglichkeiten der neuen Pseudo-Befehle aber noch nicht erschöpft. Sehen Sie sich folgende neue Version des vorigen Programmteils an:

```
                        ORG   $D3A0
D3A0   4D4553   LABL1   DB    'MESSAGE 1'
D3A9   CE98     LABL2   DW    $98CE
D3BB            LABL3   DS    $10
D3BB            DATA1   EQU   LABL3

Symboltabelle:
LABL1 = D3A0: LABL2 = D3A9: LABL3 = D3AB
DATA1 = D3AB
ASSEMBLIERUNG BEENDET — KEINE FEHLER
```

Operand der DB-Anweisung ist der String „MESSAGE1". Der Assembler hat die Speicherstellen von $D3A0 bis $D3A8 mit den ASCII-Werten der von Anführungsstrichen eingeschlossenen Zeichen initialisiert. Die Spalte der Speicheradressen macht dies deutlich: Der Inhalt der drei Bytes von $D3A0 bis $D3A2 wird mit $4D, $45 und $53 angegeben – den Hexzahlen der ASCII-Codes für die Buchstaben „M", „E" und „S".

Diese Einrichtung befreit den Programmierer nicht nur von der mühsamen Übersetzung von Zeichen und Meldungen in Folgen von ASCII-Codes, sondern macht auch das Programm leichter lesbar. Das Modul deutet außerdem an, daß Assemblerprogramme auch Bildschirmanzeigen enthalten können und Ergebnisse nicht immer im Speicher abgelegt werden müssen, damit sie dort mit dem Monitorprogramm betrachtet werden können. Im weiteren Verlauf dieser Serie werden wir auch die Anzeigemöglichkeiten der Assemblersprache behandeln. Bevor wir uns aber diesem Bereich zuwenden, muß auf einige andere Abläufe eingegangen werden. Da Sie jedoch wissen, daß aus Speichertabellen aufgebaute Bildschirmanzeigen nichts anderes sind als bestimmte Speicherbereiche, können Sie auch schon den Bildschirm adressieren.

Das Wichtigste der neuen DB-Anweisung ist jedoch, daß LABL1 damit der Status einer BASIC-String-Variablen zugeordnet wird. Die BASIC-Anweisung

```
200 LET A$="MESSAGE1"
```

richtet eigentlich einen Zeiger auf den Anfang einer Bytetabelle ein, die die ASCII-Codes für

Übungen

1) Im ersten Programmteil des Haupttextes reserviert der Pseudo-Op-code DS $10 Speicherbytes von der Adresse an, die durch LABL1 dargestellt wird. Schreiben Sie ein Assemblerprogramm, das die Zahlen $0F bis $00 in diesem Block in absteigender Reihenfolge speichert – eine Zahl pro Byte. Eine Schleife mit indizierter Adressierung sollte die Aufgabe ausführen. Zu diesem Zweck benötigen Sie den Befehl DEX (Register X verringern) oder DEC (IX+0) (IX erweitern). Die Schleife sollte ausgeführt werden, solange das Indexregister nicht das Setzen des Nullflag auslöst. Verwenden Sie daher die Verzweigungsbefehle BNE oder JR NZ.

2) Schreiben sie nach der Methode der ersten Übung ein Programm, das eine Meldung kopiert, die von dem Pseudo-Op-code unter LABL1 abgelegt wurde (siehe zweiter Programmteil des Haupttextes). Die Kopie soll in einen Speicherblock gelegt werden, dessen Anfangsadresse von dem Pseudo-Op-code in LABL2 gespeichert wurde. Da die Adresse $98CE sich für Ihren Computer nicht eignen mag, muß hier eventuell die Initialisierung geändert werden. Das Programm sollte für alle Adressen und für Meldungen jeder Länge funktionieren. Verwenden Sie daher die Zeichenzahl als Schleifenzähler, oder bauen Sie in das Programm eine Endabfrage der Meldung ein – zum Beispiel durch einen Stern als letztes Zeichen jeder Meldung.

„M", „E", „S" enthält. Findet der BASIC-Interpreter einen Bezug auf A$, dann sucht er in seiner Symboltabelle nach dem Zeiger der entsprechenden Speicheradresse, das heißt nach der Anfangsadresse des Inhalts von A$. In unserem Assemblerprogramm können wir LABL1 wie A$ behandeln, vorausgesetzt, wir haben ein Modul, das mit der Methode der indizierten Adressierung eine Speichertabelle bearbeiten kann.

Maschinencode für GOSUB

Durch Pseudo-Op-codes können wir absolute Adressen und feste Werte durch Symbole ersetzen und damit die meisten Schwierigkeiten für den universellen Einsatz unserer Programme beseitigen. Wir müssen diese frei verschiebbaren Module nun nur noch von unserem Hauptprogramm aus adressieren können, das heißt, wir brauchen einen Maschinencodebefehl, der dem GOSUB des BASIC entspricht.

Die Befehle JSR und CALL führen diesen Sprung für den 6502 und den Z80 aus. Beide Befehle benötigen eine absolute Adresse (die ein Label sein kann) als Operanden, und beide ersetzen den Inhalt des Programmzählers durch die Adresse, die ihr Operand angibt. Der nächste Befehl, der zur Ausführung kommt, ist dann die erste Anweisung der adressierten Unterroutine. Die Programmausführung setzt sich solange fort, bis sie einen Rücksprungbefehl findet (RTS oder RET). Der Rücksprung ersetzt den augenblicklichen Inhalt des Programmzählers durch die Adresse, die vor Ausführung des JSR- oder CALL-Befehls dort vorhanden war. Der nächste Befehl, der ausgeführt wird, ist daher die Anweisung, die JSR oder CALL unmittelbar folgt. Auch der BASIC-Interpreter benutzt diesen Mechanismus bei der Ausführung und der Rückkehr von einem GOSUB-Befehl. Hierbei entsteht jedoch die Frage, wie der ursprüngliche Inhalt des Programmzählers beim Aufruf des Rücksprungs wiederhergestellt werden kann. Die Antwort ist einfach: Die Befehle JSR und CALL schieben (push) den Inhalt des Programmzählers auf den Stapel (stack), bevor sie ihn durch die Adresse der Unterroutine ersetzen. RTS oder RET ziehen diese Adresse vom Stapel wieder herunter (pop) und setzen sie in den Programmzähler ein. Was ein Stapel ist, welche Funktion er ausübt und wie sich Werte dort ablegen oder herunterziehen lassen, werden wir in der nächsten Folge behandeln. Anhand von weiteren Übungen können Sie danach Ihre Kenntnisse überprüfen.

Befehlssatz

BEQ – **Verzweigen bei Null** — Relativ – FO (2 Bytes) — **6502**

Wenn das Nullflag gesetzt ist, wird der Programmzähler um den Wert des Bytes erhöht, das dem Op-code unmittelbar folgt.

Beeinflußt PSR-Register

	S	V		B	D	I	Z	C
MSB								LSB

KEINE AUSWIRKUNG

Speicheradresse	Maschinencode	Beispiel: Assemblerbefehle
8F00	FO 16	BEQ $16

	Vorher		Nachher
Programmzähler	02	lo	18
	8F	hi	8F

Programmspeicher: FO, 16 — $8F00

JR Z – **Verzweigen bei Null** — Relativ – 28 (2 Bytes) — **Z80**

Wenn das Nullflag gesetzt ist, wird der Programmzähler um den Wert des Bytes erhöht, das dem Op-code unmittelbar folgt.

