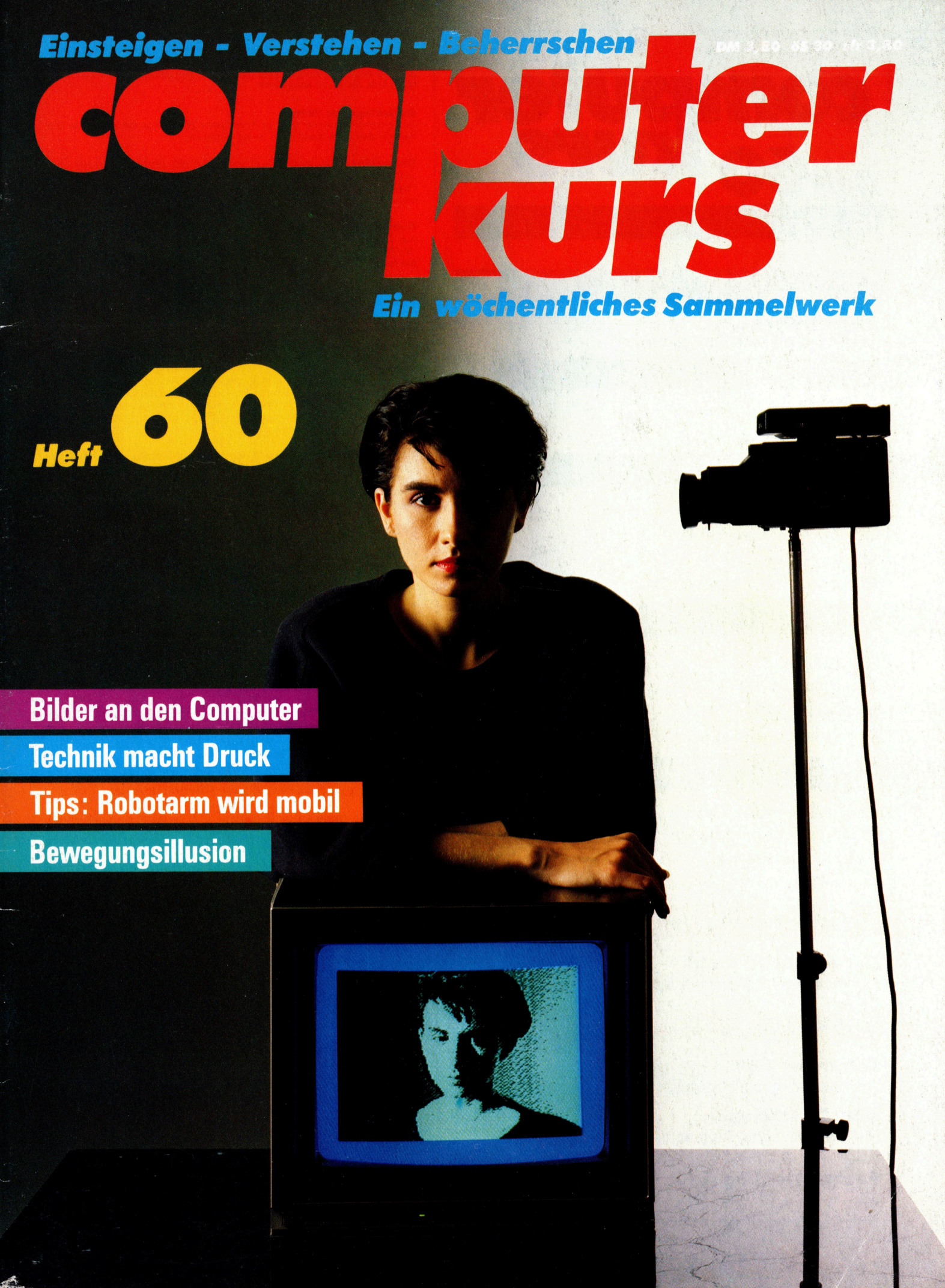
Einsteigen - Verstehen - Beherrschen

DM 3,80 öS 30 sfr 3,80

computer kurs

Ein wöchentliches Sammelwerk

Heft 60



Bilder an den Computer

Technik macht Druck

Tips: Robotarm wird mobil

Bewegungsillusion

computer kurs

Heft 60

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.

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.

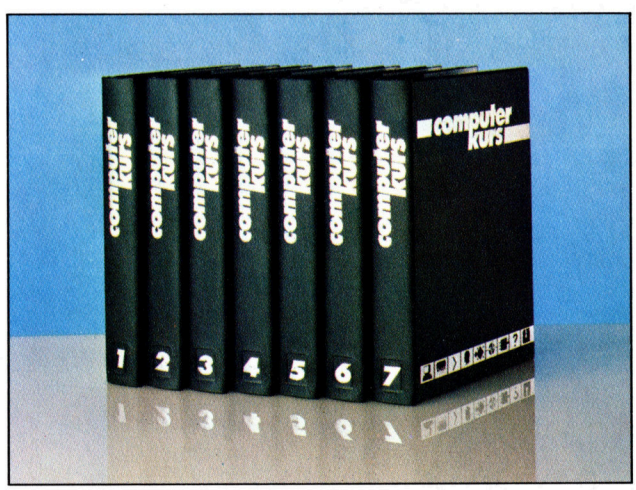
Schweiz: Der Sammelordner kostet sfr 15. Bitte wenden Sie sich an Ihren Kiosk; dort werden Sie jederzeit die gewünschten Exemplare erhalten.

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), Peter Aldick, Holger Neuhaus, Uta Brandl (Layout), Sammelwerk Redaktions-Service GmbH, Paulstraße 3, 2000 Hamburg 1

Vertrieb: Marshall Cavendish International Ltd., Heidenkampsweg 74, 2000 Hamburg 1



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

Videosignale

Für die Digitalisierung von Videosignalen gibt es inzwischen kompakte und erschwingliche Lösungen, die speziell auf Heimcomputer zugeschnitten sind. Ein Beispiel ist der „Video Digitiser".

Ein Video-Digitalisierer setzt die Analogspannung einer Videosignalquelle (vorzugsweise einer Kamera, einer Laser-Bildplatte oder eines Videorecorders) in digitale Form um, wobei jedes Bild durch eine Reihe von Zahlen repräsentiert wird. Diese Information läßt sich im Rechner speichern, in hochauflösender Grafik darstellen, softwaremäßig bearbeiten und ausdrucken.

Für derartige Video-Digitalisierer gibt es Hunderte von Einsatzmöglichkeiten, von ernsthaften Anwendungen wie Alarmsystemen bis zu „computerisierten" Bildern für Ansteck-Buttons. Die kurioseste uns bekannte Anwendung praktiziert ein Friseur, der seinen Kunden auf dem Bildschirm mit Lichtgriffel und entsprechender Software alle möglichen Frisuren verpaßt, bevor er zur Schere greift.

Dem Gerät der Firma Print-Technik traut man soviel zunächst gar nicht zu – es handelt sich um ein unscheinbares, schwarzes Kästchen, das beim C64 in den User-Port gesteckt wird und lediglich eine normale Videobuchse für den Anschluß der Signalquelle aufweist. Weitere Leitungen sind nicht nötig, weil der Rechner die Stromversorgung übernimmt. Mit drei kleinen Drehreglern werden Helligkeit, Kontrast und Bandbreite eingestellt.

Die gesamte Steuerung erfolgt über entsprechende Software. Das auf Diskette mitgelieferte Programmpaket ist zwar funktionell aufgebaut, aber die tatsächlichen Möglichkeiten des Geräts werden kaum ausgeschöpft. Den Anwender begrüßt ein optisch reizvolles Menü. Zur Anwahl der gewünschten Variante drückt man die Return-Taste. So läßt sich ein Signal digitalisieren oder ein gespeichertes Bild wiedergeben, auf Diskette SAVEn oder ausdrucken. Die Software ist für eine Vielzahl von Druckern, etwa die Commodore-Typen 801 und 1525, eingerichtet. Mit einem speziellen Programm namens „16 Colors" kann man über die fünf gängigsten Farb-Matrixdrucker auch Bilder in bis zu 16 Farben ausgeben. Bei der Bildschirmwiedergabe sind höchstens vier Farbtöne darstellbar, obgleich der Digitalisierer selbst bis zu 256 Abstufungen „wahrnehmen" kann.

Günstig, daß sich die Software auch in Verbindung mit anderen Programmen benutzen läßt. Die Variante „Lightpen" ruft das Lichtgriffel-Paket von Print-Technik auf, das allerdings nicht zum Lieferumfang gehört. Bedeutender ist, daß die Bilder auch in einem Diskettenformat abspeicherbar sind, das mit dem Koala-Pad und Paintmagic kompatibel ist. Wer damit künstlerische Entwürfe und Grafiken ausführen will, findet im Video-Digitiser eine wertvolle Ergänzung. Alle zugehörigen SAVE- und LOAD-Programme arbeiten mit schnellen Speicherroutinen.

Die Digitalisierzeit von vier Sekunden wirkt sich erschwerend für den Einsatz des Geräts aus. Als preiswertes Aufnahmesystem bietet sich eine einfache Videokamera an, wie sie auch für Überwachungszwecke eingesetzt wird. Dabei ist allerdings ein normaler Videomonitor fast unentbehrlich, um das Aufnahmeobjekt richtig zu plazieren und die Schärfe einzustellen, bevor der Digitalisierer in Gang gesetzt wird. Ohne zusätzlichen Monitor ist das System im Prinzip zwar auch betriebsfähig, aber eine optimale Bildqualität ist so nicht richtig zu erzielen.

Wegen der langen „Belichtungszeit" ist es auch schwierig, bewegte Objekte zu erfassen:

Der Video-Digitiser von Print-Technik digitalisiert Bilder einer Fernsehkamera oder eines Videorecorders, so daß sie auf dem Rechnermonitor in vier Farbtönen wiedergegeben oder auch ausgedruckt werden können. Die Bilder lassen sich im gleichen Format, das von Grafikpaketen wie dem Koala-Pad verwendet wird, auf Floppy speichern.

Print-Technik Video-Digitizer

ABMESSUNGEN
80 x 65 x 20 mm

BEDIENUNGS-ELEMENTE
Drehregler für Helligkeit, Kontrast und Bildbreite

AUSLÖSUNG
256 x 256 Pixel bei (theoretisch) 256 Helligkeitsstufen; Wiedergabe normalerweise im Format 160 x 200 mit vier Stufen

GESCHWINDIGKEIT
Vier Sekunden pro Bild

SOFTWARE
Die Standardsoftware unterstützt Digitalisieren, Abspeichern mit 256 x 256 Punkten, Ausdrucken und Farbwahl.

DOKUMENTATION
Leider enthält die knappe Anleitung kaum Angaben über technische Details; die Bedienung ist einfach.

STÄRKEN
Das Gerät ist handlich, nicht zu teuer und arbeitet ausgezeichnet. Die Software ist einfachen Anwendungen angemessen.

SCHWÄCHEN
Ohne zusätzlichen Monitor ist nicht auszukommen, und die Wiedergabemöglichkeiten des C64 schöpfen die Möglichkeiten des Digitizers nicht aus.

Für brauchbare Porträts muß das Modell vier Sekunden absolut stillhalten, es sei denn, man legt Wert auf Verwackel-Effekte. Abhilfe ist auf zweierlei Weise möglich: Das Ideale ist ein „Frame Grabber" (Bildgreifer), der das Bild elektronisch einfriert. Die Alternative besteht darin, einfach ein Foto zu digitalisieren.

Wenn das bildgebende Gerät richtig eingestellt ist, kann man die Digitalisierung allenfalls noch durch die drei Potentiometer am Umsetzer beeinflussen. Sie sind werksseitig zwar optimal voreingestellt, aber ungünstige oder ungewöhnliche Bedingungen können doch eine Korrektur erforderlich machen. Die richtige Helligkeits- und Kontrast-Einstellung muß in Versuchen ermittelt werden, weil sich die Wirkung nicht direkt auf dem Monitor erkennen läßt. Mit dem Bildbreiteregler läßt sich die Aufnahme auf dem Schirm dehnen oder stauchen, um natürliche oder auch willkürliche Proportionen zu erzielen – so läßt sich beispielsweise ein schlankes Gesicht beliebig in die Breite ziehen.

Elektronischer Nachtwächter

Das digitalisierte Bild besteht aus einer 256* 256-Matrix und ist etwa im 160x200-Farbgrafikformat des Commodore 64 nur ausschnittweise darstellbar. Die Software erlaubt es aber, das Wiedergabefenster mit den Cursortasten über das ganze Bild hinwegzuschieben. Die Darstellung erfolgt in vier Helligkeitsstufen, obwohl der Analog/Digital-Converter in 256 Stufen arbeitet. Über die Funktionstasten können für die Wiedergabe den Graustufen vier Farben nach Wahl zugeordnet werden.

Im mitgelieferten Paket sind auch einige Programme zur Bildabtastung enthalten, von denen „Alarm" das trickreichste ist: Es macht aus Ihrer Videokamera in Verbindung mit dem Digitalisierer einen elektronischen Nachtwächter. Der Rechner digitalisiert dabei ein Bild nach dem andern, das heißt, er fertigt von der Szene etwa alle fünf Sekunden eine neue Aufnahme an.

Koala-Pad-Format

Jedes neue Bild wird mit dem vorherigen Pixel für Pixel verglichen, und wenn sich dabei mehr als eine vorgegebene Anzahl von Abweichungen ergibt, schlägt der C64 Alarm. Ist die Alarmschwelle sehr niedrig (etwa auf 200 Differenzpunkte) eingestellt, erfolgt die Auslösung schon durch einen leichten Schatten, der in den Kamerawinkel fällt.

Die Software enthält auch ein einfaches Programm zur Wiedergabe von Bildern im Koala-Pad-Format in rascher Folge sowie eine Routine, die eine Ausgabe der Bilder im Rahmen eigener BASIC-Programme über SYS-Aufruf gestattet.

Der Print-Technik-Digitiser ermöglicht, mit dem Rechner hochwertige Grafiken zu erzeugen; die kompakte Hardware bringt überraschend gute Ergebnisse, und die mitgelieferte Software bietet zahlreiche Möglichkeiten. Die praktische Anwendung erfordert aber in jedem Fall zusätzliche Ausrüstung: eine Fernsehkamera, einen Videomonitor und ein vernünftiges Grafikprogramm wie Paintmagic oder das Koala-Pad.

Das Gerät ist daher vor allem für Anwender von Interesse, die es als seriöses Hilfsmittel für den Entwurf aufwendiger Grafiken oder für interaktive Anwendungen der beschriebenen Art einsetzen wollen.

Die digitalisierten Bilder können mit verschiedenen Druckern auf Papier gebracht werden, einfarbig (in vier Helligkeitsstufen) etwa mit den Commodore-Modellen MPS 801 und 1525 oder mit einem Farbdrucker in bis zu 16 Farbtönen.

FORTH-Arrays

FORTH verfügt zwar standardmäßig nicht über Arrays, kann diese Strukturen aber definieren. Wir demonstrieren diese Definitionsart.

Bei der Programmierung des Sieb des Eratosthenes haben wir mit CREATE bereits einige Konzepte gestreift, die bei Array-Definitionen eingesetzt werden. CREATE hatte dort die Funktion

```
CREATE BITS 8192 ALLOT
```

und enthielt einen (im Vokabular eingetragenen) Header namens BITS und ein Codefeld. Das Codefeld veranlaßte, daß BITS während der Laufzeit seine Parameterfeldadresse auf dem Stapel hinterließ. Da CREATE selbst jedoch kein Parameterfeld anlegt, zieht ALLOT eine Zahl vom Stapel und reserviert im Vokabular Platz für diese Anzahl Bytes. Der Platz liegt unmittelbar hinter dem Codefeld, das von CREATE eingeschlossen wird und dient BITS als Parameterfeld.

Hier ein einfaches Beispiel für die Definition des Wortes 3**, das eine dritte Potenz berechnet. Zum Speichern aller nur möglichen Ergebnisse verwenden wir ein Array, das mit CREATE und dem FORTH Wort „," (Komma) angelegt wird:

```
VARIABLE MAXINDEX
:POTENZ, (——)
   (schließt alle Dreierpotenzen ein, die in
   zwei Bytes passen und setzt den Index der
   höchsten Potenz in MAXINDEX.)
—1 MAXINDEX !
1 (kleinste Potenz)
BEGIN
   DUP,( Potenzzahl ins Vokabular aufnehmen )
   1 MAXINDEX +!
   3 UM* ( nächste Potenz, doppelte Länge )
      ( in FORTH-79 und figFORTH UM*
      durch U* ersetzen )
UNTIL (bis die nächste Potenz mehr als 2
      Bytes belegt )
DROP ( niederwertige Bytes der Potenz mit
      doppelter Länge )
;
CREATE 3POTENZ POWERS
      ( ALLOT wird nicht gebraucht, da das
      Komma das Einschließen erledigt )
:3** ( n —— 3**n )
MAXINDEX @ OVER U<IF
   DROP 0 (n liegt nicht zwischen 0 und
      MAXINDEX. Ergibt 0)
   ELSE ( der Speicherinhalt von 3POTENZ
      + 2*n wird gebraucht)
   2 * 3POTENZ +@
THEN
```

3POTENZ ist ein eindimensionales numeri-

Array-Strukturen

Mit CREATE und DOES> lassen sich in FORTH leicht Arrays anlegen. Das Diagramm zeigt Schritt für Schritt, wie FORTH elf 1ARRAY RUNS ausführt. 1ARRAY wurde zuvor im Vokabular als Colon-Definition eingerichtet. Das Bild zeigt auch die Auswirkungen auf den Stapel.