Beeinflußt PSR-Register

	S	Z		H		V	N	C
MSB								LSB

KEINE AUSWIRKUNG

Speicheradresse	Maschinencode	Beispiel: Assemblerbefehle
8F00	28 16	JR Z,$16

	Vorher		Nachher
Programmzähler	02	lo	18
	8F	hi	8F

Programmspeicher: 28, 16 — $8F00

INX – **Register X erweitern** — implizit – F8 (1 Byte) — **6502**

Der Inhalt des Registers X wird um 1 erhöht.

Beeinflußt PSR-Register

	S	V		B	D	I	Z	C
MSB	X						X	LSB

Speicheradresse	Maschinencode	Beispiel: Assemblerbefehle
F391	E8	INX

	Vorher	Nachher
PSR	????????	0??????1
X	FF	00

Programmspeicher: E8 — $F391

INC IX – **Register IX erweitern** — implizit – DD 23 (2 Bytes) — **Z80**

Der Inhalt des Registers IX wird um 1 erhöht.

Beeinflußt PSR-Register

	S	Z		H		V	N	C
MSB								LSB

KEINE AUSWIRKUNG

Speicheradresse	Maschinencode	Beispiel: Assemblerbefehle
F391	DD23	INC IX

	Vorher		Nachher
Programmzähler	FF	lo	00
	E7	hi	E8

Programmspeicher: DD, 23 — $F391

Mit etwas Mut

Der vielleicht interessanteste aller japanischen Computer-Hersteller ist zugleich auch einer der unbekanntesten: SORD. Die vertrauten großen Namen von Hitachi bis Sony sind riesige Unternehmen, die Tausende von Leuten beschäftigen und über enorme Mittel verfügen. SORD dagegen ist ein kleiner Betrieb, in dem nur wenige hundert Menschen arbeiten.

Der Name „SORD" wurde aus der Kombination von SOftware und HaRDware entwikkelt. Ein passender Name, da das Unternehmen der Entwicklung von Software stets ebensoviel Aufmerksamkeit schenkte wie der von Hardware.

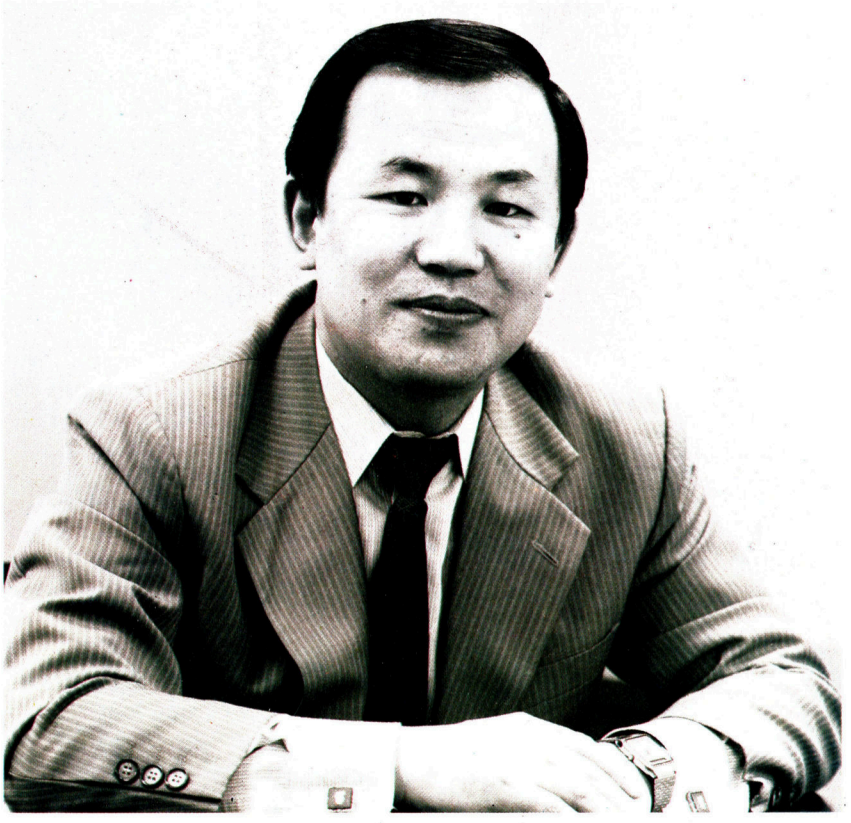
Die Anfänge des Unternehmens können nicht genau zurückverfolgt werden. Takayoshii Shiina, der Präsident der Firma, verließ die Universität und begann bei Rikei Industries, einem Unternehmen, das 1967 an der Tokioer Börse lediglich einen untergeordneten Rang hatte. Er hatte die Aufgabe, die Marketingpolitik des Unternehmens neu zu organisieren.

Daß Shiina überhaupt von Rikei eingestellt wurde, ist ein kleines Wunder in diesem Land, in dem eigentlich nur Stellen auf Lebenszeit vergeben werden. – Denn beim Einstellungsgespräch verkündete er ganz offen, daß er nur für ein paar Jahre bleiben wolle und beabsichtige, ein eigenes Unternehmen zu gründen.

1970 waren dann Shiina und ein Freund so weit, daß sie auf eigene Rechnung starten konnten, und SORD wurde mit einem Stammkapital von 650 000 Yen (nach heutigem Wechselkurs etwa 8000 Mark) gegründet. Shiina arbeitete noch bis Ende Dezember des Jahres für Rikei. SORDs erstes Produkt war ein Logik-Tester; ferner produzierte man Programme für Einzelaufträge.

Anfang 1971 war SORD bereits gut ausgelastet. Der Arbeitsschwerpunkt lag noch immer in der Software-Produktion. 1973 begann SORD mit der Hardware-Herstellung und hatte bis Ende 1974 eine Diskettenstation – ein Fremdprodukt – mit Hilfe eines SORD-eigenen Interfaces lauffähig gemacht. Darauf folgte bald der SMP-80/20, einer der ersten japanischen Computer auf Basis des Intel-8080-Microprozessors.

Der Rechner SMP-80/20 war ein äußerst erfolgreiches Produkt. Man konnte sich vor Aufträgen kaum retten. Doch Shiina hatte Expansionsabsichten und gewann 1977 Toppan, eine der größten japanischen Druckereien, als Partner. Deren Einlage von 20 % bot die finanziellen Voraussetzungen für die Produktion von Software als Ergänzung zur ständig wachsenden Hardware-Palette.



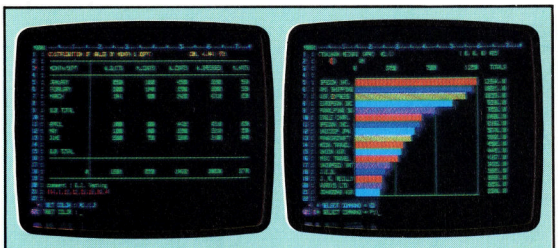
Takayoshii Shiina

Ausgezeichnete Software

PIPS, ein Software-Paket, war seiner Zeit (1981) weit voraus. Als eines der ersten Beispiele integrierter Software half es SORD, dessen Position im Markt zu festigen. In PIPS sind die Funktionen eines Spreadsheets, einer Textverarbeitung und einer Datenbank in einer Form kombiniert, daß selbst Menschen, die vorher keine Erfahrung im Umgang mit Computern hatten, schon nach wenigen Stunden damit arbeiten können.

Der phänomenale Erfolg von PIPS erklärt sich aus einer damaligen Besonderheit des japanischen Marktes: Es war in Japan üblich, Computer ohne unterstützende Software zu verkaufen.

SORD begann mit der Entwicklung seiner M200-Computerserie auf der Basis des Z80, worauf die M23- und M343-Serien folgten. Für den Heimbereich entwickelte man den M5, und heute gibt es die Serie M68, die sich sowohl des Z80 als auch des Motorola-68000 als



PIPS
Pan Information Processing System heißt SORDs selbstentwickeltes Software-Paket für kommerzielle Anwender. Es ist ein Vielzweckprogramm, das von der Kontoführung bis zur Grafikdarstellung fast alle Aufgaben im Geschäftsbereich lösen kann. Mit PIPS wurde ein interessanter Marketing-Versuch gestartet: Das Programm wird nur mit bestimmten SORD-Rechnern geliefert – und zwar kostenlos.

Prozessor bedient. Ferner bietet SORD Rechner für Kleinbetriebe (den M243) und den 32-Bit-Computer M285, auf dem auch VAX-11-Software für CAD laufen kann.

Kein anderes Unternehmen der Welt bietet ein so breites Rechnerspektrum an. Für jedes System steht eine Fülle von Software zur Verfügung. Allerdings scheint SORD ein Problem zu haben, das als „Turn-Key" bezeichnet wird. Darunter versteht man, daß die Systeme als vollarbeitende Rechner betrachtet werden und Hard- wie Software einschließen. Das hat zur Folge, daß SORD-Anwender kaum Möglichkeiten haben, auf Programme von Fremdanbietern zurückzugreifen. Es gibt nur wenige sogenannte „Third Party"-Firmen, die SORD-kompatible Software produzieren.

Um dem entgegenzuwirken, stattete das Unternehmen seine M23-Serie (mit dem Z80 als Grundlage) mit dem Betriebssystem SB-80 als Option aus. SB-80 entspricht dem von Digital Research entwickelten Betriebssystem CP/M (2.2). Somit ist es erstmals möglich, CP/M-Software auf SORD-Computern zu betreiben.

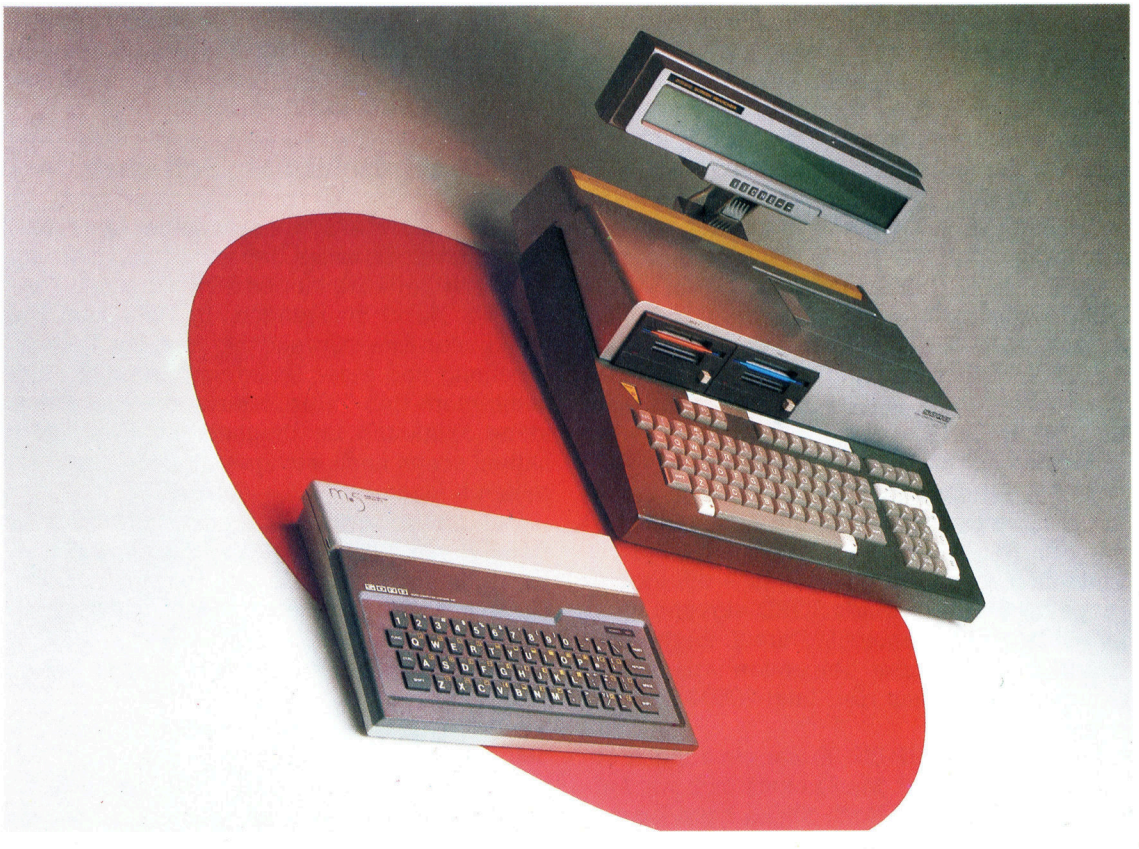
Anwendern, die sich mit SORD-eigener Software begnügen, stehen mehrere BASIC-Versionen zur Verfügung, ferner das eigene Betriebssystem, PIPS sowie eine Wang-ähnliche Textverarbeitung.

Neben dem Angebot von CP/M und p-System folgt SORD einem Weg, den neben anderen Minicomputer-Herstellern auch DEC eingeschlagen hat, nämlich Systeme anzubieten, die im wesentlichen von selbst entwickelten Programmen abhängig sind.

Weiterhin produziert SORD innovative Software. Kritiker merken jedoch dazu an, daß die Dokumentation den Ansprüchen qualitativ bei weitem nicht standhält. In den vergangenen Jahren hat SORD erhebliche Anstrengungen unternommen, gute Unterlagen für PIPS wie für den Wortprozessor zu erstellen.

Die Hardware-Dokumentation läßt aber unverändert zu wünschen übrig bzw. ist nicht vorhanden, und noch immer weigern sich Third-Party-Häuser, Programme für diese Rechner zu schreiben.

Die nächsten Jahre werden für SORD sicherlich kritisch sein, da der Wettbewerb mit den größeren japanischen Firmen und IBM noch härter wird. Unverändert glaubt Shiina fest an die Möglichkeiten eines kleinen Unternehmens und an Individualismus.



SORDs bekannteste Rechner sind die Heimcomputer, so der M5 und der tragbare M32P-Business-Rechner. Der M32P setzt mit Verwendung der Sony-Floppy-Disk und einem 80-Zeichen-LCD-Schirm neue Maßstäbe.