Colon-Definition	Elf „1ARRAY RUNS“	Stapel
: 1ARRAY CREATE	CREATE teilt RUNS ein Namensfeld und ein Codefeld zu. RUNS hinterläßt beim Einsatz nun seine Pfa auf dem Stapel.	11
DUP +	DUP+ multipliziert 11 mit zwei und schiebt das Ergebnis auf den Stapel.	22
HERE	HERE schiebt die nächste verfügbare Adresse des Vokabulars auf den Stapel. Da wir gerade RUNS eingeschlossen haben, ist dies die Pfa von RUNS.	PFA 22
OVER	OVER kopiert 22 oben auf den Stapel (direkt über die Pfa von RUNS).	22 PFA 22
ALLOT	ALLOT zieht 22 vom Stapel und reserviert im Vokabular 22 Bytes, die (da wir gerade RUNS dort eingesetzt haben) das Parameterfeld für RUNS bildet.	PFA 22
SWAP 0 FILL	SWAP setzt 22 über die Pfa von RUNS, schiebt dann 0 auf den Stapel. FILL nimmt die drei Parameter und stellt alle 22 Bytes des Parameterfeldes auf Null.	0 22 PFA → LEER
DOES>	Laufzeit (hinter DOES>)	
	99 2 RUNS ! setzt beispielsweise den Stapel wie im Bild gezeigt und speichert den Wert 99 im Element 2 des Arrays RUNS wie folgt:	PFA 2 99
SWAP	SWAP setzt den Arrayindex (2) an die Stapelspitze.	2 PFA 99
DUP +	DUP+ verdoppelt den Wert ...	4 PFA 99
+	+ addiert den Index (der nun den Offset bildet) auf die Pfa von RUNS ...	PFA +OFF–SET
;		
!	! nimmt 99 und speichert den Wert an der richtigen Adresse.	LEER

sches Array. Sie könnten es ebensogut mit ALLOT und ! anlegen. String-Arrays werden auf ähnliche Weise aufgebaut. Weil dabei jedes Zeichen aber immer nur von einem Byte dargestellt wird, müssen Sie C, , C! und C@ nehmen, statt , ,! und @.

Array-Einsatz

Der Einsatz eines Arrays wie 3POTENZ muß allerdings mit

```
2 * 3POTENZ +
```

gekennzeichnet werden, da 3POTENZ nicht weiß, daß es ein Array ist. Das Wort hinterläßt nur seine Parameterfeldadresse zur weiteren Bearbeitung. Mit CREATE (figFORTH: <BUILDS) und DOES> lassen sich jedoch „intelligentere" Wörter anlegen, CREATE und DOES> definieren neue Wörter, die eigentlich verbesserte Versionen von CREATE – und damit neue Definitionswörter sind. Sie enthalten nicht nur CREATE, sondern führen gleichzeitig auch andere Vorgänge aus. Die neuen Definitionswörter können beispielsweise darüber informieren, welche Aufgaben ein neu angelegtes Wort hat, statt nur die Parameterfeldadresse (pfa) auf den Stapel zu schieben.

```
:1ARRAY ( n —— )
   ( baut ein eindimensionales numerisches
   Array mit der Dimension n auf )
CREATE ( der Name wird beim Einsatz von der
   Position hinter 1ARRAY genommen )
DUP + (ein schneller Weg, 2 * auszuführen)
HERE (beachten Sie die Parameterfeld-
   adresse  für FILL)
OVER ALLOT
   ( jetzt 2*n, pfa)
SWAP 0 FILL
   ( alle Bytes des Parameterfeldes auf
      0 stellen )
   DOES> (Subscript, pfa —— Adresse des
   Elements )
SWAP DUP + (pfa, 2* subscript)
+
;
```

Das Beispiel besteht aus zwei Teilen: Die vor DOES> stehende Definition wird beim Einsatz von 1ARRAY ausgeführt. Sie legt das Parameterfeld des neuen Wortes an.

```
11 1ARRAY RUNS
```

richtet das neue Wort RUNS ein und ordnet ihm mit ALLOT ein aus 22 Bytes bestehendes, und mit Nullen initialisiertes Parameterfeld zu. Der zweite Teil von 1ARRAY (nach DOES>) wird beim Ablauf aktiviert. RUNS schiebt dabei zwar auch hier noch seine Parameterfeldadresse ganz einfach auf den Stapel, setzt dann aber seine „Intelligenz" ein und legt an dieser Stelle elf Variablen mit den Namen:

```
0 RUNS
1 RUNS
:

:
10 RUNS
```

an, die sich wie normale Variablen einsetzen lassen. Hier ein Beispiel:

```
99 2 RUNS !
4 RUNS @.
```

Dieser Einsatz von CREATE...DOES> war recht einfach. Mit etwas mehr Erfahrung können Sie jedoch weitaus komplexere Aufgaben damit erledigen. Bei der Entwicklung von FORTH-Programmen wird in Variablendefinitionen oft das @ vergessen. Mit CREATE und DOES> lassen sich nun Variablen definieren, die ohne @ auskommen. Normalerweise wird dabei nur der Variablenwert (wie bei einer Konstanten) auf dem Stapel zurückgelassen. Wenn Sie den Wert jedoch ändern wollen, entsteht ein Problem, das Sie am besten mit dem Wort → lösen. Hier der Ablauf:

Neuer Wert → Variablenname im neuen Stil → hat nur die Aufgabe, dem System (durch Setzen einer Variablen) mitzuteilen, daß dieses Wort verwendet wurde. Die Variable „im neuen Stil" weiß dadurch jedoch, daß sie einen neuen Wert vom Stack ziehen muß, statt ihren aktuellen Wert auf den Stapel zu schieben.

```
VARIABLE → USED
0 → USED !
: → — 1 → USED!;
```

Wir wollen nun für das Definitionswort VARIABLE ein neues definieren. Das neue Wort ist ebenfalls ein Definitionswort und arbeitet mit

```
CREATE...DOES>.
  : VAR ( — — )
     CREATE
     0, (teilt 2 Bytes zu, die mit 0 initialisiert sind )
  DOES >
     → USED @ IF ( neuer Wert, pfa — — )
     ! (neuen Wert setzen )
     0 → USED!
  ELSE (pfa — — aktueller Wert )
  @
  THEN
  ;
```

Hier die Definition der VAR Variablen x. Sie wird auf 99 gesetzt und der Wert angezeigt:

```
VARx
99 → x
x.
```

CREATE und DOES> lassen sich außerordentlich flexibel einsetzen, da sie zwei Eigenschaften miteinander kombinieren, die normalerweise separat arbeiten. Die meisten Program-

miersprachen enthalten „passive" Daten, die bearbeitet werden und „aktive" Routinen, die die Arbeit ausführen und mit Daten gefüttert werden. Mit CREATE und DOES> aber können Sie intelligente Daten anlegen, die sich während des Programmablaufs selbst bearbeiten. Wenn Sie dieses flexible Konzept einsetzen, müssen Sie jedoch die Trennung zwischen Daten und Programmen vergessen, die bei den meisten anderen Programmiersprachen verlangt wird.

Nützliche Wörter

,(n — —) trägt n mit zwei Bytes im Vokabular ein. Die Ein-Byte Version ist C, .
HERE (— — Adresse) schiebt die Adresse des im Vokabular reservierten Platzes auf den Stapel und kennzeichnet damit die Stelle, an der der nächste Eintrag stattfinden soll.
FILL (Adresse, n, Füllelement — —) füllt von der angegebenen Adresse n Bytes mit dem Füllelement aus.

Der Array-Aufbau

Die mit 1ARRAY definierten Arrays arbeiten schnell, prüfen aber nicht, ob der angegebene Index überhaupt zulässig ist und können so durchaus Werte außerhalb des Arrays überschreiben. Wenn bei 1ARRAY die Dimension am Anfang des Parameterfeldes gespeichert wird, kann der Ablaufteil prüfen, ob ein angegebener Index kleiner ist als die Dimensionierung und eine Fehlermeldung ausgeben, falls dies nicht der Fall ist.

Einige Sprachen wie PASCAL und BASIC überprüfen Arrayindizes automatisch, während andere (beispielsweise C) für hohe Geschwindigkeit ausgelegt sind und keinen Test ausführen. In FORTH können Sie selbst entscheiden, welchen Weg Sie gehen wollen und hierzu dann die Sprache entsprechend programmieren.

Weiterhin setzen einige Programmiersprachen den Index 0 als erstes Element ein, andere dagegen das Element 1 (oder ein anderes). Auch hier läßt Ihnen FORTH weiterhin uneingeschränkt die freie Wahl.

Mehrfachdimensionale Arrays arbeiten nach dem gleichen Prinzip, sind aber komplizierter aufgebaut. Ein mehrfachdimensionales Array stellt man sich normalerweise als dreidimensionalen Block vor. Der Computer speichert ihn Zeile für Zeile in einer langen Liste. Sie können die Indizes als Offsets für diese Liste einsetzen, indem Sie diese mit den Dimensionen multiplizieren.

Für ein dreidimensionales Array mit den Dimensionen 2, 3 und 4 (insgesamt 24 Elemente) lassen sich die Indizes i, j und k folgendermaßen in Offsets umwandeln:

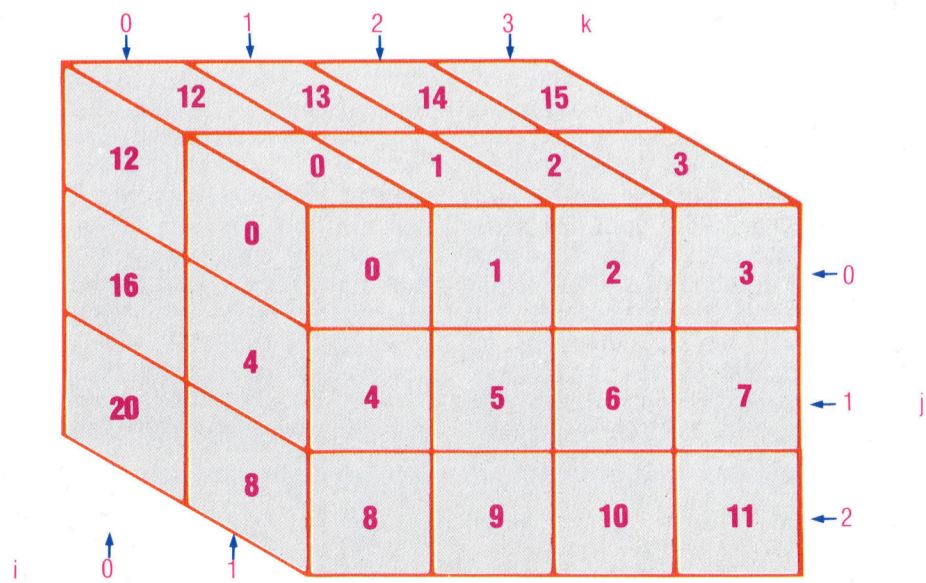
```
(i*3 + j) * 4 + k
```

Hier die Definition von 3ARRAY:

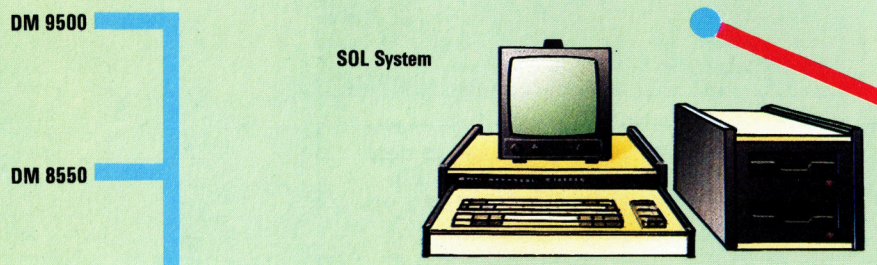
```
: 3ARRAY (d, e, f — —)
   ( legt ein dreidimensionales Array mit den
   Dimensionen d, e und f an)
CREATE
   OVER, DUP, ( e und f eintragen )
   HERE ( d, e, f, pfa+4 )
   2 ROT * ROT * ROT * ( pfa+4, 2*d*e*f )
   DUP ALLOT ( 2*d+e+f Bytes eintragen)
   0 FILL ( mit 0 initialisieren )
DOES> ( i, j, k, pfa — — Adresse )
   DUP@ ( i, j, k, pfa, e )
   4 ROLL * ( j, k, pfa, i*e )
   3 ROLL + ( k, pfa, i*e+j )
   OVER 2+@ ( k, pfa, i*e+j, f )
   * ( k, pfa, [i*e+j]* f )
   ROT + DUP + ( pfa, 2*[[i*e+j]*f+k] )
   SWAP 4 + +
;
```

(Beachten Sie, daß die Zahl vor ROLL in FORTH-79 um eins höher sein muß.)

Nach dieser Methode lassen sich ähnliche Arrays (4ARRAY, 5ARRAY, etc.) anlegen. Es kann sogar ein Wort definiert werden, das eine Zahl vom Stapel zieht und Ihnen mitteilt, wie viele Dimensionen sich auf dem Stapel befinden. Falls Ihnen dies zu kompliziert erscheint, sollten Sie nachlesen, wie die Programmiersprache C Arrays anlegt.



Dynamisch gesteuert



SOL System

In dieser neuen Serie untersuchen wir die Entwicklung, Struktur und die Funktion des weitverbreiteten Betriebssystems CP/M. Nachdem CP/M sich im kommerziellen Bereich schon seit 1975 einen ausgezeichneten Ruf aufgebaut hat, wird es jetzt mehr und mehr auch auf Heimcomputern eingesetzt.

Als in den frühen siebziger Jahren die ersten Diskettensysteme auf den Markt kamen, gab es noch kein Betriebssystem, das die auf Diskette gespeicherten Informationen organisieren konnte. Der amerikanische Microchiphersteller Intel entschied sich daher, für seine kurz zuvor eingeführte Prozessorserie 8000 ein Diskettensystem zu entwickeln. Chef des Entwicklungsteams war der junge Ingenieur Gary Kildall. Das fertige Betriebssystem wurde unter der Bezeichnung CP/M (zunächst die Abkürzung für „Control Program/Monitor", später umbenannt in „Control Program for Microprocessors" = Steuerprogramm für Microprozessoren) bekannt.

Schon kurz nach der Vorstellung im Jahre 1975 war das Paket so erfolgreich, daß eine Gruppe von Konstrukteuren Intel verließ, die Firma Zilog Inc. gründete und einen eigenen Chip – den Z80 – entwickelte. Der Z80 war mit dem 8008 kompatibel und konnte ebenfalls mit CP/M arbeiten. Auch Kildall gründete seine eigene Firma, Digital Research, die inzwischen zu den größten Softwarefirmen der Welt zählt.

Der Erfolg läßt sich daran messen, daß CP/M schon während der siebziger Jahre zu einem Industriestandard wurde – trotz vieler konkurrierender Betriebssysteme für Acht-Bit-Prozessoren (speziell der nicht CP/M-kompatiblen Systeme mit dem 6502 Prozessor). Besonders erstaunlich ist die im Maßstab der Microtechnologie außerordentlich lange Marktpräsenz von CP/M (nun schon über zehn Jahre).

Ein für CP/M geeigneter Computer muß über einen 8008 Microprozessor (oder einen kompatiblen Chip), ein Diskettenlaufwerk und mindestens 48 KByte RAM verfügen. Der große Arbeitsspeicher ist notwendig, da im RAM außer den Diskettendateien auch das CP/M-Programm und eine der CP/M-Befehlsdateien (bis acht KBytes) Platz finden müssen. Mit der Einführung der 16-Bit-Computer scheint das System im kommerziellen Bereich nun ausgedient zu haben. CP/M wurde inzwischen jedoch auch von den Heimcomputer-Herstellern weitgehend erfolgreich übernommen. Durch den Einbau einer CP/M-kompatiblen Diskettenstation steht dem Anwender sofort die riesige Softwarebasis zur Verfügung, die ursprünglich für die kommerziellen Micros der siebziger und frühen achtziger Jahre entwickelt wurde. Die Computer-Hersteller umgehen damit das Problem, eigene Software produzieren zu müssen. Zu den Firmen, die diesen Weg in den letzten Jahren eingeschlagen haben, gehören Memotech, Research Machines und Amstrad (Schneider).

Im Gegensatz zu den meisten Heimcomputern mit fest eingebauten Betriebssystemen (die beim Einschalten aktiviert werden), ist CP/M fast immer auf einer (System-)Diskette untergebracht und wird beim Anschalten des Rechners von dort geladen. Einige Hersteller bieten CP/M aber auch als ROM-Version an. Der Ursprung der Diskettenversion ergibt sich aus der Geschichte der Microcomputer. Auf den frühen Computermodellen war der Speicherplatz knapp bemessen. Ein installiertes, aber nicht eingesetztes CP/M hätte Platz belegt, der für andere Zwecke nicht mehr zur Verfügung stünde. Das System wurde daher „ausgelagert" und nur bei Bedarf geladen.

Eine typische CP/M-Systemdiskette enthält das CP/M-Programm und eine Reihe kleinerer Module, die über CP/M aufgerufen werden und spezielle Diskettenfunktionen ausführen.

CP/M ist aber nicht nur für den Diskettenzugriff zuständig – es ist ein völlig eigenständiges Betriebssystem.

Wenn CP/M in einen Computer mit einem (anderen) im ROM gespeicherten Betriebssystem geladen wird, überlagert es das eingebaute System und übernimmt die Steuerung der gesamten Maschine. Normale BASIC-Befehle verursachen nun eine Fehlermeldung, da sie von CP/M nicht erkannt werden. BASIC-Programme funktionieren unter CP/M nur mit einem BASIC-Interpreter oder Compiler.