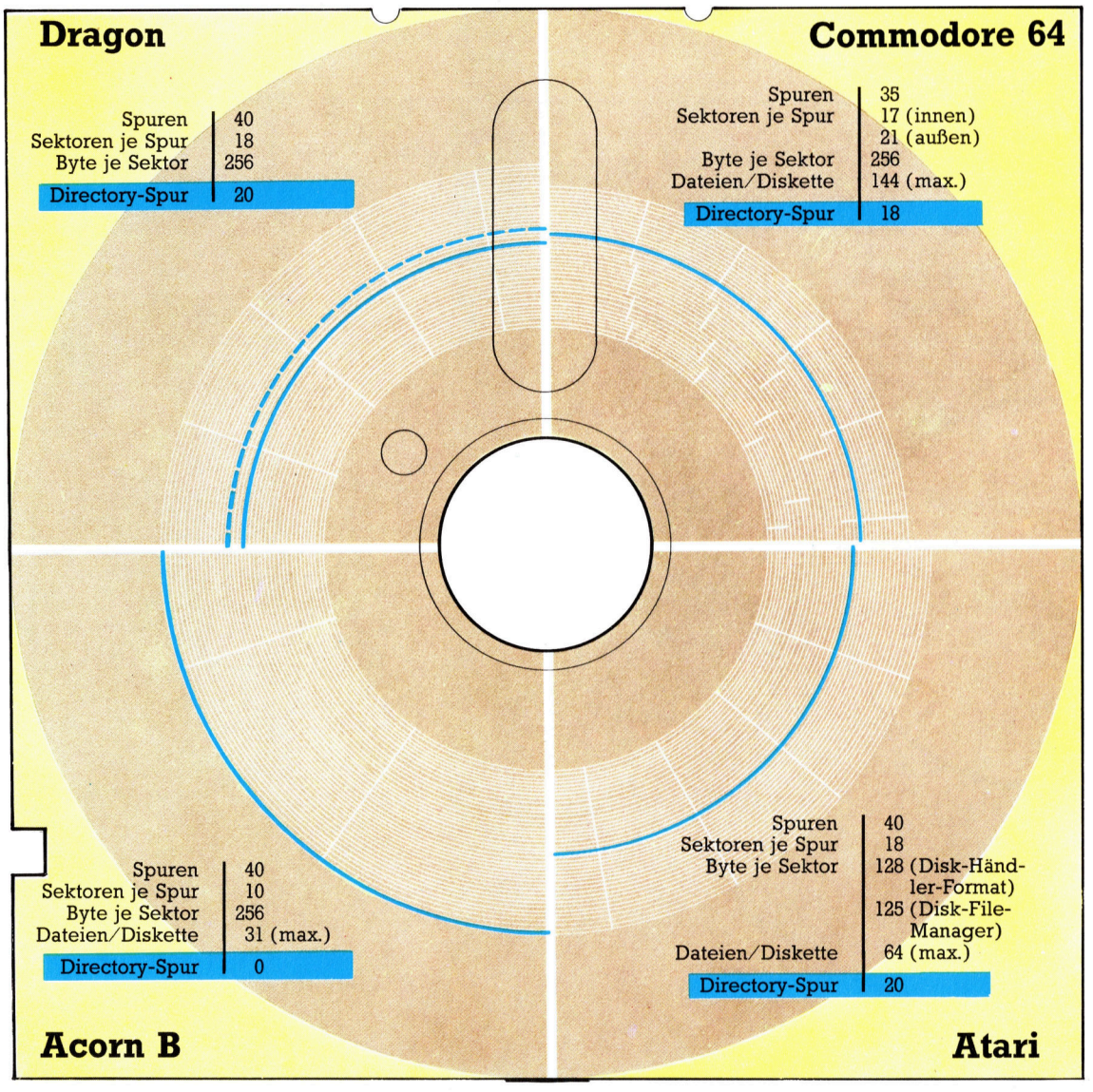
Sektoren-Grenze

Der heutige Stand der Anwender-Software wäre ohne Diskettenlaufwerke nicht denkbar. Sie erlauben den Aufbau von Datenbanken, wie sie früher nur bei Großrechenanlagen möglich waren. Disketten haben aber auch ihre Schattenseiten.

Bevor Sie auf einer Diskette Daten ablegen können, muß sie speziell für Ihren Rechner formatiert werden. Das geschieht immer nach dem gleichen Prinzip, aber die Details sind von Gerät zu Gerät verschieden. Deshalb sind beschriebene Disketten zumeist nicht auf einem anderen Computer lauffähig, auch wenn die Information selbst weitgehende Kompatibilität zulassen sollte.

Das Formatieren läßt sich in etwa mit dem Ziehen von Linien auf einem Blatt Papier vergleichen. Die Magnetschicht der Diskette wird dabei vom Rechner mit Informationen versehen, durch die eine Anzahl konzentrischer Aufzeichnungsspuren (Tracks) festgelegt wird, die tortenstückartig in Sektoren unterteilt sind. Einige Laufwerke arbeiten mit 40, andere mit 80 Spuren; einige formatieren die Disketten mit zwei Schreib/Lese-Köpfen beidseitig, andere nutzen nur die Oberseite. Die Definition der Sektoren durch das Formatieren heißt „Softsektorierung", während bei der mittlerweile überholten „Hardsektorierung" am Innenrand der Diskette eingestanzte Löcher die Sektoren kennzeichnen.

Bei der Formatierung wird stets nur etwa ein



Obwohl beim Acorn B, beim Dragon, beim Commodore 64 und beim Atari der gleiche Diskettentyp verwendet wird, ist die Formatierung bei jedem Gerät anders. Die Anzahl der speicherbaren Dateien hängt vom Betriebssystem ab. Das DOS legt das Dateiverzeichnis (Directory) beim Acorn B auf der äußersten Spur (auf Spur 0) an, bei den anderen drei Rechnern dagegen auf einer mittleren Spur. Diese Anordnung bietet den Vorteil, daß der Schreib/Lese-Kopf einen kürzeren Weg zurücklegen muß, um zwischen der Directory-Spur und den jeweiligen Datenspuren hin- und herzuwechseln. So ergeben sich kürzere Zugriffszeiten.

Synchronisationssignal 1
Justiert die Lesegeschwindigkeit des Schreibkopfes anhand der Umdrehungsgeschwindigkeit der Diskette

Spurnummer
Ermöglicht die Erkennung der jeweiligen Spur

Sektornummer
Bezeichnet den jeweiligen Sektor

Prüfsumme 1
Wird zur Fehlererkennung bei den Identifikationsdaten verwendet

Identifikationsdaten

Lücke 1
Trennt Identifikationsdaten vom eigentlichen Datenfeld

Synchronisationssignal 2

Kennungsdatei
Gibt an, wo der nächste Sektor der jeweiligen Datei zu finden ist

Datenblock
Kann 128, 256 oder 512 Datenbytes enthalten, je nach Disketten-Betriebssystem