Im Normalfall lädt sich CP/M beim Einschalten des Computers automatisch in die Maschine. Der Bildschirm zeigt ein Directory (Diskettenihnhalt) mit einigen Systembefehlen. Bei genauerer Betrachtung gibt uns das Directory Einblick in die Struktur von CP/M. So haben die Systembefehle (wie alle CP/M-Dateien) einen zweiteiligen Namen. Bei Aufruf eines Befehls braucht nur der erste Teil eingegeben zu werden, gefolgt vom Namen der Datei, auf die sich der Befehl bezieht.

```
WRENMENU version 1.1
(c) 1984 Quantec Systems and Software Ltd
A)dir
A: WRENMENU COM : CPM3     SYS : COPYSYS  COM : ERASE   COM : SET     COM
A: SHOW     COM : SUBMIT   COM : TYPE     COM : DIR     COM : GET     COM
A: PUT      COM : RENAME   COM : SETDEF   COM : PIP     COM : HELP    COM
A: HELP     HLP : FC       COM : DEVICE   COM : FORMAT  SUB : FC      GET
A: FORMAT   GET : CONFIGUR COM
A>
```

Das Bild zeigt eine Liste der CP/M-Befehlsdateien, die auf der Systemdiskette gespeichert sind. Alle Systembefehle sind hinter dem Dateinamen durch die Erweiterung .COM gekennzeichnet.

Der zweite Teil des Dateinamens (rechts vom Punkt) wird „Erweiterung" genannt und teilt CP/M (und Ihnen) mit, welcher Dateityp vorliegt. Alle Befehlsdateien der CP/M-Systemdiskette haben die Erweiterung „.COM". Wir werden später in der Serie auch auf andere Arten von Erweiterungen eingehen.

Wir hatten in früheren Artikeln schon erwähnt, daß Betriebssysteme so einfach wie möglich aufgebaut und leicht zu bedienen sein sollten. CP/M erfüllt diese Anforderungen. Der Bootvorgang lädt CP/M automatisch in den Arbeitsspeicher, wo es bis zum Abschalten des Computers bleibt. Im RAM wird außerdem eine Reihe oft eingesetzter Hilfsmodule gespeichert. Diese Module müssen beim Abruf nun nicht erst von der Diskette nachgeladen werden. Sie sind – ebenso wie das CP/M-Programm – für das Directory „unsichtbar".

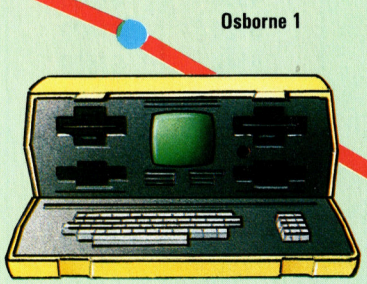
Die auf Diskette gespeicherten Befehlsmodule werden „Diskettenbefehle" (oder „transient" = flüchtig) genannt. Da CP/M sie nach der Ausführung automatisch löscht, müssen sie für jeden Einsatz neu geladen werden.

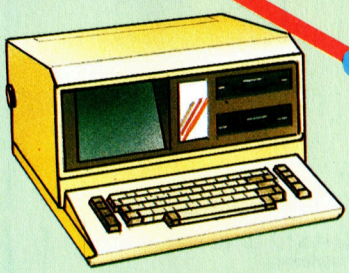
Dieser auf den ersten Blick recht umständliche Aufbau gibt dem System jedoch große Flexibilität und berücksichtigt bestehende Hardwaregrenzen. CP/M ist auf Acht-Bit-Prozessoren ausgerichtet, die nur 64 KBytes direkt adressieren können. Wenn alle Befehle, die zum Funktionieren von CP/M notwendig sind, ständig im Speicher wären, hätte der Anwender kaum Platz für Programme und Daten. Die Designer von CP/M luden daher nur die wichtigsten Befehle ins RAM.

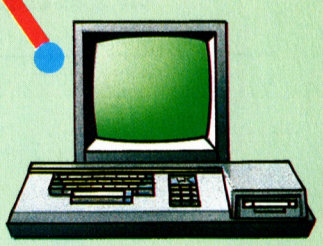
Ein System dieser Art hat viele Vorteile. So lassen sich beispielsweise zusätzliche Systembefehle (im Format .COM) auf der Diskette anlegen. Auf der Grundlage dieser Flexibilität entstand eine große Menge von Anwenderprogrammen, die unter CP/M laufen. Beispielsweise wird das bekannte Textverarbeitungssystem WordStar (obwohl es ein separates Programm ist) von CP/M wie ein Diskettenbefehl behandelt.

WordStar läßt sich damit auf jedem CP/M System einsetzen. Es benötigt keine eigenen Routinen zur Diskettenverwaltung, sondern ruft einfach die CP/M-Funktionen auf. WordStar braucht dadurch weit weniger Speicherplatz, als bei einer zusätzlichen Diskettenverwaltung nötig wäre.

In der nächsten Folge gehen wir genauer auf einige Dienstprogramme von CP/M ein.


Osborne 1


Wren


Schneider CPC 664

1981 1982 1983 1984 1985

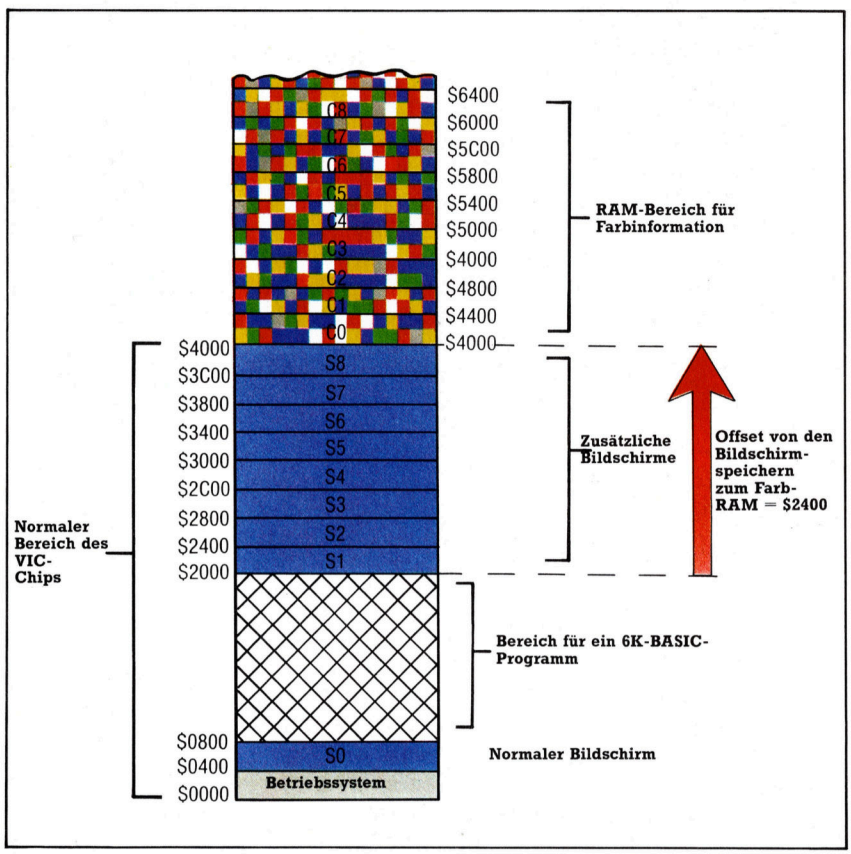
Bildfolgen

Der „Video Controller"-Chip (VIC) des Commodore 64 kann 16K RAM ansprechen. Normalerweise sind dies die ersten 16 K zwischen $ 0000 und $3FFF. Durch die Umstellung bestimmter Registerwerte des VIC können jedoch auch die anderen drei 16 K Blöcke zugänglich gemacht werden. Das Programm „Bildschirmwechsel" legt in den 16 K, die dem VIC normalerweise zur Verfügung stehen, „alternative" Bildschirmspeicher an. Die Farbinformationen jedes Bildschirms liegen direkt über diesem 16K-Bereich.

Auf dem Commodore läßt sich der Bildschirmspeicher problemlos in einen anderen RAM-Bereich verlegen. Wir sehen uns eine Maschinencoderoutine an, die bis zu acht Bildschirminhalte anlegen und zur Weiterverarbeitung speichern kann.

Auf dem Commodore 64 steuert der Spezialchip VIC II die Bildschirmanzeige und die Sprite-Verarbeitung. Dazu ruft der VIC-Chip Informationen aus verschiedenen Speicherbereichen ab, darunter vom Zeichen-ROM die Formen der Zeichen und vom Farb-RAM die Farbinformation. Der Bildschirmspeicher gibt an, welches Zeichen an den 1000 Bildschirmpositionen (25 Zeilen * 40 Spalten) dargestellt werden soll.

Beim Einschalten des Commodore 64 nimmt der VIC-II-Chip an, daß der Bildschirminhalt in 1000 Bytes Länge als Bildschirmspeicher interpretiert werden soll. Die Tabelle zeigt die Bitwerte, die den 16 möglichen Bildschirmpositionen entsprechen.



Bitmuster	Anfang Bildschirminhalte	
0000XXXX	0	$0000
0001XXXX	1024	$0400 *
0010XXXX	2048	$0800
0011XXXX	3072	$0C00
0100XXXX	4096	$1000
0101XXXX	5120	$1400
0110XXXX	6144	$1800
0111XXXX	7168	$1C00
1000XXXX	8192	$2000
1001XXXX	9216	$2400
1010XXXX	10240	$2800
1011XXXX	11264	$2C00
1100XXXX	12288	$3000
1101XXXX	13312	$3400
1110XXXX	14336	$3800
1111XXXX	15360	$3C00

* Systemeinstellung

den 1000 Bytes gespeichert ist, die bei Adresse 1024 ($0400) anfangen. Seine erste Bildschirminformation holt er sich daher aus diesem Bereich. Durch eine Änderung eines Registerwertes im Inneren des VIC-Chips können wir jedoch veranlassen, daß der Chip auf einen anderen Bereich – gewöhnlich auf die ersten 16 KByte des Speichers – zugreift. Die oberen vier Bits des VIC-Steuerregisters bei 53272 ($D018) bestimmen, welcher Block von

Für die Verlegung des Bildschirmspeichers an eine andere Stelle müssen wir die oberen vier Bits der Adresse 53272 ($D018) auf die Tabellenwerte setzen. Die unteren vier Bits dürfen nicht verändert werden (in der Tabelle durch XXXX gekennzeichnet), da sie eine andere Funktion steuern. Um auf die oberen vier Bits zugreifen zu können, ohne den Wert der unteren vier Bits zu verändern, müssen wir mit dem Inhalt des Registers ein AND mit 15 (binär 00001111) ausführen. Danach können wir mit OR den neuen Registerinhalt speichern. Für die Verlegung des Bildschirms in den letzten für den VIC zugänglichen Speicherbereich – Anfang bei 15360 ($3C00) – muß mit dem Inhalt des Registers ein OR mit 240 (binär 11110000) ausgeführt werden. Diese Aufgabe erledigt folgende BASIC-Zeile:

```
POKE 53272,(PEEK(53272) AND 15) OR 240
```

Bevor wir nun aber in den neuen Bildschirmspeicher schreiben können, müssen wir dem Betriebssystem des Commodore 64 mitteilen, daß sich die Position des Speichers verschoben hat. Das wird erreicht, indem wir das Hi-Byte der Anfangsadresse des neuen Speicherbereiches in die Adresse 648 ($0288) schrei-

ben. Als höchstmögliche Adresse muß hier $3C eingetragen werden (für den entsprechenden BASIC-Wert muß nur die Startadresse durch 256 geteilt werden).

Nachdem beide Register geändert wurden, können Sie wie gewohnt mit dem Speicher arbeiten. Es lohnt sich, für diesen Ablauf ein kurzes BASIC-Programm zu schreiben.

Durch die Verlegung des Bildschirmspeichers lassen sich interessante Effekte hervorrufen, da der Bildschirminhalt jetzt leicht ausgetauscht werden kann. Es muß allerdings auch das Farb-RAM versetzt werden, da es wichtige Informationen für den Bildschirmaufbau liefert. Nun lassen sich zwar mehrere Bildschirmbereiche einrichten, doch gibt es nur einen Bereich für Farbdaten, der sich auch nicht verlegen läßt. Wir müssen daher 1000 Bytes für die Farbinformationen reservieren. Beim Wechsel des Bildschirminhalts werden diese Bytewerte dann in das Farb-RAM kopiert und dessen bisheriger Inhalt gesichert.

Hauptaufgabe dieser Hilfsroutine ist es, den Speicher so zu organisieren, daß er zusätzliche Bildschirminhalte (mit ihren Farbdaten) aufnehmen kann und außerdem die Verschiebung der Speicherblöcke vornimmt. Da dem VIC-Chip 16 KBytes zugänglich sind, können wir mit unserem System bis zu acht Bildschirminhalte über ein BASIC-Programm steuern. Das Ablaufdiagramm zeigt, wie die Routine den Speicher anspricht.

Um sicherstellen zu können, daß kein Bildschirm oder Farbspeicher von BASIC-Programmen überschrieben wird, muß im Speicher die Obergrenze für BASIC-Programme mit folgendem Befehl heruntergesetzt werden:

POKE 55,0: POKE 56,32:CLR

Die Basisadresse jedes Bildschirminhalts kann nun mit dieser Zahl und der Formel

Bildschirmbasis = $1C00 + ($0400 * Nummer des Bildschirms)

berechnet werden. Für die Basisadresse des entsprechenden Farb-RAMs wird einfach ein Offset auf die Basisadresse des Bildschirmspeichers addiert. Die Formel lautet:

Farbbasis = $2400 + Bildschirmbasis

Neue Rahmenfarbe

Die Farbbereiche könnten zwar an jeder beliebigen Stelle des RAM liegen, doch bietet es sich an, sie direkt über den Bildschirmbereich zu setzen, den der VIC-Chip ansprechen kann. Beachten Sie, daß auch für den Bildschirm 0 ein Farbbereich abgestellt wird, dessen Farb-RAM ein unterschiedliches Offset hat.

Das VIC-Steuerregister und das Betriebssystemregister lassen sich auf folgende Weise einstellen:

VIC-Register = $70 + ($10 * Nummer des Bildschirms)
OS-Register = HI-Byte der Bildschirm-Basisadresse

Die Routine setzt aber nicht nur Register oder verschiebt Farbdaten, sondern legt auch die Farbe des Vordergrundes und des Rahmens fest. Diese beiden Parameter werden von einem Registerpaar des VIC gesteuert: 53280 ($D020) legt die Farbe des Rahmens fest und 53281 ($D021) die des Bildschirmhintergrundes. Unsere Routine richtet dafür eine eigene Tabelle ein.

Hilfreiches Flag

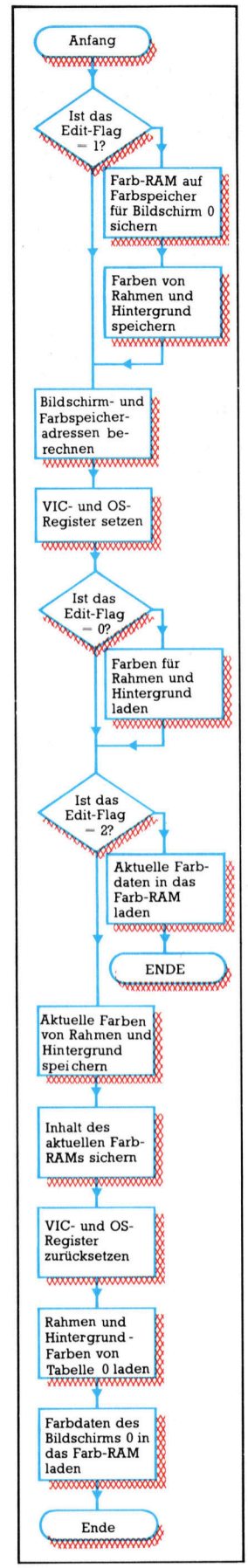
Die Routine hat zwei Betriebsarten: Sie kann einen bestimmten Bildschirminhalt entweder anzeigen oder editieren. In jedem Fall muß aber die Nummer des gewünschten Bildschirms mit POKE in 49152 ($C000) gesetzt werden. Um die Routine mit dem gleichen SYS-Aufruf ansprechen zu können, zeigt ein Flag an, welche Betriebsart gemeint ist. Das Flag wird durch ein POKE in die Adresse 49153 ($C001) gesetzt.

0 = anzeigen
1 = editieren

Der Editiermodus arbeitet mit dem normalen Bildschirmeditor und kann daher auch die Textfarben ändern, die Negativdarstellung setzen und den Bildschirm löschen. Dazu wird die Routine zuerst aufgerufen und das Edit-Flag auf Eins gesetzt. Nach Sicherung der normalen Farbdaten und der Hintergrund- und Rahmenfarben setzt die Routine die Register des VIC und des Betriebssystems und kehrt dann zum BASIC zurück. Das aufrufende BASIC-Programm übernimmt nun die Steuerung, stellt den Cursor in die linke obere Ecke und aktiviert den BASIC-Befehl INPUT. Dieser Befehl wartet, bis er ein Return (ASCII-Zeichen 13) erhält und ruft dann das restliche Programm auf. In der Zwischenzeit kann der Bildschirminhalt jedoch normal editiert werden.

Nach Beendigung des Editierens wird die Routine ein zweites Mal aufgerufen und das Edit-Flag auf 2 gesetzt. Dadurch werden vor dem Aufbau des normalen Bildschirms die Farbdaten und die Rahmen- und Hintergrundfarben gespeichert. Wenn nun die beiden letzten Parameter geändert werden sollen, müssen die entsprechenden Farbcodes mit POKE in die Speicherstellen 49154 ($C002) und 49155 ($C003) gesetzt werden.

Das BASIC-Programm stellt ein Menü dar, das die Wahl zwischen Anzeigen oder Editieren anbietet. Die Anzeigeroutine ruft auf Tastendruck jeden der acht unterschiedlichen Bildschirme nacheinander auf und wiederholt diesen Vorgang so lange, bis Return eingegeben wird. Das Programm stellt dann den normalen Bildschirminhalt wieder her und kehrt zum Menü zurück. Bei Wahl der Editieroption kann der Anwender für den gewählten Bildschirminhalt die Farben für Hintergrund und Rahmen angeben und dann mit dem Bildschirmeditor ein Bild erzeugen.



Ladeprogramm in BASIC

```
10 REM ************************
20 REM ** BASIC LOADER FOR  **
30 REM ** ALTERNATE SCREENS **
40 REM ************************
50 :
60 FORI=49152 TO 49439
65 READA:POKE I,A
70 CC=CC+A
75 NEXT
80 READ CS:IFCS<>CC THENPRINT"CHECKSUM ERROR":STOP
100 DATA255,255,0,0,191,255,0,0,255
110 DATA191,0,0,191,191,0,0,255,191,0
120 DATA0,255,191,0,0,255,255,0,173,1
130 DATA192,201,1,208,30,169,0,141,6
140 DATA192,169,64,141,7,192,32,224
150 DATA192,162,0,32,246,192,173,2,192
160 DATA141,32,208,173,3,192,141,33
170 DATA208,173,0,192,208,6,32,32,193
180 DATA76,139,192,169,0,141,4,192,141
190 DATA5,192,174,0,192,173,5,192,24
200 DATA105,4,202,208,250,24,105,28
210 DATA141,5,192,24,105,36,141,7,192
220 DATA173,0,192,162,4,10,202,208,252
230 DATA24,105,112,141,8,192,173,24
240 DATA208,41,15,13,8,192,141,24,208
250 DATA173,5,192,141,136,2,173,1,192
260 DATA208,6,174,0,192,32,211,192,201
270 DATA2,240,4,32,189,192,96,174,0
280 DATA192,32,246,192,32,224,192,32
290 DATA32,193,169,0,141,6,192,169,64
300 DATA141,7,192,162,0,32,211,192,32
310 DATA189,192,96,173,6,192,133,251
320 DATA173,7,192,133,252,169,0,133
330 DATA253,169,216,133,254,32,3,193
340 DATA96,189,9,192,141,32,208,189,18
350 DATA192,141,33,208,96,173,6,192
360 DATA133,253,173,7,192,133,254,169
370 DATA0,133,251,169,216,133,252,32,3
380 DATA193,96,173,32,208,157,9,192
390 DATA173,33,208,157,18,192,96,162,3
400 DATA160,0,177,251,145,253,136,208
410 DATA249,230,252,230,254,202,48,10
420 DATA208,240,177,251,145,253,160
430 DATA231,208,232,96
440 DATA36606:REM*CHECKSUM*
```

Bildschirmwechsel

```
3 REM  ********************************
5 REM  **                            **
10 REM **    ALTERNATE SCREENS       **
11 REM **                            **
12 REM ********************************
13 :
15 DN=8:REM FOR CASS DN=1
20 IFA=0 THEN A=1:LOAD"ALT SCREENS.HEX",DN,1
25 POKE55,0:POKE56,32:CLR:REM LOWER MEMTOP
40 SCNUMB=49152:  REM SCREEN NUMBER
70 EDITFG=49153:  REM 0=DISPLAY SCREEN
75 :              REM 1=EDIT SCREEN
77 :              REM 2=COPY C.RAM
80 ALT=49179:     REM M/C START ADDRESS
85 BRDCOL=49154:  REM BORDER COLOUR
87 PAPCOL=49155:  REM PAPER COLOUR
90 :
95 REM **** MAIN MENU ****
96 :
100 PRINTCHR$(147):REM CLEAR SCREEN
105 PRINTCHR$(154):REM LT BLUE LETTERS
110 DN$="■":REM Q = CURSOR DOWN CHARS
130 PRINTTAB(8);DN$;"CBM 64 ALTERNATE SCREENS"
135 PRINTTAB(8);"------------------------"
140 PRINTTAB(5);DN$;"F1 - EDIT PICTURE"
160 PRINTTAB(5);DN$;"F3 - DISPLAY PICTURE SEQUENCE"
200 :
210 GETA$:IFA$=""THEN210:REM AWAIT KEYPRESS
220 IFA$="■" THEN GOSUB1000
230 IFA$="■" THEN GOSUB 1500
260 GOTO100
999 :
1000 REM **** EDIT SCREEN ****
1002 :
1005 EF=1:REM SET EDIT MODE
1010 PRINTCHR$(147)
1020 PRINTTAB(16);DN$;"EDIT MODE"
1030 PRINTDN$;:INPUT"SCREEN NUMBER";SN$
1050 IFASC(SN$)<48 OR ASC(SN$)>56 THEN 1030
1060 PRINTDN$;:INPUT"BORDER COLOUR";BC$
1070 IFVAL(BC$)=0 AND BC$<>"0"THEN 1060
1080 PRINTDN$;:INPUT"PAPER COLOUR";PC$
1090 IFVAL(PC$)=0 AND PC$<>"0"THEN 1080
1100 :
1110 POKEEDITFG,EF
1120 POKESCNUMB,VAL(SN$)
1130 POKEBRDCOL,VAL(BC$)
1140 POKEPAPCOL,VAL(PC$)
1150 SYS ALT
1155 REM ** WAIT FOR RETURN **
1162 INPUT"■";X$:  REM S =HOME CURSOR
1165 REM ** SAVE SCREEN **
1170 EF=2
1175 POKE EDITFG,EF
1180 SYS ALT
1185 RETURN
1190 :
1500 REM **** DISPLAY SCREEN ****
1510 EF=0
1520 PRINT CHR$(147)
1545 SN=1
1550 POKE EDITFG,EF
1555 POKE SCNUMB,SN
1560 SYS ALT
1570 GET X$:IFX$=""THEN1570:REM AWAIT KEYPRESS
1572 IFX$=CHR$(13)THEN 1580:REM NORMAL SCREEN
1575 SN=SN+1:IFSN<9 THEN 1555
1577 SN=1:GOTO1555
1580 POKESCNUMB,0:SYS ALT
1600 RETURN
```

Commodore 64

```
;++++++++++++++++++++
;++++++++++++++++++++
;++                ++
;++    ALTERNATE   ++
;++     SCREENS    ++
;++       FOR      ++
;++     CBM  64    ++
;++                ++
;++++++++++++++++++++
;++++++++++++++++++++
;
FROM    =$FB            ;0 PAGE
TO      =$FD            ;POINTERS
;
CRAMLO=$00              ;START OF COLOUR RAM
CRAMHI=$D8
NCOLLO=$00              ;NORMAL COLOUR
NCOLHI=$40              ;BASE ADDRESS
NSCRHI=$04              ;NORMAL SCREEN BASE ADDRESS
NVCPOK=$10              ;NORMAL VIC REG VALUE
;
BLOCKS=$03              ;NO OF 256 BYTE BLOCKS
EXTRA   =$E7            ;EXTRA AMOUNT TO 1000 BYTES
;
COLOFF=$24              ;COLOUR OFFSET HIBYTE
SCROFF=$1C              ;SCREEN OFFSET HIBYTE
VICOFF=$70              ;VIC CTRL REG OFFSET
VCMASK=$0F              ;VIC CTRL REG LOBYTE MASK
VICREG=$D018            ;SCREEN LOCATION CONTROL REG
EDREG=$0288             ;SCREEN EDITOR KERNAL REG
BORDER=$D020            ;BORDER COLOUR REG
PAPER   =$D021          ;BACKGROUND COLOUR REG
;
*=$C000                 ;SET LOAD POINTER
;
SCNUMB  *=*+1           ;SCREEN NUMBER
EDITFG  *=*+1           ;EDIT MODE FLAG
BRDCOL  *=*+1           ;BORDER COLOUR
PAPCOL  *=*+1           ;PAPER COLOUR
;
SCBASE  *=*+2           ;SCREEN BASE STORAGE
CLBASE  *=*+2           ;COLOUR BASE STORAGE
VPOKE   *=*+1           ;TEMP STORE FOR VIC NUMBER
BRDTAB  *=*+9           ;BORDER COLOURS TABLE
PAPTAB  *=*+9           ;PAPER COLOURS TABLE
;
;++++ SAVE SCREEN 0 ++++
        LDA EDITFG
        CMP #$01        ;IF 0 OR 2
        BNE CALC        ;THEN DON'T SAVE
;
        LDA #NCOLLO
        STA CLBASE
        LDA #NCOLHI
        STA CLBASE+1
        JSR SAVE        ;SAVE CRAM TO C0
;
        LDX #$00
        JSR SAVEBP      ;SAVE B/P REGS
;
        LDA BRDCOL
        STA BORDER      ;SET NEW B/P
        LDA PAPCOL      ;COLOURS
        STA PAPER
;
;++++ CALCULATE COLOUR BASE ++++
;
CALC
        LDA SCNUMB
        BNE NOZERO
        JSR RESET       ;SET NORMAL REGS
        JMP TESTFG
;
NOZERO
        LDA #$00
        STA SCBASE
        STA SCBASE+1    ;INIT SCBASE
;
        LDX SCNUMB      ;LOAD SCREEN NUMBER
        LDA SCBASE+1    ;HI BYTE ONLY
MULT
        CLC
        ADC #$04
        DEX
        BNE MULT
;
        CLC
        ADC #SCROFF     ;ADD SCREEN OFFSET
        STA SCBASE+1
;
        CLC
        ADC #COLOFF     ;ADD COLOUR OFFSET
        STA CLBASE+1
;
;++++ SET VIC AND EDITOR REGISTERS ++++
        LDA SCNUMB
        LDX #$04
MORE
        ASL A
        DEX
        BNE MORE        ;MULT BY 16
;
        CLC
        ADC #VICOFF     ;ADD OFFSET
        STA VPOKE
        LDA VICREG
        AND #VCMASK
        ORA VPOKE
        STA VICREG      ;SET VIC REG
;
        LDA SCBASE+1
        STA EDREG       ;SET EDITOR REG
;
;++++ TEST STATUS OF EDIT FLAG ++++
TESTFG
        LDA EDITFG
        BNE NOLOAD      ;IF 0 DISPLAY MODE
        LDX SCNUMB
        JSR LOADBP      ;LOAD B/P TO REGS
;
;
NOLOAD
        CMP #$02
        BEQ CONT        ;IF 2 THEN SAVE RAM
;
        JSR LOAD        ;LOAD COL TO CRAM
        RTS
;
CONT
        LDX SCNUMB
        JSR SAVEBP      ;SAVE BP REGS
        JSR SAVE        ;SAVE CRAM TO COL
        JSR RESET       ;SET NORMAL REGS
        LDA #NCOLLO
        STA CLBASE      ;LOAD CBASE WITH C0 BASE
        LDA #NCOLHI
        STA CLBASE+1
        LDX #$00
        JSR LOADBP      ;RELOAD ORIG BORD/PAP COLS
        JSR LOAD        ;LOAD NRML SCR COLS
        RTS
;
;+++++ TRANSFER TO RAM S/R ++++
LOAD
        LDA CLBASE
        STA FROM        ;LOAD 0 PAGE PTRS
        LDA CLBASE+1
        STA FROM+1
;
        LDA #CRAMLO
        STA TO
        LDA #CRAMHI
        STA TO+1
;
        JSR COPY        ;COPY RAM AREA
        RTS
;
;++++ LOAD BORDER & PAPER COLS S/R ++++
LOADBP
        LDA BRDTAB,X
        STA BORDER
        LDA PAPTAB,X
        STA PAPER
        RTS
;
;++++ TRANSFER FROM RAM S/R ++++
SAVE
        LDA CLBASE
        STA TO          ;LOAD 0 PAGE PTRS
        LDA CLBASE+1
        STA TO+1
;
        LDA #CRAMLO
        STA FROM
        LDA #CRAMHI
        STA FROM+1
;
        JSR COPY        ;COPY RAM AREA
        RTS
;
;++++ SAVE BORDER & PAPER COLS S/R ++++
SAVEBP
        LDA BORDER
        STA BRDTAB,X
        LDA PAPER
        STA PAPTAB,X
        RTS
;
;++++ COPY 1000 BYTES S/R ++++
;
COPY
        LDX #BLOCKS
        LDY #$00
NEXT
        LDA (FROM),Y
        STA (TO),Y
        DEY
        BNE NEXT
NXBLOC  INC FROM+1
        INC TO+1
        DEX
        BMI FINISH
        BNE NEXT
        LDA (FROM),Y
        STA (TO),Y
        LDY #EXTRA      ;EXTRA BYTES
        BNE NEXT
FINISH
        RTS
;
;++++ RESET VIC AND EDIT REGS S/R ++++
RESET
        LDA #NCOLLO
        STA CLBASE
        LDA #NCOLHI
        STA CLBASE+1
        LDA VICREG
        AND #VCMASK
        ORA #NVCPOK
        STA VICREG      ;RESTORE VIC REG
        LDA #NSCRHI
        STA EDREG       ;RESTORE EDREG
        RTS
```

Technik macht Druck

Viele Zeitungen benutzen zur Satzerstellung noch heute Maschinen, die als „Linotype" bekannt sind. Die 1886 erfundene Linotype stellt aus Bleilettern Satzzeilen her. Aus Gründen der Kostenreduzierung findet Computertechnologie vor allem bei Lokalzeitungen Anwendung. Die neuen Maschinen erlauben elektronische Texterstellung.

Der Einsatz elektronischer Technologie im industriellen Bereich nimmt ständig an Bedeutung zu. Am Beispiel der Druckindustrie zeigen wir die Problematik auf.

Die Behauptung, daß die Computertechnologie unsere Lebens- und Arbeitswelt radikal verändert, ist mittlerweile zu einem Klischee geworden. Besonders deutlich wird diese Entwicklung im Verlagsbereich. Trotz bedeutender technologischer Fortschritte im Bereich der Textverarbeitung und des computerisierten Layouts findet man bei kleinen Tageszeitungen mancherorts noch heute Maschinen, die aus den dreißiger Jahren stammen. Nicht nur die Kosten für neue Technologie begünstigen das Nebeneinander von Alt und Neu. Auch die in diesem Berufszweig traditionell stark gewerkschaftlich organisierte Arbeiterschaft macht sich gegen Rechnereinsatz stark, der reihenweise Arbeitsplätze wegrationalisiert.

Was vermag die Technik überhaupt? Viele Autoren haben ihre Schreibmaschinen durch Mikrocomputer ersetzt. Sie erlauben es ihnen, Texte in einem Arbeitsgang zu schreiben und zu korrigieren, die Arbeitsleistung zu erhöhen und saubere, preiswerte Manuskripte schnell für Verleger zu erstellen.

Ob man einen Rundbrief an Clubmitglieder oder einen Zeitungsartikel schreiben will, ein transportabler Computer ist ein leistungsfähiges Hilfsmittel. Aus vor Ort getippten Notizen wächst ein Manuskript, nach Bedarf versehen mit Fußnoten, automatischer Paginierung, Kapitelüberschriften und Randbemerkungen – alles kann vor dem Ausdruck eingefügt werden, um ein sauberes, einwandfreies Manuskript zu erstellen. Es ist der erste Schritt auf dem langen Weg zum gedruckten Endprodukt.

Bei den meisten heute produzierten Magazinen finden nch immer vorwiegend traditionelle Produktionsprozesse Anwendung. Üblicherweise wird der Autor eines Artikels aufgefordert, ein maschinengeschriebenes Manuskript einzureichen. Es wird dann in der Redaktion überarbeitet, redigiert. Ein Redakteur korrigiert das Manuskript auf inhaltliche Richtigkeit. Das Manuskript wird dann an einen Korrektor weitergeleitet, besser einen weiteren Redakteur, der wiederum handschriftliche Korrekturen anfügt und Teile umschreibt, damit der Stil dem Verlagshaus entspricht. Das Layout nimmt die Auszeichnung

für die Setzmaschinen vor, um dem Setzer Angaben über Schriftart, Größe, Durchschuß und Zeilenbreite zu geben. Sie erfolgt ebenfalls handschriftlich auf dem Manuskript des Autoren. Aber damit nicht genug.

Fahnenkorrektur

Ist der Artikel gesetzt, ähnelt er fast der druckfertigen Form. Es können allerdings noch Fehler darin stecken, etwa wenn die Setzer bei der Übertragung Fehler gemacht haben. Der abgezogene Artikel wird als erster Fahnenabzug bezeichnet. Der Produktionsredakteur liest ihn wieder sorgfältig und sendet ihn korrigiert an den Satz zurück. Daraufhin wird eine zweite Version erstellt, der zweite, nun fehlerfreie Abzug.

Der Layouter greift wiederum beim Eintreffen der ersten Fahne in den Produktionsprozeß ein. Seine Aufgabe ist es, mit dem Abzug ein Seitenlayout zu erstellen, die Form zu schaffen, in der der Artikel gedruckt erscheint. Natürlich enthalten die meisten Magazine nicht nur Text, sondern auch Fotos und Zeichnungen, zu denen Bildunterschriften und Anmerkungen gehören. Der Layouter muß all diese Komponenten zu einer in sich geschlossenen und zugleich interessant wirkenden Seite komponieren, die dem Auge etwas bietet.

Üblicherweise wird dafür zuerst ein „Scribble", eine Rohskizze, angefertigt, in der die Einzelelemente wie Steine eines Puzzles verschoben werden. Dabei wird zuweilen auch der erste Fahnenabzug ins Layout integriert. Manche Layouter schreiben einfach die Textlänge hinein, nachdem sie nachgemessen haben, wieviel Raum der Text beansprucht.