Prüfsumme 2
Ermöglicht eine Fehlererkennung innerhalb der abgelegten Daten

Datenfeld

Lücke 2
Dient zur Trennung vom nachfolgenden Sektor

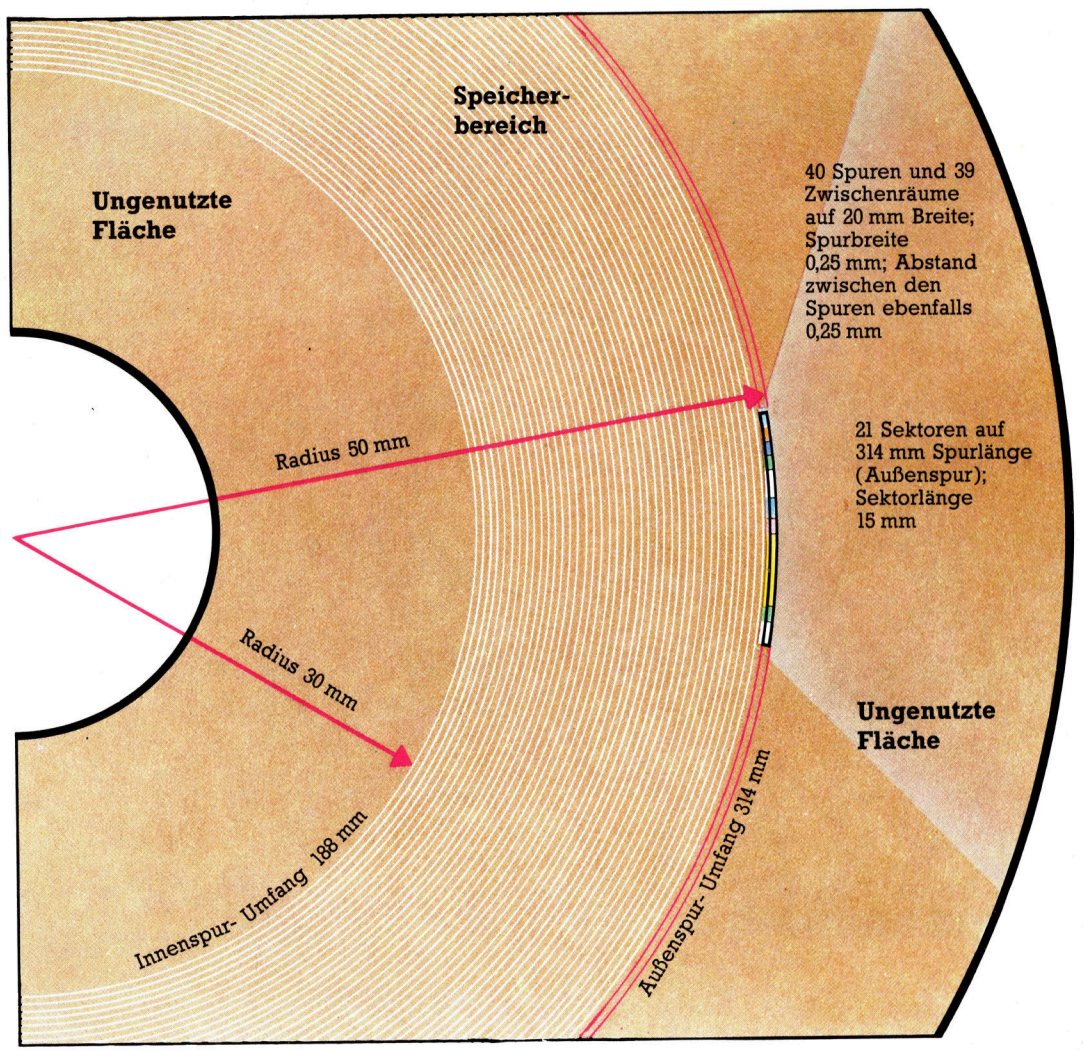
Die Information wird in Form von konzentrischen Spuren aufgezeichnet, die wiederum in Sektoren unterteilt sind. Jeder Sektor einer Spur enthält einen Block mit Daten. Diese Datenblöcke werden vom Betriebssystem automatisch mit Fehlererkennungs-Bytes und Identifikationsfeldern versehen, die für Datei- und Blockverwaltung erforderlich sind.

Drittel der Diskettenoberfläche zum Speichern präpariert. Die Speicherkapazität ist eine Frage der Aufzeichnungsdichte (Density). – Auf der gleichen Fläche ist bei einer „Double Density"-Diskette doppelt so viel unterzubringen wie bei „Single Density". Im Endergebnis kann sich die Kapazität von ca. 90 KByte für einseitige Single-Density-Disketten steigern.

Nach der Formatierung können Sie die Diskette beschreiben. Die Information wird blockweise in Dateien eingetragen, die vom Betriebssystem (Disk Operating System = DOS) mit Hilfe eines Dateiverzeichnisses (Directory) verwaltet werden. Das DOS verwendet zum Speichern beliebige freie Sektoren. Es setzt eine Kennung in jeden benutzten Sektor, die angibt, wo der nächste Teil der Datei steht, und führt eine Belegungsliste.

Einführung von Standards

Bis vor kurzem gab es bei der Formatierung kaum eine Standardisierung oder Kompatibilität. Ob die auf einem System beschriebene Diskette auf einem anderen läuft, hängt einerseits von der Formatierung ab, andererseits von der Anordnung der Dateien. Glücklicherweise verwenden inzwischen viele Microcomputer das gleiche Formatierungssystem oder zumindest Hilfsprogramme, die verschiedene Dateiformate bearbeiten können. Viele Rechner arbeiten mit Standard-Betriebssystemen wie CP/M (Control Program for Microcomputers) oder MS-DOS (Microsoft-DOS), so daß auch eine einheitliche Anordnung der Dateien gewährleistet ist.



Grade der Präzision

Im zweiten Teil unserer Serie über Mathematik bei der BASIC-Programmierung fahren wir mit der Betrachtung von trigonometrischen Funktionen fort. Wie können die Sinus- und Kosinus-Funktionen in BASIC-Programmen verwendet werden?

Da BASIC sowohl mit der Sinus- als auch mit der Kosinus-Funktion ausgestattet ist, sollte die Berechnung der Position eines Punktes auf einer Linie nach einer Drehung um einen bestimmten Winkel eigentlich kein Problem darstellen. Der cos von Θ gibt die Position auf der X-Achse (X-Koordinate) an, und der sin von Θ die Position auf der Y-Achse (Y-Koordinate). Bei der Verwendung der beiden Funktionen müssen Sie daran denken, daß die meisten BASIC-Versionen mit dem Bogenmaß (Radiant) und nicht mit Grad arbeiten.

Zunächst soll versucht werden, den Unterschied zwischen Grad und Radiant zu überbrücken. Wenn ein Teil eines Kreises (genannt ein Bogen) so gezeichnet wird, daß seine Länge exakt dem Radius des Kreises entspricht, ist der Winkel im Zentrum ein Radiant (siehe Bild). Wenn der Radius des Kreises eine Einheit ist, hat dieser Teil des Kreisumfanges ebenfalls die Länge von einer Einheit. Die Formel zur Ermittlung des Kreisumfanges lautet $2\pi r$. Das bedeutet, daß eine komplette Umdrehung 2π Radianten sein müssen. Wie Sie sehen, entsprechen also 360° genau 2π Radianten. Dies gestattet uns nun, Grad und Radiant zueinander ins Verhältnis zu setzen:

360° = 2π Radianten
180° = π Radianten
90° = $\pi/2$ Radianten
1° = $\pi/180$ = 0,0174 Radianten

Ein BASIC-Programm, das den Kosinus eines in Grad gemessenen Winkels herausfinden soll, muß zuerst das Winkelmaß von Grad in Radiant umrechnen und dann die cos-Funktion anwenden. Testen Sie das folgende Programm:

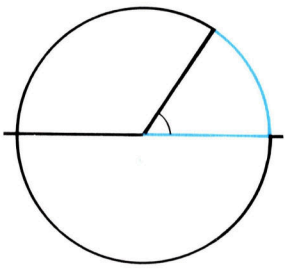
```
10 INPUT "GIB DEN WINKEL IN GRAD
   AN";A
20 LET B# = A * 0.0174
30 LET C# = COS(B#)
40 PRINT "DER KOSINUS VON ";A;" GRAD
   BETRAEGT ";C#
50 END
```

Das Nummernzeichen bedeutet, daß die Variablen in diesem Programm doppelt genau sind. Eine einfache Änderung an diesem Programm unter Verwendung der sin-Funktion liest alle Werte von theta von 0° bis 360° ein und gibt die jeweiligen Sinus-Werte in Form einer Tabelle aus. Werden diese Werte entlang der Y-Achse eines Diagramms dargestellt, erhält man die Sinus-Kurve. Diese Kurve ist nichts anderes als die Darstellung aller Positionen der Schnittstelle der Hypotenuse mit dem Einheitskreis für alle Winkel einer Drehung. Mit anderen Worten: Dies ist ein alternativer Weg, einen Kreis mathematisch zu beschreiben.