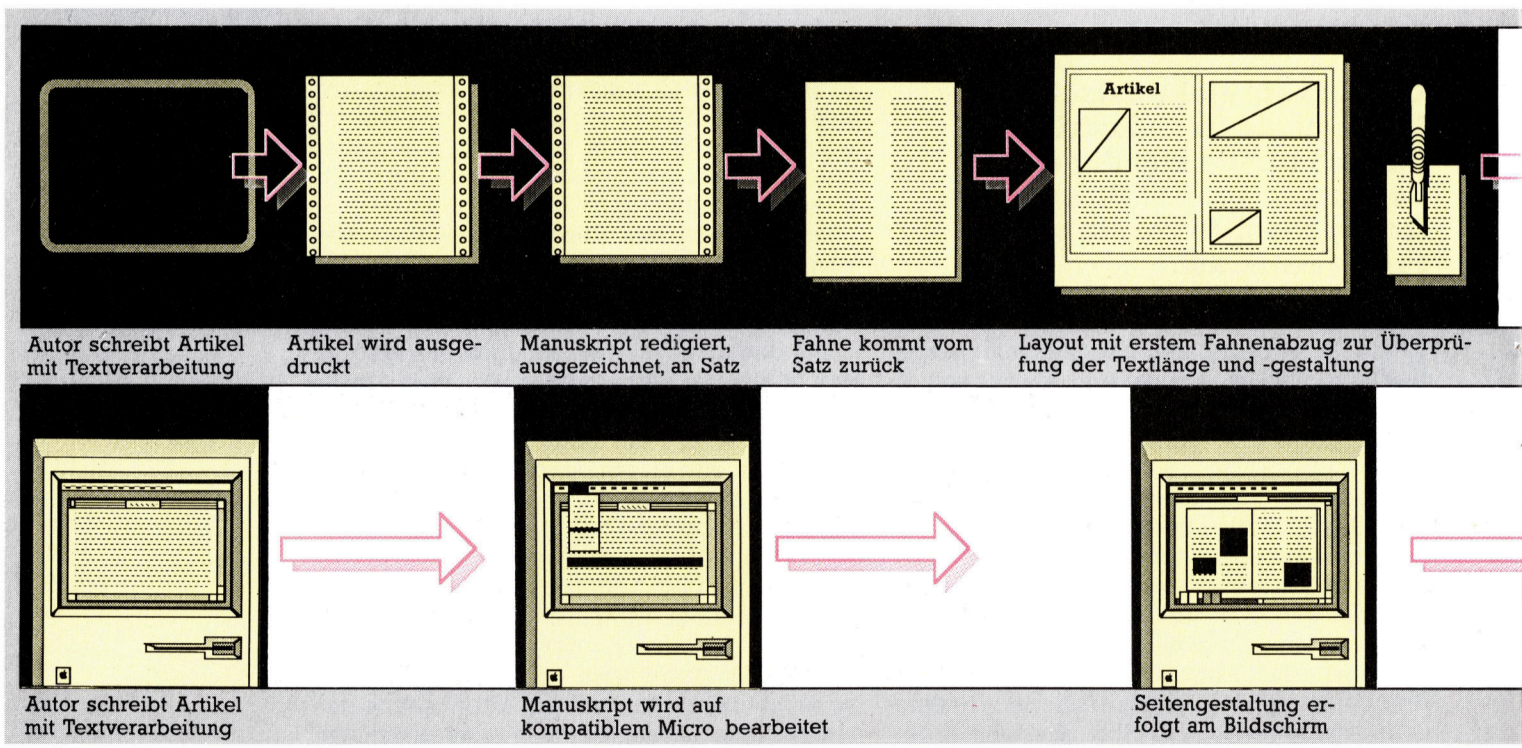
Dabei zeigt sich oft, daß Artikel zu lang oder zu kurz für den vorgesehenen Raum sind. In diesem Fall müssen Redakteur und Produktionsredakteur Kürzungen vornehmen oder Text ergänzen. Kürzungen oder Ergänzungen werden auf der ersten Fahne angebracht, so daß die zweite Fahne bereits die richtige Länge hat. Schließlich wird die Seite mit Diagrammen, Fotos und dem richtigen Text montiert und zum Drucker geschickt.

Es geht auch schneller

Aus dieser Kurzbeschreibung der derzeitigen Magazinproduktion ist zu ersehen, welchen Produktionsprozeß der Beitrag eines Autoren bis zum Fertigprodukt durchläuft. Und wieviel Zeit dabei vergeudet wird. Der Computer des Autoren steht völlig isoliert und ist kaum in den Produktionsprozeß integriert.

Doch es geht auch einfacher, mit einem einer normalen Textverarbeitung nicht unähnlichen Programm. Es enthält zusätzliche Befehle, mit denen etwa unterschiedliche Schriftarten gewählt werden können. Nehmen wir einmal an, die Redaktion benutzt ein Textverarbeitungsprogramm, das mit dem Computer des Autoren kompatibel ist. Der Autor kann eine Diskette abgeben, auf der sein Artikel abgespeichert ist. Ja, er könnte den Beitrag mit einem Modem sogar über Telefon senden. Das Redigieren würde am Bildschirm erfolgen. Selbst wenn daran anschließend der gewohnte Produktionsprozeß einsetzt (Ausdruck des Ma-

Druck auf neuen Wegen



nuskripts und Übergabe an die Setzerei), würde doch der Arbeitsablauf beschleunigt.

Ist die Setzmaschine in der Lage, die Diskette, auf die das Original-Manuskript (redigiert) gebracht wurde, zu lesen, erfolgt seine Weiterverarbeitung lediglich durch einige „magnetische Schritte" – bis zur Herstellung des Satzes.

Man bezeichnet das als „Direkteingabe-System": Der Text wird nur einmal vom Autor eingegeben und dann nacheinander in andere Maschinen gebracht, indem man ihn von Diskette lädt.

Die Möglichkeiten der neuen Technologie indes reichen viel weiter. Neuere Computerentwicklungen, etwa WIMP (Windows (Fenster), Icons (Piktogramme), Mouse (Maus)-Programme) erleichtern dem Layout die Arbeit. Der Layouter kann seine Seite auf dem Bildschirm gestalten, bereits digitalisierte Bilder und Fotos einfügen. Unterschiedliche Schriftgrößen und Verzerrungen werden mit Hilfe der Maus erzeugt. Eine Kopie der Seite wird dann direkt von einem Hochleistungsdrucker ausgegeben.

Tag für Tag

Der Produktionsablauf von Zeitungen unterscheidet sich nur teilweise von dem bei Magazinen. Die schnellen Verarbeitungsanforderungen bei Tageszeitungen in Relation zu ihrer hohen Auflage bedeuten letztlich, daß die neue Technologie diesen Publikationszweig noch viel stärker beeinflussen wird. Die Veränderungen zeichnen sich bereits ab.

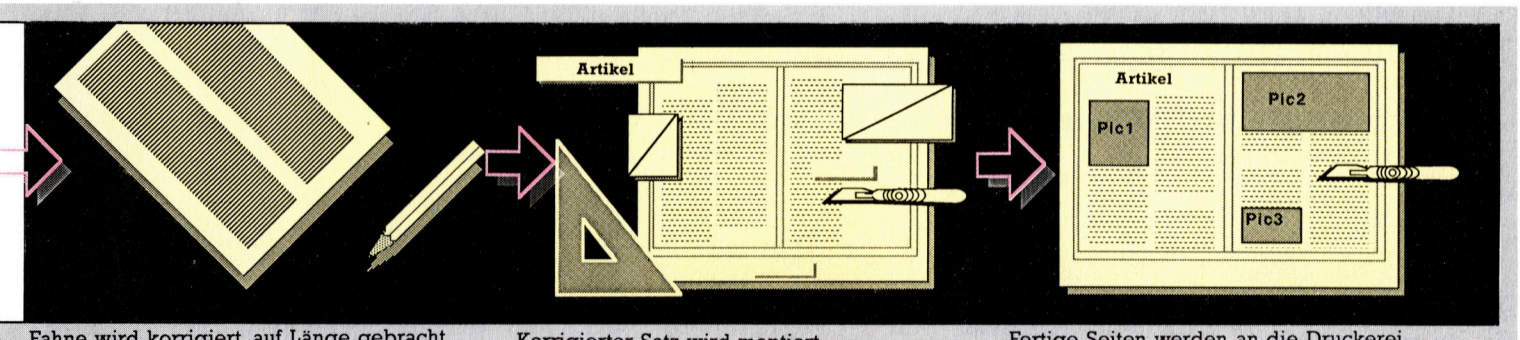
Einige Lokalzeitungen haben als Reaktion auf die zahlreichen kostenlosen Anzeigenblätter, die billig hergestellt werden, ihre Produktionsprozesse computerisiert. Sie kommen mit einem Fünftel der Belegschaft aus, die derzeit erforderlich ist.

Schlagzeile via Modem

Ein wichtiger Aspekt der elektronischen Verlagsverwaltung betrifft den Vertrieb. Überregionale Tageszeitungen und Magazine werden an einem, vielleicht zwei Druckorten hergestellt und dann über Straße und Schiene transportiert und ausgeliefert. Mit elektronisch erzeugten Zeitungen ließe sich dieses Verfahren radikal ändern. Die Herstellung des Papiers mag zwar an einer Stelle erfolgen, der Druck indes kann mittels Datenfernübertragung über gewöhnliche Telefonleitungen an mehreren Orten des Landes erfolgen. Damit werden die Vertriebskosten drastisch reduziert. Zudem ist es leichter, Lokalausgaben herzustellen: Lokalteile werden dem Mantel einfach beigelegt.

Einerseits steckt in der neuen Technik die Gefahr des Arbeitsplatzabbaus. Andererseits können aber auch kleine Zeitungen sich halten, die sonst der Konkurrenz zum Opfer fielen.

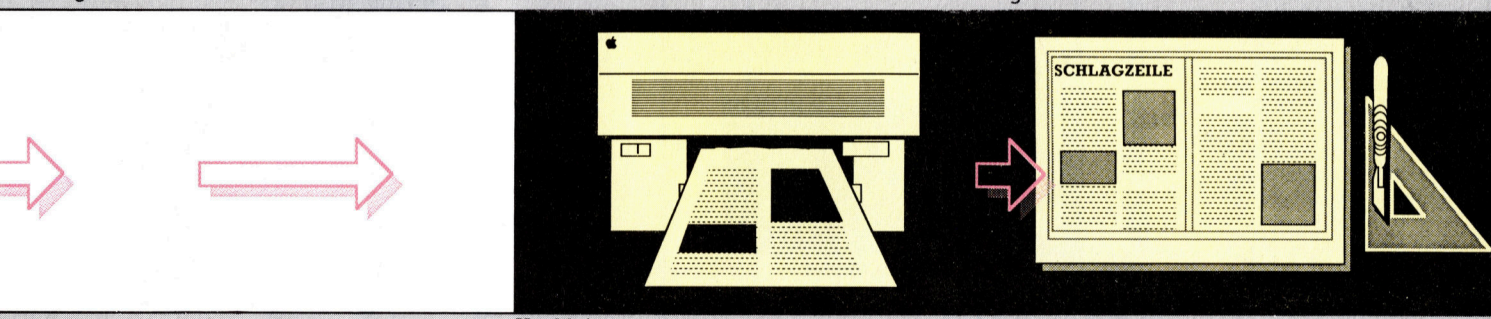
Im zweiten Teil unseres Überblicks wollen wir uns damit befassen, welche elektronischen Herstellungsmethoden auf dem Microcomputermarkt angeboten werden und wie ein Magazin vom Autoren bis zum Fertigprodukt auf einem einzigen Rechner hergestellt werden kann.



Fahne wird korrigiert, auf Länge gebracht und geht an Satz zurück

Korrigierter Satz wird montiert

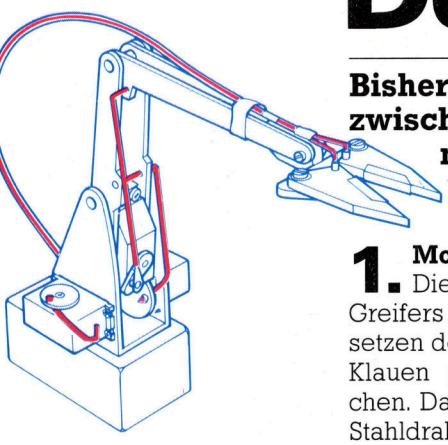
Fertige Seiten werden an die Druckerei gesandt



Hochleistungsdrucker wirft fertige Seite aus

Fertige Seiten werden an die Druckerei gesandt

Der Arm wird mobil



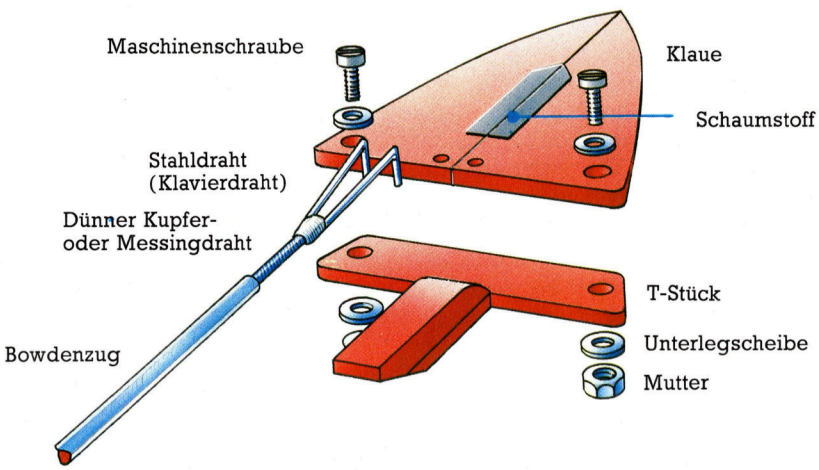
Bisher war unser Robot-Arm noch unbeweglich – Schubstangen zwischen Motoren und den beweglichen Gliedern sollen ihm jetzt die nötige Kraft für zukünftige Aufgaben verleihen.

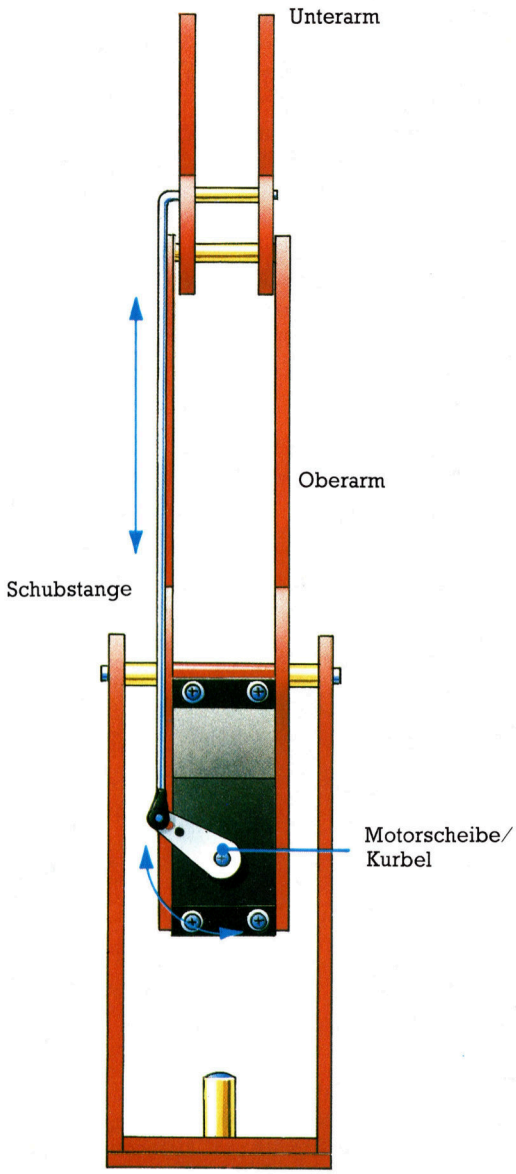
1. Montage des Greifers

Die am T-Stück und an den Klauen des Greifers markierten Stellen werden zum Einsetzen der Gewindeschrauben durchbohrt. Die Klauen polstern Sie mit Schaumstoff-Stückchen. Dann noch zwei kleinere Löcher für den Stahldraht (Klavierdraht) des Bedienungsmechanismus bohren. Klauen und T-Stück mit Gewindeschrauben und -Muttern und Unterlegscheiben verbinden. An einem Ende des 500 mm langen Bowdenzuges befestigen Sie zwei 30-mm-Stücke Stahldraht – dazu umwikkeln Sie die Stahldrähte und den Bowdenzug-Innendraht mit feiner Kupferlitze und verzinnen die Wickelung. Die freien Drahtenden im rechten Winkel biegen und in die Klauen-Löcher einführen. Bei geschlossenen Klauen nun den Außenmantel des Bowdenzuges am T-Stück festbinden und mit Klebstoff fixieren.

2. Schubstange für Unterarm

Der Unterarm wird von dem bereits im anderen Armteil montierten Motor angetrieben. Mit einer Kurbel oder Scheibe wird die

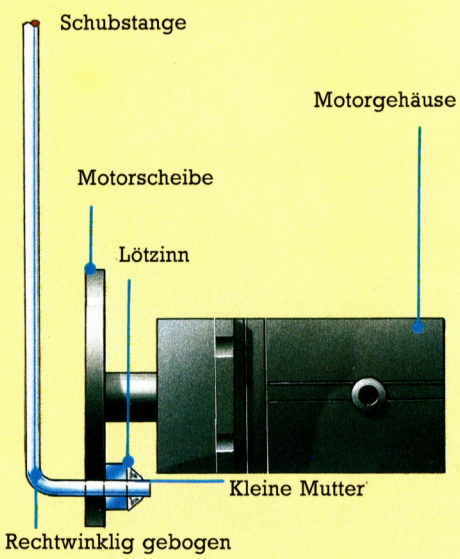




Schubstangen und Motorscheiben verbinden

Schubstangen kann man in Modellbau-Geschäften kaufen oder aber auch einfach aus normalem 2-mm-Stahldraht herstellen. Der Stahl wird an den Enden rechtwinklig abgebogen und durch die Löcher der Motorscheibe bzw. des entsprechenden Armteils gesteckt. Durch Anlöten einer Mutter wird ein Rausrutschen verhindert.

Die Schubstangen-Löcher in den Holzteilen können Sie durch Einkleben eines Messingröhrchens vor Abnutzung schützen. Das abgebogene Ende des Drahtes reibt dann nicht direkt auf dem Holz, sondern läuft innerhalb des Röhrchens.



computer kurs

Heft 49–60

Computer Welt

Hardware

Tips für die Praxis

Software

Basic

Peripherie

Bits und Bytes

Prolog

Lisp

Forth

Index Band 5

Alle zwölf Hefte erscheint ein solcher Teilindex. Der Gesamtindex erscheint mit dem letzten Heft — darin einbezogen sind Kreuzverweise auf die Artikel, die mit dem gesuchten Stichwort in Verbindung stehen.

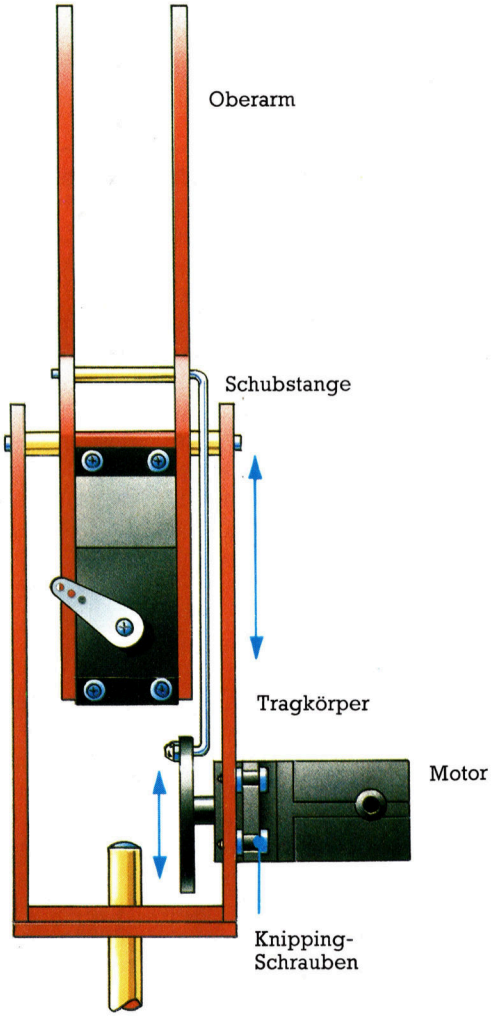
Stichwörter, die auf die vorletzte Heftseite hinweisen, sind als VS und mit der betreffenden Heftnummer gekennzeichnet.

Drehbewegung des Motors dazu in eine Hin- und Herbewegung umgesetzt. Die Scheibe bzw. Kurbel bekommt ein Loch für das Ende der Schubstange. Setzen Sie die Scheibe oder Kurbel in gleicher Stellung wie in der Zeichnung auf den Motor.

Als Schubstange nehmen Sie einen 2 mm starken Rundstahl, dessen Enden rechtwinklig abgebogen werden. Die beiden umgebogenen Enden passen in die Scheibe oder Kurbel am Motor und am Unterarm. Die Schubstange hat die richtige Länge, wenn Motorscheibe bzw. Kurbel auf ihrem unteren Totpunkt und die beiden Armteile dann im rechten Winkel stehen. Motorachse von Hand im Uhrzeigersinn drehen. Ist der Mechanismus leichtgängig und bringt den Unterarm langsam in ausgestreckte Stellung, die Schubstange fixieren.

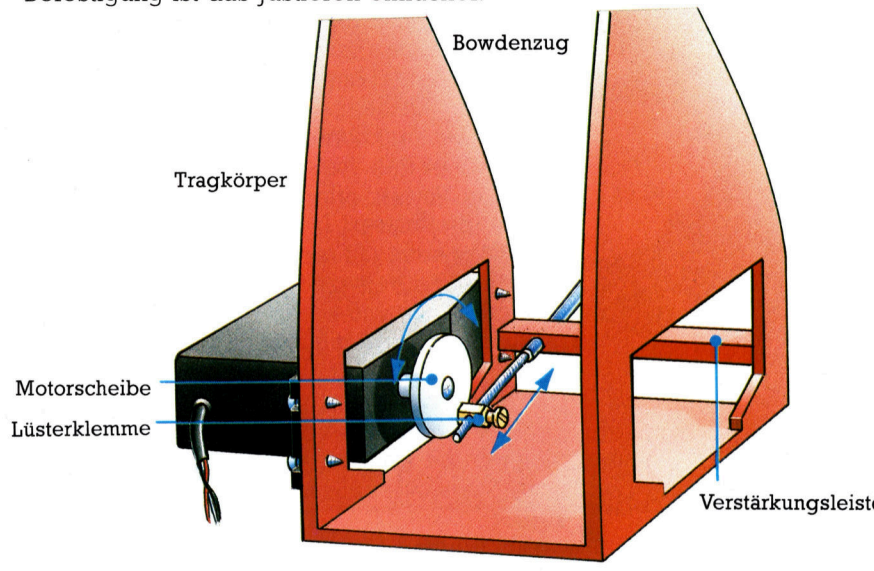
3. Schubstange für Oberarm

Die Bilder zeigen, wie die beiden anderen Motoren am Tragkörper montiert werden. Der Motor rechts steuert den Oberarm. Scheibe des Motors auf unteren Totpunkt stellen und zwischen Scheibe und Oberarm die Schubstange einsetzen. Die Länge der Stange ist korrekt, wenn der Oberarm waagerecht liegt und die Scheibe des Motors sich im unteren Totpunkt befindet.



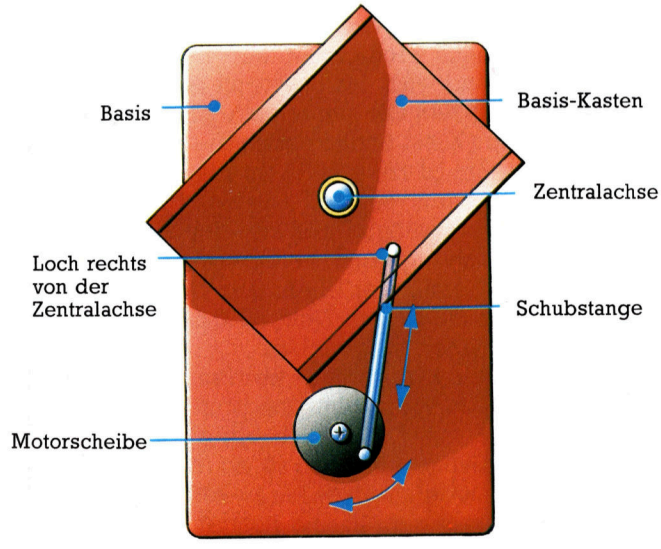
4. Bowdenzug montieren

Der am Tragkörper links montierte Motor öffnet und schließt über den Bowdenzug die Greifklauen. Der Bowdenzug wird durch die Verstärkungsleiste des Tragkörpers gesteckt und fixiert. Den Stahldraht auf der Scheibe des Motors mit einer Lüsterklemme wie gezeigt befestigen. Sie müssen wahrscheinlich etwas experimentieren, bis sich die Klauen beim Drehen des Motors sauber öffnen und schließen. Sie können den Draht auch rechtwinklig abbiegen und durch ein Loch in der Scheibe hindurchstecken und auf der Rückseite befestigen. Bei der Lüsterklemmen-Befestigung ist das Justieren einfacher.



5. Schubstange für den Tragkörper

Die letzte Schubstange stellt die Verbindung zwischen dem in der Basis eingebauten Motor und dem Tragkörper her. Ziehen Sie den Tragkörper vom zentralen Stahlstift ab und bohren ein kleines Loch in seine Unterseite rechts an der Kante des Tragkörper-Bodens. Tragkörper und Basis wieder zusammensetzen und Motorscheibe und Tragkörper wie gezeigt ausrichten. Schubstange montieren.



Auf hoher See

Jetzt, da die Segel gesetzt sind, muß wöchentlich über Mannschaft und Schiff Buch geführt werden. Diese Aufgabe wird durch das Logbuch-Modul erfüllt. Das Führen eines Logbuches ist sehr wichtig.

Während der Fahrt in die Neue Welt ist das Tempo des Schiffes von der Mannschaftsanzahl und -stärke abhängig. Eine große, gesunde Mannschaft erhöht zwar nicht das Fahrtempo; sinkt jedoch die Gesamtstärke unter einen bestimmten Punkt, so verlängert sich die Gesamtdauer der Reisezeit aufgrund geringerer Effizienz.

Eine der Funktionen des Logbuchs ist, den Spieler über den Tod eines Mannes, zur Neige gehende Vorräte oder verbleibende Restfahrzeit zu informieren. In Zeile 41 wird die Variable EW gesetzt. Sie repräsentiert die zusätzlich erforderliche Reisezeit in Wochen, sofern die Gesamtstärke unter eine bestimmte Grenze gesunken ist, und wird zu den ursprünglichen acht Wochen addiert. Damit die Reisezeit in Ganzzahlen dargestellt werden kann, ohne den Rest von EW zu verlieren, wird die Zeit berechnet und als Realzahl in EW abgelegt und unter Verwendung von INT als Ganzzahl zu JL addiert.

Der Programmcode für den Wochenbericht befindet sich in der Unterroutine ab Zeile 5300. Erste Aufgabe der Wochenbericht-Unterroutine ist es, mittels der Schleife bei Zeile 5325 zu überprüfen, ob jemand gestorben ist. Dazu wird das erste Element des Arrays T$(,) für jedes der 16 Crewmitglieder daraufhin untersucht, ob der Stärkewert auf −999 (=Tod) gesetzt ist.

Die „Verstorbenen" werden zu X addiert. Beträgt der Wert nicht −999, erfolgt der nächste Schleifendurchlauf. Ansonsten wird in Zeile 5340 die Stärke und das Typ-Element auf 0 gesetzt, damit der Verstorbene nicht in der nächsten Woche noch einmal aufgeführt wird. Anschließend wird der Seemann von der Mannschaftsanzahl CN subtrahiert. Ist niemand gestorben (X bei Schleifende=0), verzweigt das Programm zu Zeile 5400, wo die Vorräte überprüft werden. Sind alle Seeleute gestorben, wird ausgegeben: „All the remaining crew died. Your ship drifts helplessly on the empty ocean. Game over." Lebt ein Teil der Mannschaft noch, wird in Zeile 5390 die Anzahl der Verstorbenen ausgegeben.

Verbrauchte Vorräte werden ähnlich verbucht. Das Programm überprüft mittels einer Schleife von 1 bis 4, ob ein Vorratstyp verbraucht ist. Ist das der Fall, so ist der entsprechende Wert im Array PA() −999. In Zeile 5415 wird darauf geprüft und nötigenfalls die Meldung „You have run out of . . ." ausgegeben.

Anschließend wird das Element in PA() auf 0 gesetzt. Sind mehrere Vorräte ausgegangen, wird ein „AND" zwischen den Begriffen ausgegeben. Während des Spiels ist es möglich, ausgegangene Vorräte wieder aufzufrischen.

Die zusätzliche Reisezeit wird berechnet, indem aus dem Stärke-Array die Gesamtstärke ermittelt und mit der Minimalstärke verglichen wird. Um die Fahrt mit maximaler Effizienz fortzusetzen, muß die Gesamtstärke mindestens 800 betragen, also die Crew etwa aus acht Mann voller Stärke oder 16 Mann halber Stärke bestehen. Zeile 5465 setzt eine Schleife für jedes Element des Mannschafts-Arrays, und Zeile 5470 ermittelt die Stärke in X.

In Zeile 5480 wird überprüft, ob die Gesamtstärke größer 799 ist. Wenn ja, so wird die Reisezeit nicht erhöht, und das Programm verzweigt zu Zeile 5494. Eine neue Woche beginnt. Die Reise geht weiter und weitere Abenteuer stehen der Mannschaft bevor. Wir werden sehen, welche.

Essen gleichmäßig teilen

Ist die Stärke kleiner als 800, berechnet die Formel in Zeile 5489 die zusätzliche Reisezeit. Dazu wird die Gesamtstärke von 800 subtrahiert, durch 800 dividiert und das Ergebnis als Realzahl in EW gespeichert. In Zeile 5490 wird der Integer von EW zur Reisezeit addiert, so daß die verbleibende Dauer nicht in Teilen einer Woche dargestellt wird. Um zu gewährleisten, daß die Stärke eines „toten" Seemannes nicht durch WF in der Unterroutine ab Zeile 9300 (diese wird im nächsten Artikel besprochen) reduziert wird, fügen Sie bitte folgende Zeile ein:

```
9315 IF TS(S1,2)=0 THEN 9340
```

Während das Schiff in die Neue Welt segelt, beeinflussen verschiedene Ereignisse die Stärke der Mannschaft. Bisher ist der einzige Faktor die Nahrung. Wie in der Realität gilt auch in unserem Spiel „Essen an Bord ist dreimal so wichtig wie an Land". Also muß der Ernährung der Seeleute große Aufmerksamkeit geschenkt werden. Abwechslung und reichlich ist die Devise des Smutje. Da das Essen jedoch gleichmäßig aufgeteilt wird, reduziert sich die Stärke aller Seeleute bei unzureichen-

Modul fünf: Kapitänslogbuch

Wochenbericht

```
5300 REM END OF WEEK REPORT
5305 PRINTCHR$(147)
5310 S$="           CAPTAINS LOG*":GOSUB9100
5312 S$="           ------------*":GOSUB9100
5314 GOSUB9200
5316 PRINT:PRINT"          END OF WEEK";WK
5318 GOSUB9200
5320 X=0
5325 FORT=1TO16
5330 IFTS(T,2)<>-999THEN5350
5335 X=X+1
5340 TS(T,1)=0:TS(T,2)=0
5345 CN=CN-1
5350 NEXT
5355 IFX=0THEN5400
5358 PRINT:PRINT
5360 S$="DURING THE PAST WEEK*":GOSUB9100
5365 IFCN>0THEN5390
5367 S$="ALL THE REMAINING CREW DIED*":GOSUB9100
5369 GOSUB9200:PRINT:GOSUB9200
5373 S$="THE SHIP DRIFTS HELPLESSLY*":GOSUB9100
5374 GOSUB9200
5376 S$="ON THE EMPTY OCEAN.....*":GOSUB9100
5378 GOSUB9200
5380 PRINT:PRINT
5382 S$="      GAME OVER*":GOSUB9100
5384 PRINT:PRINT
5386 GOSUB9200:END
5388 GOTO5386
5390 PRINTX;
5392 IF X=1 THEN S$="CREW MEMBER HAS DIED*"
5394 IFX<>1 THEN S$="CREW MEMBERS HAVE DIED*"
5396 GOSUB9100
5398 GOSUB9200
5400 PRINT:PRINT
5405 S$="YOU HAVE NOW RUN OUT OF "
5410 FOR T=1TO4
5415 IFPA(T)<>-999THEN5440
5420 PRINTS$;P$(T)
5425 PA(T)=0
5428 S$="                    AND "
5440 NEXT
5450 GOSUB9200
5455 REM CALC NEW JL
5460 X=0
5465 FORT=1TO16
5470 X=X+TS(T,2)
5475 NEXT
5480 IFX>799THEN5494
5481 PRINT:PRINT
5482 S$="THE CREW IS BELOW FULL STRENGTH*":GOSUB91
00
5483 S$="SO JOURNEY MIGHT TAKE LONGER*":GOSUB9100
5489 EW=EW+((800-X)/800)
5490 JL=JL+INT(EW)
5492 PRINT
5494 S$=K$:GOSUB9100
5496 GETI$:IFI$=""THEN5496
5499 RETURN
```

Wochen-Variable

```
41 EW=0: REM EXTRA WEEKS
```

Änderung an Hauptschleife

```
880 GOSUB 5300: REM END-OF-WEEK REPORT
```

BASIC-Dialekte

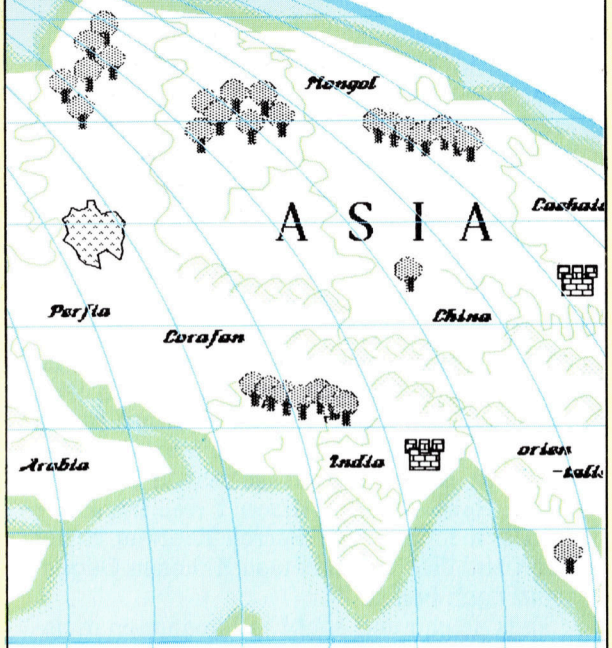
Spectrum:
Ändern Sie das Programm wie folgt:

```
5305 CLS
5496 LET I$=INKEY$: IF I$="" THEN GOTO 5496
```

Acorn B:
Führen Sie folgende Änderungen aus:

```
5305 CLS
5496 I$=GET$
```

Nachdem Modul 5 des Programms eingegeben ist, können wir die bisher programmierten Abschnitte unseres Spiels mittels eines Flußdiagramms noch einmal überprüfen.