Einige BASIC-Versionen erlauben, daß die sin- und cos-Funktionen sowohl mit Grad als auch mit Radiant arbeiten, indem man eine spezielle Funktion aufruft. Bei den meisten Versionen ist dies jedoch nicht möglich. Wenn Sie es vorziehen, ständig mit Grad zu arbeiten, können Sie eine Funktion definieren, die die Umrechnung für Sie erledigt. Das folgende Programm stellt eine Möglichkeit dar:

```
10 REM FUNKTION FUER
   BERECHNUNGEN IN GRAD
20 DEF FNGSIN (D#)=SIN
   (D#*0.017453293)
30 INPUT "GIB DEN WINKEL IN GRAD
   AN";D#
40 PRINT "DER SINUS VON #";D#;" GRAD
   BETRAEGT ";FNGSIN(D#)
50 END
```

Ein Radiant



Fehler bei der Sinus-Funktion

```
 1 REM TEST FUER DIE BERECHNUNG DER SIN-FUNKTION
10 LET X=10/6
20 PRINT "WIEDERHOLUNG", "WERT VON X", "WERT VON SIN(X)"
30 FOR I=1 TO 40
40 LET X=X/10
50 PRINT I,X,SIN(X)
60 NEXT I
70 END
```

WIEDERHOLUNG	WERT VON X	WERT VON SIN(X)
1	.166667	.165896
2	.0166667	.0166659
3	1.66667E-03	1.66667E-03
4	1.66667E-04	1.66667E-04
5	1.66667E-05	1.66667E-05
6	1.66667E-06	1.66667E-06
7	1.66667E-07	1.66667E-07
8	1.66667E-08	1.66667E-08
9	1.66667E-09	1.66667E-09
10	1.66667E-10	1.66667E-10
11	1.66667E-11	1.66667E-11
12	1.66667E-12	1.66667E-12
13	1.66667E-13	1.66667E-13
14	1.66667E-14	1.66667E-14
15	1.66667E-15	1.66667E-15
16	1.66667E-16	1.66667E-16
17	1.66667E-17	1.66667E-17
18	1.66667E-18	1.66667E-18
19	1.66667E-19	1.66667E-19
20	1.66667E-20	1.66667E-20
21	1.66667E-21	1.66667E-21
22	1.66667E-22	1.66667E-22
23	1.66667E-23	1.66667E-23
24	1.66667E-24	1.66667E-24
25	1.66667E-25	1.66667E-25
26	1.66667E-26	1.66667E-26
27	1.66667E-27	1.66667E-27
28	1.66667E-28	1.66667E-28
29	1.66667E-29	1.66667E-29
30	1.66667E-30	1.66667E-30
31	1.66667E-31	1.66667E-31
32	1.66667E-32	1.66667E-32
33	1.66667E-33	1.66667E-33
34	1.66667E-34	1.66667E-34
35	1.66667E-35	1.66667E-35
36	1.66667E-36	1.66667E-36
37	1.66667E-37	1.66667E-37
38	1.66667E-38	1.66667E-38
39	0	0
40	0	0

In Zeile 20 wird eine Funktion GSIN aufgerufen (GSIN steht für GRAD/SINUS), die als Parameter die doppelt genaue Variable D# verwendet. Die rechte Hälfte der Definition zeigt lediglich, wie der Wert, der durch die Funktion zurückgegeben werden soll (der Sinus eines Winkels in Grad), abgeleitet wird.

Eines der Probleme bei Verwendung der Sinus-Funktion in BASIC ist, daß nicht alle BASIC-Versionen sie korrekt handhaben, wenn der Wert von theta sich 0 nähert. BASIC arbeitet nur in einem bestimmten Zahlenbereich sehr genau. Wenn theta sehr klein ist (zum Beispiel 1,0E-36 = 1 × 10 hoch Minus 36), ist BASIC überfordert und gibt einfach einen Wert 0 als Ergebnis für den Sinus einer solchen Zahl aus. Bevor Sie die Sinus-Funktion ausprobieren, testen Sie zuerst einmal Ihr BASIC mit dem Programm der vorherigen Seite.

Wir haben einen Beispiellauf dieses Programmes in Microsoft-MBASIC beigefügt. Dieser spezielle BASIC-Interpreter bearbeitet den sin von kleinen Zahlen recht genau und verursacht keine Probleme, bis zu einem Wert von Θ kleiner als 1,0E-38 (ein Dezimalpunkt gefolgt von 37 Nullen).

Das gezeigte Programm ist abhängig von einem angemessenen dynamischen BASIC-Bereich in bezug auf arithmetische Fließkomma-Operationen. Es ist also ratsam, vor Verwendung mathematischer Operationen zuerst einmal den Bereich der Zahlen, der korrekt verarbeitet werden kann, zu überprüfen.

Beachten Sie auch, daß ein Variablenname allein, wie beispielsweise X oder TREND, automatisch nur einfach genau ist (das heißt nicht mehr als sieben Stellen speichern kann). Variablen können auch von vornherein als einfach genau spezifiziert oder dazu „gemacht" werden, indem man ein Ausrufungszeichen anhängt (in diesem Fall X! oder TREND!). Doppelt genaue Variablen können 17 Stellen speichern. Sie werden spezifiziert, indem man ein Nummernzeichen anhängt, wie bei X# oder TREND#. Integer-Variablen werden bei vielen BASIC-Versionen spezifiziert, indem dem Variablennamen ein Prozentzeichen angehängt wird, hier beispielsweise X% oder TREND%.

Wir beenden diesen Artikel mit einem kurzen Programm, mit dem Sie testen können, wie viele Stellen in einer Variable Ihrer BASIC-Version gespeichert werden können. Zusätzlich zeigen wir Ihnen einen Ausdruck eines Programmlaufs unter Verwendung von Microsoft-BASIC. Es gibt zwei Versionen des Programms, eine zum Testen von kleinen Zahlen und eine für große Zahlen. Der Ausdruck für die kleinen Zahlen zeigt, daß BASIC die Zahlen zu 0 rundet, wenn die Zahlen sehr klein werden (kleiner als 3,3×10E-38). Für große Zahlen (größer als 3,3×10E37) tritt ein Überlauf ein, und die Ergebnisse sind nicht mehr exakt. Wenn Sie mit sehr großen oder sehr kleinen Zahlen arbeiten müssen, kann es sein, daß Sie spezielle Routinen entwickeln müssen, um diese Einschränkungen zu umgehen.