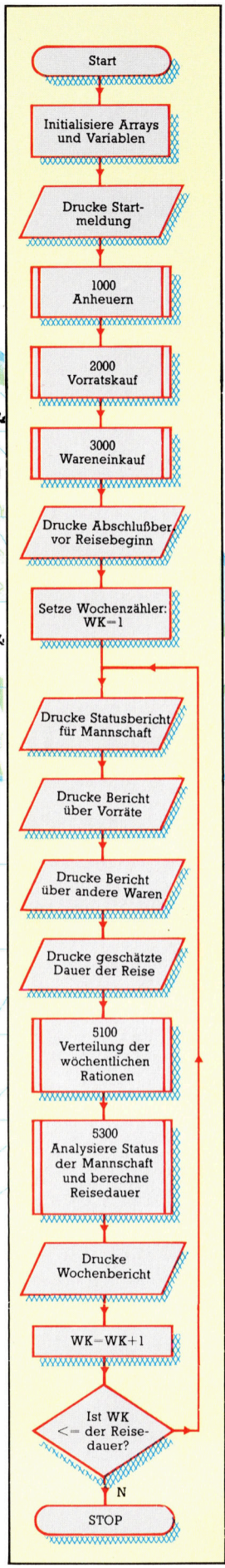
der Nahrung um denselben Wert. Um jetzt schon die Stärkeverhältnisse der Seemänner zu variieren, kann bei Beginn der Reise eine Routine eingefügt werden, die die Stärkewerte zufällig bestimmt. Denken Sie jedoch daran, diese Routine vor Eingabe des nächsten Programmteils im nächsten Artikel wieder zu entfernen.

```
601 REM TEST CODE
602 FOR T=1 TO 16
603 IF TS(T,2)=100 THEN TS(T,2)=
    INT(RND(1)*100)+1
604 NEXT
```

Die Routine in Zeile 603 untersucht jeweils das zweite Element des Mannschafts-Arrays. Dabei wird der Stärkewert von 100 mit der RND-Formel auf einen Zufallswert zwischen 1 und 99 geändert (1 wird addiert, damit der Wert nicht 0 betragen kann).

Die Wochenbericht-Routine wird mit GOSUB aus der Hauptschleife zwischen den Zeilen 820 bis 889 aufgerufen. Ursprünglich wurde diese Routine durch FOR...NEXT kontrolliert, mit der Reisedauer JL als Obergrenze. Da das neue Modul diesen Wert jedoch verändern kann, muß die FOR-NEXT-Schleife durch folgende Zeilen ersetzt werden:

```
820 WK=1:REM MAIN VOYAGE LOOP
889 WK=WK+1:IF WK<=JL THEN 825
```



Kostümprobe

Um die Spielfiguren in unserem Programm zu beleben, brauchen wir ein Software-Modul zur Verwaltung ihrer verschiedenen Eigenschaften. Sie sind in fast allen Situationen bedeutsam für den kontrollierten Fortgang des Spiels.

BASIC-Adventureprogramme enthalten im allgemeinen zahlreiche IF-Bedingungen. Ein bekanntes Buch über die Adventure-Programmierung enthält beispielsweise ein Listing mit nicht weniger als vier vollen Seiten IF-THEN-Anweisungen. Mut für die Programmerstellung in BASIC macht ein solcher Anblick nicht gerade . . .

Das Problem bei Adventure-Spielen ist die große Zahl von Variablen. Im Spiel „Dog and Bucket" etwa besitzt jede Spielfigur zehn Eigenschaften, die vom entsprechenden Programmteil ständig berücksichtigt und verändert werden müssen. Das geschieht in festen Abläufen, die etwa so aussehen: „IF Person X in der Halle ist AND Person Y den Drink von Person X nimmt AND Person X etwas in der Hand hat, THEN wirft Person X diesen Gegenstand nach Person Y".

Falls solche und ähnliche Situationen in unserem Spiel vorkommen, müssen wir wohl oder übel mit einer Vielzahl von IFs im Programm rechnen. Schön ist das nicht, schon das Eintippen des Programms wird langwierig und mühsam. Wie läßt sich die Zahl der IF-Anweisungen reduzieren oder – noch besser – wie können sie ganz vermieden werden? Zum Glück gibt es mehrere Alternativen zum Eingeben endloser IF-THEN-Anweisungen. Da diese kürzer sind, kann der Rechner sie auch schneller ausführen – wir sparen also nicht nur beim Eintippen Zeit, sondern beschleunigen auch den Programmablauf.

Zur Demonstration der unterschiedlichen Möglichkeiten wollen wir ein kurzes Programm erstellen. Bedingungen sollen damit definiert und abgefragt werden können, danach soll das Programm die entsprechenden Maßnahmen ergreifen. Aus den verschiedenen Lösungsansätzen können wir uns den günstigsten Weg heraussuchen. Geben Sie zuerst das Programm-Modul (Zeile 4000 bis 4140) aus dem letzten Heft ein. Als nächstes werden diese Zeilen eingetippt:

```
10 h$="human ":a$="animal ":m$="male ":f
$="female ":q$="Are you ": DIM c(4)
20 GOSUB 4050: REM clear the screen
30 REM set up the four variables
40 PRINT q$;h$;: INPUT i$: c(1)=ABS(i$="
y" OR i$="Y")
50 PRINT q$;a$;: INPUT i$: c(2)=ABS(i$="
y" OR i$="Y")
60 PRINT q$;f$;: INPUT i$: c(3)=ABS(i$="
y" OR i$="Y")
70 PRINT q$;m$;: INPUT i$: c(4)=ABS(i$="
y" OR i$="Y")
80 PRINT
```

Mit Zeile 10 dieses Moduls werden die in den Zeilen 40 bis 70 verwendeten Stringvariablen eingerichtet und das Array c mit vier Elementen DIMensioniert. Dieses Array speichert vier Werte, die von der Antwort auf bestimmte Fragen abhängen. Bei einigen Heimcomputern werden Wahrheitswerte mit 0 für falsch und –1 für wahr ausgegeben, der Spectrum arbeitet mit +1. Um das Minuszeichen zu beseitigen, haben wir den ABS-Befehl vorgesehen.

Mit RUN erzeugt das Modul ein Array c(4), das später abgefragt werden kann. Hier liegt die Schwierigkeit: Vier einzelne Werte ergeben die Möglichkeit von 16 verschiedenen Kombinationen. Möchte man etwa wissen, ob auf die erste Frage ein „Y" oder ein „N" eingegeben wurde, kann man natürlich mit einem IF-THEN-Befehl arbeiten. Bei 15 oder 16 Möglichkeiten müßten jedoch 16 Zeilen geschrieben werden. Unser Beispiel zeigt Ihnen, wie so etwas aussehen würde:

```
90 REM Print messages
100 IF c(1)=1 AND c(2)=1 AND c(3)=1 AND
c(4)=1 THEN PRINT "You're a mixed up wer
ewolf!"
110 IF c(1)=1 AND c(2)=1 AND c(3)=1 AND
c(4)=0 THEN PRINT "You're a female werew
olf!"
```

. . . und so weiter. Das ist sehr langwierig – zum Glück gibt es jedoch Alternativen. Die einfachste Möglichkeit ist ein ON-GOTO-Befehl. Aber auch dabei stößt man bald an Grenzen, weil dieser Befehl relativ unflexibel ist und es bessere Möglichkeiten gibt.

Für unser Spielfiguren-Modul wollen wir einen anderen Weg beschreiten. Bei genauer Betrachtung stellt sich heraus, daß der „Entscheidungsbaum" die beste Lösung ist.

Lassen Sie uns beim ersten Beispiel mit vier Bedingungen bleiben und den gleichen Ablauf nicht mit IF-Befehlen, sondern in einer Baumstruktur verwirklichen. Das Bild zeigt eine grafische Darstellung dieser Prozedur.

Am Verzweigungspunkt 1 wird die erste Bedingung „Human" (Mensch ?) geprüft. Eine positive Antwort (Feldvariable c(1)=1) führt zum Verzweigungspunkt 3; falls negativ geantwor-

tet wird, erfolgt die Verzweigung zu Punkt 2. Dieser Vorgang wird bis zu den „Wurzeln" des Baums fortgesetzt – den untersten, mit den Ziffern 16 bis 31 bezeichneten Punkten. Jeder Verzweigungspunkt kann bei Bedarf eine Nachricht erzeugen oder ein Unterprogramm aufrufen. Wenn alle vier Fragen mit „nein" bzw. mit „ja" beantwortet werden, landet der Anwender bei Punkt 16 bzw. 31 – die entsprechenden Meldungen für diesen Fall haben wir abgedruckt.

Bevor wir den „Entscheidungsbaum" in unser Programm integrieren können, sollten wir ihn genau analysieren: Er ist vierstufig aufgebaut. Die Bedingungen, auf die in den verschiedenen Stufen geprüft wird, sind die ähnlich: Stufe 1 fragt nach „Mensch?", Stufe 2 nach „Tier?", Stufe 3 nach „weiblich?", Stufe 4 nach „männlich?". Die Nummern der einzelnen Stufen entsprechen den Elementen des Feldes c(4), bei dem c(1) die Bedingung „Mensch" enthält usw.

Nun muß die Beziehung zwischen den Nummern der einzelnen Verzweigungspunkte erklärt werden. Jede Entscheidung verzweigt zu zwei untergeordneten weiteren Knotenpunkten, deren Nummer so berechnet wird:

Nummer des unteren Verzweigungspunktes = (gewählter Verzweigungspunkt*2)+Wahrheitswert

Der Wahrheitswert aus Feld c ist 0 oder 1.

Wenn Sie diesen Programmteil eingeben und starten, sollte Sie das Programm mit einer Ihrer vorherigen Eingabe entsprechenden Meldung begrüßen.

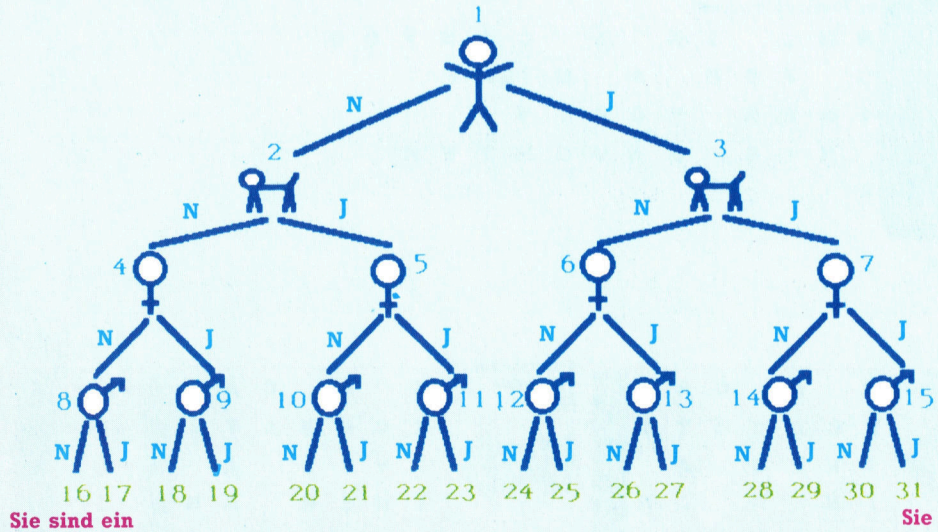
```
90 REM sort through the tree
100 n=1: REM start at node 1
110 FOR k=1 TO 4: n=(2*n)+c(k): NEXT k
120 RESTORE: REM set data pointer to sta
rt of messages
130 FOR x=1 TO n-15: READ z$: NEXT x: PR
INT "You're a ";z$: PRINT
140 PRINT "Hit a key to continue..."
150 GOSUB 4130: REM get a character
160 GOTO 20
170 REM data for messages
180 DATA "mystery","male mineral?!","fem
ale mineral?!","ridiculous time-waster",
"neuter animal","male animal","female an
imal","snail?  They can be bisexual, you
 know","human, sex unknown","man","woman
","mixed-up kid","neutered werewolf","ma
le werewolf"
190 DATA "female werewolf","mixed-up wer
ewolf"
```

Wie arbeitet dieses Programm-Modul? Wir starten bei Verzweigungspunkt 1 und wollen im Auge behalten, an welcher Stelle des „Entscheidungsbaums" wir uns befinden. Dazu wird die Variable n mit der Nummer des Verzweigungspunktes initialisiert. Zeile 110 bewirkt die Fortbewegung durch den Baum, von der Spitze bis zu den Wurzeln. Mit der Variablen k wird nach der Formel – Nummer des nächsten Verzweigungspunktes = (2*Verzweigungspunkt-Nummer)+c (Stufennummer) – der nächste Punkt angesteuert.

Nach dem Durchlaufen des Baums bis zu einem der Punkte 16 bis 31 wird 15 von der Nummer des Endpunktes abgezogen. Mit dem Ergebnis erhalten wir durch Zeile 130 die dem erreichten Punkt entsprechende Meldung aus den DATA-Zeilen. Die Zeilen 140–160 warten auf die Wiederholung des Ablaufs.

Natürlich könnte die vorliegende Aufgabe auch anders gemeistert werden – trotzdem zeigt bereits dieses einfache Beispiel, wie hilfreich die Baumstruktur auch in komplizierteren Bedingungs-Strukturen sein kann.

Baum der Erkenntnis



Speziell bei einer Vielzahl möglicher Entscheidungen kann ein Programm gesteuert werden. Der abgebildete Baum ist relativ einfach strukturiert – an jedem Knotenpunkt sind nur zwei Entscheidungen möglich. Auch in BASIC lassen sich kompliziertere Verzweigungen aufbauen – Programmiersprachen wie LISP, LOGO oder PASCAL eignen sich allerdings besser für diesen Zweck. Im gezeigten Baum werden vier Fragen gestellt, für die insgesamt 16 Antwort-Kombinationen möglich sind. Allerdings ist nicht jede Kombination gleichermaßen sinnvoll und wahrscheinlich ...

Weich gerollt

Viele Arcade-Spiele simulieren schnelle Bewegung durch einen „vorbeiziehenden" Hintergrund. Auch der Commodore 64 unterstützt ein „weiches Scrollen" in die vertikale und horizontale Richtung.

Das Scrollen von Bildschirmen läuft in drei Stufen ab. Zunächst wird jedes Byte des Bildschirmspeichers um eine Position abwärtsbewegt. Da der Bildschirmspeicher die Zeilen als sequentielle Daten speichert, scheint sich dadurch jedes Zeichen um eine Position nach links zu bewegen – mit Ausnahme der Spalte am linken Bildschirmrand.

Bei diesem Vorgang verschwindet das Zeichen links oben im Bildschirm, während ein anderes Zeichen an der linken unteren Bildschirmkante auftaucht.

In der zweiten Stufe wird die entsprechende Spalte des Bildschirmspeichers in die rechte Spalte des Schirms kopiert.

Wenn nun die Scroll-Register des VIC verändert werden, scheint der Bildschirminhalt pixelweise in den Schirm „hineinzuscrollen".

Der Video Controller Chip (VIC) des Commodore 64 kann den Bildschirm des Commodore in jede Richtung um bis zu acht Pixel verschieben. Die horizontale Bewegung wird von den drei niederwertigen Bits des VIC-Registers bei 53270 ($DO16) gesteuert. Wenn diese Bits nacheinander auf Werte von 7 bis 0 gesetzt werden, verschiebt sich der Bildschirminhalt jeweils um ein Pixel nach links.

POKE 53270, (PEEK(53270) AND 248) + P

erledigt diesen Vorgang von BASIC aus (P ist ein Wert von 0 bis 7).

Wenn dieser Vorgang in einer Maschinencoderoutine verwendet wird, die sämtliche Bildschirmdaten um eine Position nach links bewegt und auf der rechten Seite eine neue Spalte einfügt, entsteht ein „sanft" rollendes Bild. Damit die Daten sich nicht ruckartig bewegen, muß allerdings die Bildschirmbreite des Commodore 64 von 40 auf 38 Spalten gesetzt werden. Dafür wird Bit 3 des Registers für horizontales Scrollen mit folgendem BASIC-Befehl auf 0 gesetzt:

POKE 53270, (PEEK(53270) AND 247)

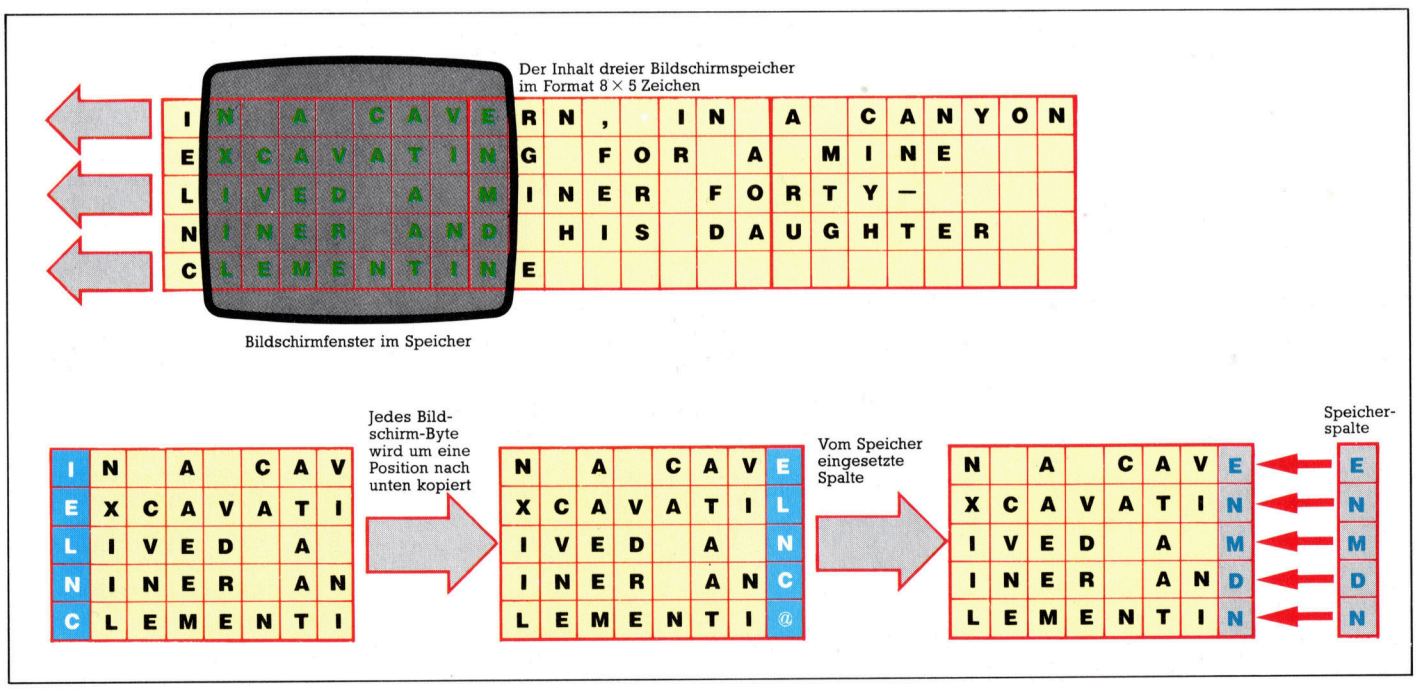
Wenn Bit 3 auf Eins steht, hat der Bildschirm 40 Spalten.