Kleine Zahlen in BASIC

```
1 REM TESTET DIE HANDHABUNG VON KLEINEN ZAHLEN IN BASIC
10 LET X# =.00003333333333#
20 PRINT "WIEDERHOLUNG","  DOPPELT GENAU","  "," EINFACH GENAU"
30 PRINT
40 FOR I=1 TO 40
50 LET X#=X#/10
60 LET X!=X#
70 PRINT I,X#,X!
80 NEXT I
90 END
```

WIEDERHOLUNG	DOPPELT GENAU	EINFACH GENAU
1	.000003333333333	3.33333E-06
2	.0000003333333333	3.33333E-07
3	3.333333333D-08	3.33333E-08
4	3.333333333D-09	3.33333E-09
5	3.333333333D-10	3.33333E-10
6	3.333333333D-11	3.33333E-11
7	3.333333333D-12	3.33333E-12
8	3.333333333D-13	3.33333E-13
9	3.333333333D-14	3.33333E-14
10	3.333333333D-15	3.33333E-15
11	3.333333333D-16	3.33333E-16
12	3.333333333D-17	3.33333E-17
13	3.333333333D-18	3.33333E-18
14	3.333333333D-19	3.33333E-19
15	3.333333333D-20	3.33333E-20
16	3.333333333D-21	3.33333E-21
17	3.333333333D-22	3.33333E-22
18	3.333333333D-23	3.33333E-23
19	3.333333333D-24	3.33333E-24
20	3.333333333D-25	3.33333E-25
21	3.333333333D-26	3.33333E-26
22	3.333333333D-27	3.33333E-27
23	3.333333333D-28	3.33333E-28
24	3.333333333D-29	3.33333E-29
25	3.333333333D-30	3.33333E-30
26	3.333333333D-31	3.33333E-31
27	3.333333333D-32	3.33333E-32
28	3.333333333D-33	3.33333E-33
29	3.333333333D-34	3.33333E-34
30	3.333333333D-35	3.33333E-35
31	3.333333333D-36	3.33333E-36
32	3.333333333D-37	3.33333E-37
33	3.333333333D-38	3.33334E-38
34	0	0
35	0	0
36	0	0
37	0	0
38	0	0
39	0	0
40	0	0

Große Zahlen in BASIC

```
1 REM TESTET DIE HANDHABUNG VON GROSSEN ZAHLEN IN BASIC
10 LET X# = 3.333333333333334#
20 PRINT "WIEDERHOLUNG","  DOPPELT GENAU","  "," EINFACH GENAU"
30 PRINT
40 FOR I=1 TO 40
50 LET X#=X#*10
60 LET X!=X#
70 PRINT I, X#,X!
80 NEXT I
90 END
```

WIEDERHOLUNG	DOPPELT GENAU	EINFACH GENAU
1	33.33333333333334	33.3333
2	333.3333333333334	333.333
3	3333.333333333334	3333.33
4	33333.33333333334	33333.3
5	333333.3333333334	333333
6	3333333.333333334	3.33333E+06
7	33333333.33333334	3.33333E+07
8	333333333.3333334	3.33333E+08
9	3333333333.333334	3.33333E+09
10	33333333333.33334	3.33333E+10
11	333333333333.3334	3.33333E+11
12	3333333333333.334	3.33333E+12
13	33333333333333.34	3.33333E+13
14	333333333333333.4	3.33333E+14
15	3333333333333334	3.33333E+15
16	3.333333333333334D+16	3.33333E+16
17	3.333333333333334D+17	3.33333E+17
18	3.333333333333334D+18	3.33333E+18
19	3.333333333333334D+19	3.33333E+19
20	3.333333333333334D+20	3.33333E+20
21	3.333333333333334D+21	3.33333E+21
22	3.333333333333334D+22	3.33333E+22
23	3.333333333333334D+23	3.33333E+23
24	3.333333333333334D+24	3.33333E+24
25	3.333333333333334D+25	3.33333E+25
26	3.333333333333334D+26	3.33333E+26
27	3.333333333333334D+27	3.33333E+27
28	3.333333333333334D+28	3.33333E+28
29	3.333333333333334D+29	3.33333E+29
30	3.333333333333334D+30	3.33333E+30
31	3.333333333333335D+31	3.33333E+31
32	3.333333333333335D+32	3.33333E+32
33	3.333333333333334D+33	3.33333E+33
34	3.333333333333335D+34	3.33333E+34
35	3.333333333333335D+35	3.33333E+35
36	3.333333333333335D+36	3.33333E+36
37	3.333333333333335D+37	3.33333E+37
38	1.701411834604693D+38	1.70141E+38
39	1.701411834604693D+38	1.70141E+38
40	1.701411834604693D+38	1.70141E+38

Fachwörter von A bis Z

Control Character = Steuerzeichen
Steuerzeichen sind ASCII-Codes, die einen spezifischen Vorgang bei dem Rechner oder einem Peripheriegerät auslösen. Sie gehören nicht zum alphanumerischen Zeichensatz (Buchstaben, Ziffern und Satzzeichen) und sind großenteils nicht ausdruckbar. Viele dieser Symbole sind allgemeingültig festgelegt – das ASCII-Zeichen „13" etwa bedeutet stets Zeilenvorschub – während einige andere je nach Maschinentyp unterschiedlich definiert sind. Die Oric-Rechner arbeiten beispielsweise mit Steuerzeichen zur Verdopplung der Schriftgröße, zur Inaktivierung der Bildschirmdarstellung und zum Farbwechsel. Bei Textverarbeitungssystemen erhält der Drucker über Formatsteuerzeichen Anweisungen für die Textausgabe.

Courseware = Lernsoftware
Der Begriff „Courseware" steht für Lernprogramme, die in der Ausbildung beim computerunterstützten Lernen verwendet werden. Das kann eine Zusammenstellung aus den verschiedensten Quellen und Themengebieten für einen bestimmten Rechner sein, aber auch ein Paket mit einheitlicher Dialogstruktur, wobei die Kursprogramme unterschiedliche Gegenstände behandeln.

CP/M = CP/M
CP/M ist ein Betriebssystem für Rechner auf 8080- oder Z80-Basis, das von Gary Kildall bei Digital Research entwickelt wurde und sich inzwischen als Standard behaupten konnte. CP/M (Control Program for Microprocessors) war das erste universelle Betriebssystem. Es wurde von zahlreichen Herstellern übernommen und ermöglicht die Verwendung einmal erstellter Software auf einer Vielzahl unterschiedlicher Rechner.

Bildnachweise
814, 815, 826, 827: Chris Stevens
816, 817, 819, 834: Liz Dixon
818, 823, 831, 836: Ian McKinnell
821: Rosalind Buckland
825, 830: Steve Cross
829, U3: Kevin Jones
835: Sord

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.

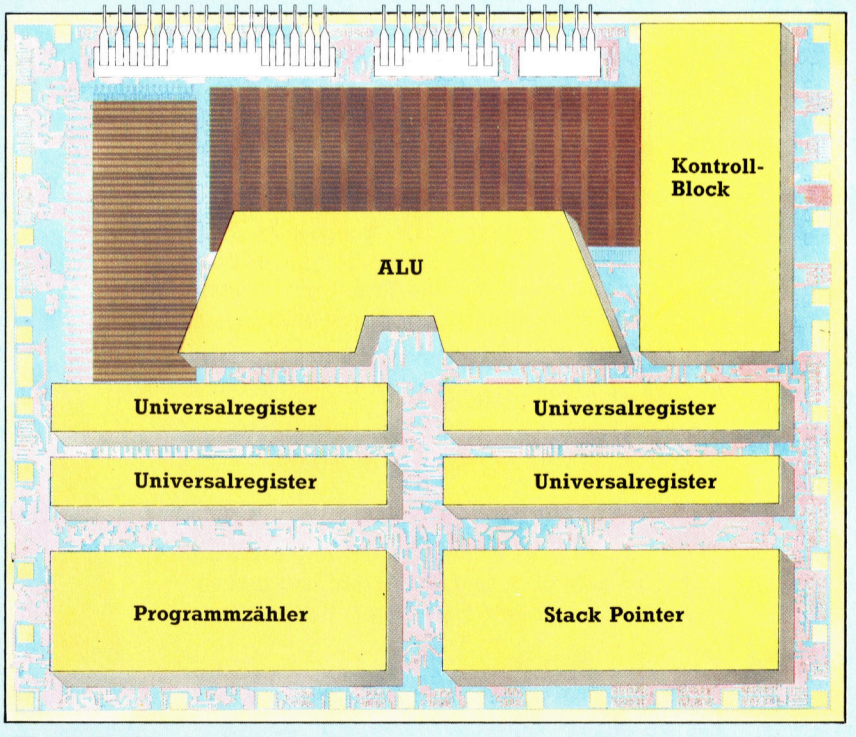
CPU = Zentraleinheit
Die CPU (Central Processing Unit) oder Zentraleinheit eines Computers ist für die interne Organisation aller Arbeitsabläufe verantwortlich.

Die Aufgabe der CPU besteht in der Entgegennahme und Ausführung von Maschinenbefehlen, zum Beispiel für das Umstrukturieren von Speicherinhalten oder für einfache arithmetische Operationen. Alle Zahlen und Befehle werden binär verschlüsselt.

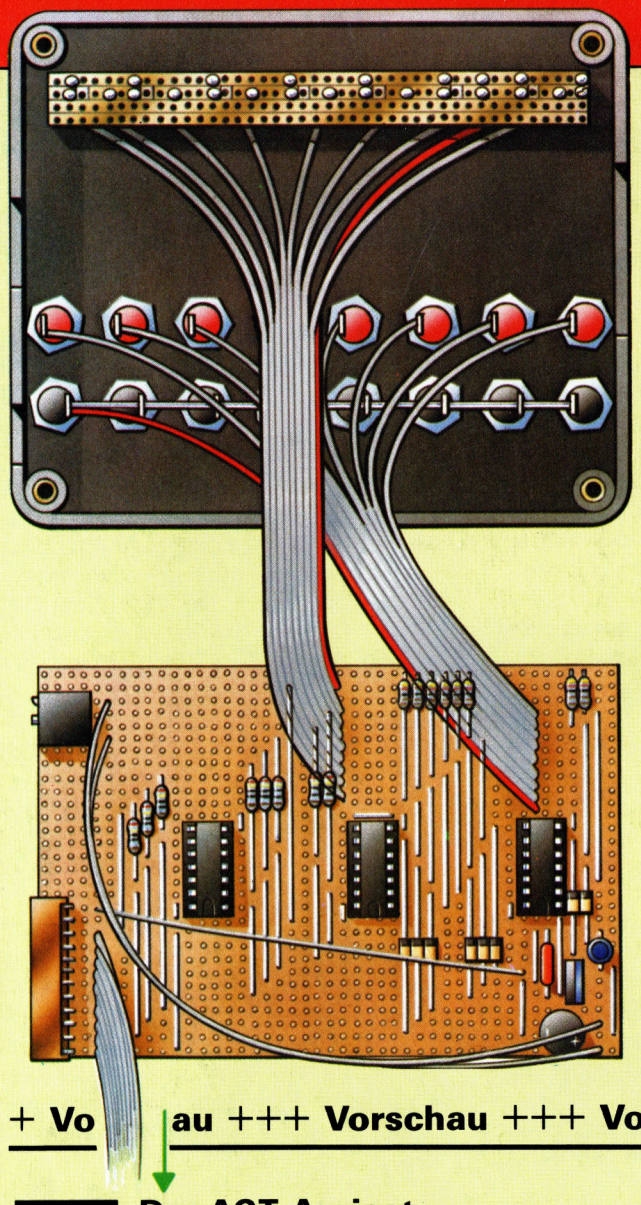
Ein CPU-Baustein enthält viele tausend elektronische Funktionselemente, wobei die Leistungsfähigkeit je nach Typ unterschiedlich ist. Viele Taschenrechner haben eine 4-Bit-CPU, Heimcomputer arbeiten meist mit 8 Bit, eine Reihe von Bürorechnern mit 16 Bit, und mittlere sowie Großrechenanlagen verfügen über eine 32- oder 64-Bit-CPU.

Unterschiede bestehen auch hinsichtlich der in die CPU einbezogenen Funktionen. Der Taktgeber für die unerläßlichen Synchron-Signale ist häufig in die CPU integriert, er kann aber auch als Zusatzbaustein erforderlich sein. Manche CPUs haben intern schon RAMs oder ROMs und Ein/Ausgabe-Baugruppen („Einchip-Microcomputer"), die meisten sind jedoch ohne externe Speicher und andere Bausteine nicht betriebsfähig.

Der Daten- und Steuersignalaustausch mit der CPU erfolgt über drei Leitungssysteme: Daten-, Adreß- und Steuerbus. Diese Leitungen werden an die „Beinchen" (Pins) gelegt, die aus dem Gehäuse der CPU herausragen und mit dem darin verborgenen Siliziumchip über dünne Drähte verbunden sind.



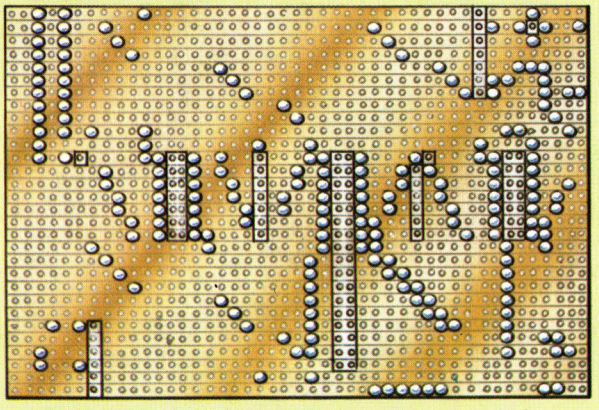
Hier sehen Sie eine typische Microcomputer-CPU mit den Bus-Anschlußstiften. Die überlagerten Blöcke stellen die ALU (Arithmetisch-logische Einheit) mit ihren verschiedenen Komponenten dar.



computer kurs

Heft 31

Der selbstgebaute Buffer für den User-Port wird in dieser Folge getestet. Wir erklären, wie die Anschlüsse geprüft werden.



+ Vo au +++ Vorschau +++ Vorschau +++ Vorschau +++ Vorschau +

Der ACT Apricot

Dieses Gerät ist in erster Linie für den kommerziellen Anwender konstruiert worden. Es ist in zwei Versionen erhältlich: mit zwei Floppies oder mit einer Hard-Disk und einem 3 $^1/_2$-Zoll-Laufwerk.

Roboter-Arme

Wir stellen Roboterarme vor, die gleiche Funktionen ausführen wie auch die Geräte in der Industrie. Zwar sind die Arme nur zu Demonstrationszwecken gedacht, arbeiten aber nach denselben Prinzipien wie die professionellen Greifer.

Logischer Schluß

Der LOGO-Kurs endet in diesem Heft: Wir beschäftigen uns mit Mosaik-Mustern.

Tabellen-Kalkulation

Kalkulations-Programme können nicht nur Informationen sammeln und darstellen, sie lassen sich auch als „Ideen-Generator“ verwenden.

Rasche Ratte

Als Konkurrenz zur bekannten „Maus“ hat die Firma „Cheetah Marketing“ ein Joy-Pad entwickelt, das als „Ratte“ ebensoviele Funktionen erfüllt.