Das Diagramm zeigt, wie das weiche, horizontale Scrollen abläuft. Dabei müssen beim Bewegen oder Einsetzen von Bildschirmdaten auch jeweils die dazugehörigen Farbdaten verändert werden.

Der Vorgang ist eigentlich sehr einfach. Die Bildschirmdaten sind normalerweise in den 100 Bytes untergebracht, die bei Adresse 1024 ($0400) anfangen: Die ersten 40 Bytes stellen die oberste Zeile dar, die nächsten 40 Zeichen die zweite Zeile etc. Für die Illusion der Linksbewegung muß nur jedes Datenbyte in das darunterliegende Byte verschoben werden. Die Routine setzt dafür die Zeiger der Zero-Page und die indirekte Adressierung ein.

Wenn SB die Basisadresse des Bildschirmbereiches darstellt, dann ist das letzte Element der obersten Zeile SB+39, das der zweiten Zeile SB+79 etc. Damit die Bildschirmdaten im Speicher auf die gleiche Weise untergebracht sind, wie sie auf dem Schirm erscheinen (das heißt in Blöcken von 1000 Bytes), haben wir direkt daneben die Daten gespeichert, die wir einsetzen wollen (als das 41., 81. etc. Byte). Unsere Abbildung zeigt die Anordnung.

Der Zeiger, der auf den Anfang dieses Speicherbereichs zeigt, wird auf das erste Byte gesetzt, das auf dem Schirm erscheinen soll. Nach dem Erscheinen der ersten Spalte inkrementiert die Routine den Zeiger um eins, kopiert eine zweite Spalte auf die rechte Bildschirmseite und scrollt sie nach links. Wenn



dieser Vorgang vierzigmal wiederholt wird, wandern alle Daten einmal über den Bildschirm. Der Zeiger sollte dann um 960 (1000–40) inkrementiert werden, damit er den Anfang des nächsten Bildschirminhalts zeigt.

Dieser Ablauf muß auch für die Farbdaten ausgeführt werden. Zur Vereinfachung haben wir die Adressen der Farb-Bytes so angelegt, daß sie sich über ein konstantes Offset auf die Adressen ihrer entsprechenden Bildschirmbytes beziehen. Der Vorgang läßt sich für alle Bildschirminhalte wiederholen, die im Speicher angelegt wurden.

Um die Scroll-Routine einsetzen zu können, müssen jedoch mehrere Informationen eingegeben werden. Die Routine benötigt folgende Informationen:

1) die Anfangsadresse des Bereichs, in dem die Bildschirmdaten gespeichert sind,
2) das Offset in dem entsprechenden Farbbereich,
3) die Anzahl der Bildschirminhalte, die gescrollt werden sollen und
4) einen Verzögerungswert, der den Scroll-Vorgang verlangsamt.

Diese Daten können mit POKE in die Adressen gesetzt weden, die das Maschinencodeprogramm dafür reserviert hat.

Aufrufendes Programm in BASIC

```
10 REM ***************************
20 REM ***************************
30 REM **                       **
40 REM **  BASIC CALLING PROG   **
50 REM **  FOR SCROLL ROUTINE   **
60 REM **                       **
70 REM ***************************
80 REM ***************************
90 :
95 DN=8:REM FOR CASS DN=1
100 IFA=0THENA=1:LOAD"SCROLL.HEX",DN,1
110 POKE55,0:POKE56,32:CLR:REM LOWER MEMTOP
115 REMGOSUB1000:REM SET UP SIMPLE DISPLAY
120 :
130 LMEM  =49664:     REM START OF MEMORY
140 HMEM  =49665:     REM AREA
150 LCOFF =49666:     REM OFFSET TO COLOUR
160 HCOFF =49667:     REM MAP
170 NMSCR =49668:     REM NUMBER OF SCREENS
190 DELAY =49669:     REM DELAY VALUE
200 SCROLL=49670:     REM PROG START ADDRESS
210 :
220 REM PRINTCHR$(147):REM CLEAR SCREEN
230 INPUT"DECIMAL START ADDRESS";SA
240 HS=INT(SA/256):LS=SA-HS*256
250 POKELMEM,LS:POKEHMEM,HS
260 :
270 INPUT"NUMBER OF SCREENS";NS
290 POKE NMSCR,NS
300 :
310 INPUT"DECIMAL OFFSET TO COLOUR MAP";OS
320 HO=INT(OS/256):LO=OS-HO*256
330 POKELCOFF,LO:POKEHCOFF,HO '
340 :
350 INPUT"DELAY VALUE < 256";DV
360 IF DV>255 OR DV<0 THEN 350
370 POKEDELAY,DV
380 :
390 SYS SCROLL
400 POKE53270,PEEK(53270)OR8
```

Das aufrufende Programm lädt den Maschinencode in den Speicher und holt sich über INPUT die notwendigen Informationen. Das Programm übersetzt die eingegebenen Daten – falls nötig – in das Format Lo-Byte/Hi-Byte und legt sie in den reservierten Speicherstellen ab. Danach wird die Maschinencoderoutine aufgerufen.

Es kann jede beliebige Anfangsadresse, jedes Offset oder jede Bildschirmzahl angegeben werden. Das Ergebnis hängt jedoch davon ab, ob an den angegebenen Positionen auch wirklich Bildschirmdaten untergebracht sind. Sie können die Routine leicht testen, wenn Sie ein kurzes BASIC-Programm laden und starten, das bei der Anfangsadresse 8192 zwei einfache Bildschirminhalte anlegt. Das Offset zu den Farbdaten ist 3000 Bytes. Für das Scrollen der Daten muß das aufrufende Programm folgende Informationen erhalten:

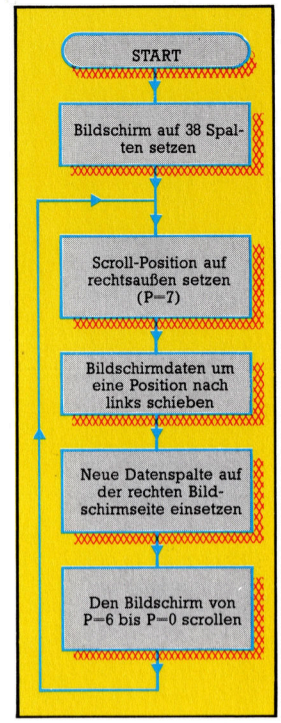
1) Anfangsadresse in Dezimal: 8192
2) Farb-Offset: 3000
3) Anzahl Bildschirme: 2
4) Verzögerung: 255

Ladeprogramm in BASIC

```
10 REM ***********************
15 REM **  BASIC LOADER     **
20 REM ** FOR HORIZONTAL    **
30 REM ** SCROLL ROUTINE    **
40 REM ***********************
50 :
60 FOR I=49670 TO 49945
70 READ A:POKEI,A
80 CC=CC+A
90 NEXT
92 READ CS:IF CS<>CC THEN PRINT"CHECKSUM ERROR":STOP
100 DATA173,22,208,41,247,141,22,208
110 DATA174,4,194,160,40,138,72,152,72
120 DATA173,22,208,41,248,24,105,7,141
130 DATA22,208,169,0,133,251,169,4,133
140 DATA252,169,0,133,253,169,216,133
150 DATA254,162,3,160,1,177,251,136
160 DATA145,251,200,177,253,136,145
170 DATA253,200,200,208,241,230,252
180 DATA230,254,177,251,198,252,136
190 DATA145,251,200,177,253,198,254
200 DATA136,145,253,230,252,230,254
210 DATA202,208,213,160,1,177,251,136
220 DATA145,251,200,177,253,136,145
230 DATA253,200,200,192,232,144,239
240 DATA173,0,194,133,253,173,1,194
250 DATA133,254,169,39,133,251,169,4
260 DATA133,252,32,241,194,173,0,194
270 DATA24,109,2,194,133,253,173,1,194
280 DATA109,3,194,133,254,169,39,133
290 DATA251,169,216,133,252,32,241,194
300 DATA162,6,173,22,208,41,248,141,22
310 DATA208,138,24,109,22,208,141,22
320 DATA208,172,5,194,136,208,253,202
330 DATA16,231,173,0,194,24,105,1,141
340 DATA0,194,173,1,194,105,0,141,1
350 DATA194,104,168,104,170,136,240,3
360 DATA76,19,194,173,0,194,24,105,192
370 DATA141,0,194,173,1,194,105,3,141
380 DATA1,194,202,240,3,76,17,194,96
390 DATA162,25,160,0,177,253,145,251
400 DATA202,240,29,165,251,24,105,40
410 DATA133,251,165,252,105,0,133,252
420 DATA165,253,24,105,40,133,253,165
430 DATA254,105,0,133,254,76,245,194
440 DATA96
450 DATA40227:REM*CHECKSUM*
```

Einstellroutine für die Anzeige

```
1000 REM **** SET UP DISPLAY ****
1010 CL=3000:REM OFFSET TO COLOUR MAP
1020 SS=8192:REM START OF DISPLAY MAP
1030 FORI=SS TO SS+479
1040 POKEI,1:REM SCREEN CODE FOR 'A'
1050 POKEI+CL,1:REM WHITE
1060 POKEI+480,2:REM SCREEN CODE FOR 'B'
1070 POKEI+CL+480,14:REM LIGHT BLUE
1080 NEXT
1085 FORI=SS+960TOSS+999
1090 POKEI,3:REM SCREEN CODE FOR 'C'
1100 POKEI+CL,3:REM CYAN
1110 NEXT
1999 :
2020 SS=9192:REM NEXT SCREEN START
2030 FORI=SS TO SS+479
2040 POKEI,3:REM SCREEN CODE FOR 'C'
2050 POKEI+CL,5:REM GREEN
2060 POKEI+480,4:REM SCREEN CODE FOR 'D'
2070 POKEI+CL+480,0:REM BLACK
2080 NEXT
2085 FORI=SS+960TOSS+999
2090 POKEI,5:REM SCREEN CODE FOR 'E'
2100 POKEI+CL,2:REM RED
2110 NEXT
```



Für eine weiche Scrollbewegung setzen wir auf dem Commodore 64 eine besondere Fähigkeit des Video Chips (VIC) ein. Mit dessen speziellen Scroll-Registern kann der Bildschirminhalt pixelweise über seine Grenzen hinausbewegt werden. Wenn diese Fähigkeit mit einer Kopierroutine im Maschinencode kombiniert wird, entsteht auf einem verkleinerten Bildschirm von 38 Spalten eine weiche Rollbewegung.

Horizontales Scrollen

```
;++++++++++++++++++++++++++
;++++++++++++++++++++++++++
;++                      ++
;++  HORIZONTAL SCROLL   ++
;++                      ++
;++     FOR CBM 64       ++
;++                      ++
;++++++++++++++++++++++++++
;++++++++++++++++++++++++++
;
SCRPTR=$FB              ;Ø PAGE
COLPTR=$FD              ;COPY POINTERS
;
MEMPTR=$FD              ;Ø PAGE PTR TO MEMORY
SCRLRG=$D016            ;HORIZ SCROLL REGISTER
SCRNLO=$00              ;SCREEN START LOBYTE
SCRNHI=$04              ;SCREEN START HIBYTE
COLRLO=$00              ;COLOUR START LOBYTE
COLRHI=$D8              ;COLOUR START HIBYTE
BLOCKS=$03              ;3*256 BYTE BLOCKS
EXTRA  =$E8             ;EXTRA BYTES TO 1000
NMCOLS=$28              ;NO OF COLUMNS
NMROWS=$19              ;NO OF ROWS
;
*=$C200
;
MEMLO  *=*+1            ;START OF MEMORY
MEMHI  *=*+1            ;TO BE SCROLLED
COFFLO *=*+1            ;OFFSET TO COLOUR MAP
COFFHI *=*+1
NMSCRN *=*+1            ;NUMBER OF SCREENS
DELAY  *=*+1            ;DELAY LOOP VALUE
;
;++++ SET 38 COLUMN MODE ++++
;
       LDA SCRLRG
       AND #$F7
       STA SCRLRG
;
;++++ SET 1ST SCROLL POSITION ++++
;
       LDX NMSCRN
NEXSCR
       LDY #NMCOLS
START
       TXA
       PHA              ;PUSH X,Y REGS
       TYA              ;ONTO STACK
       PHA
;
       LDA SCRLRG
       AND #$F8
       CLC
       ADC #$07
       STA SCRLRG
;
;++++ COPY SCRN & COLR ONE LEFT ++++
;
       LDA #SCRNLO
       STA SCRPTR       ;SET UP Ø PAGE
       LDA #SCRNHI
       STA SCRPTR+1     ;POINTERS FOR
       LDA #COLRLO
       STA COLPTR       ;COPY
       LDA #COLRHI
       STA COLPTR+1
;
       LDX #BLOCKS
AGAIN
       LDY #$01
NEXT
       LDA (SCRPTR),Y
       DEY
       STA (SCRPTR),Y
       INY
       LDA (COLPTR),Y
       DEY
       STA (COLPTR),Y
       INY
       INY
       BNE NEXT
;
;++ COPY OVER PAGE BOUNDARY ++
       INC SCRPTR+1     ;INC HIBYTES
       INC COLPTR+1     ;OF Ø PAGE PTRS
       LDA (SCRPTR),Y
       DEC SCRPTR+1
       DEY
       STA (SCRPTR),Y;COPY OVER PAGE
       INY
       LDA (COLPTR),Y
       DEC COLPTR+1
       DEY
       STA (COLPTR),Y
       INC SCRPTR+1     ;INC Ø PAGE PTRS
       INC COLPTR+1     ;AGAIN
       DEX
       BNE AGAIN
;
;++ DO EXTRA BYTES ++
       LDY #$01
ANTHER
       LDA (SCRPTR),Y
       DEY
       STA (SCRPTR),Y
       INY
       LDA (COLPTR),Y
       DEY
       STA (COLPTR),Y
       INY
       CPY #EXTRA
       BCC ANTHER       ;IF Y<EXTRA REPEAT
;
;++++ INSERT RIGHT COLUMN OF SCREEN ++++
;
       LDA MEMLO
       STA MEMPTR
       LDA MEMHI        ;SET UP Ø PAGE
       STA MEMPTR+1     ;PTRS TO MEMORY
       LDA #NMCOLS-1
       STA SCRPTR       ;SET UP Ø PAGE
       LDA #SCRNHI      ;PTRS TO SCREEN
       STA SCRPTR+1
       JSR COPY40       ;COPY COLUMN
;
;++++ INSERT RIGHT COLUMN OF COLOUR ++++
;
       LDA MEMLO
       CLC
       ADC COFFLO       ;ADD OFFSET TO
       STA MEMPTR       ;COLOUR MAP
       LDA MEMHI        ;AND SET Ø PAGE
       ADC COFFHI       ;PTRS
       STA MEMPTR+1
       LDA #NMCOLS-1    ;SET UP Ø PAGE
       STA SCRPTR       ;PTRS TO COLOUR
       LDA #COLRHI      ;RAM
       STA SCRPTR+1
       JSR COPY40       ;DO COPY
;
;++++ SCROLL POSITIONS 6 TO Ø ++++
;
       LDX #$06
MORE1
       LDA SCRLRG
       AND #$F8
       STA SCRLRG
       TXA
       CLC
       ADC SCRLRG
       STA SCRLRG
;
       LDY DELAY        ;COUNT DOWN
MORE2
       DEY              ;DELAY VALUE
       BNE MORE2
       DEX
       BPL MORE1
;
;++++ INCREMENT MEMORY POINTER ++++
;
       LDA MEMLO
       CLC
       ADC #$01
       STA MEMLO
       LDA MEMHI
       ADC #$00
       STA MEMHI
;
;++++ CHECK FOR END OF MEMORY AREA ++++
;
       PLA
       TAY              ;GET X,Y REGS BACK
       PLA
       TAX              ;OFF STACK
       DEY
       BEQ NOJMP
       JMP START
NOJMP
       LDA MEMLO
       CLC
       ADC #$C0         ;ADD 1000-40
       STA MEMLO        ;TO MEMORY POINTER
       LDA MEMHI
       ADC #$03
       STA MEMHI
       DEX
       BEQ RETRN
       JMP NEXSCR
RETRN
       RTS
;
;++++ COPY EVERY 40TH BYTE S/R ++++
;
COPY40
       LDX #NMROWS
       LDY #$00
REPEAT
       LDA (MEMPTR),Y
       STA (SCRPTR),Y
       DEX
       BEQ FINISH       ;EVERY ROW DONE?
;
       LDA SCRPTR
       CLC
       ADC #NMCOLS      ;INC PTRS BY 40
       STA SCRPTR
       LDA SCRPTR+1
       ADC #$00
       STA SCRPTR+1
;
       LDA MEMPTR
       CLC
       ADC #NMCOLS
       STA MEMPTR
       LDA MEMPTR+1
       ADC #$00
       STA MEMPTR+1
       JMP REPEAT
FINISH
       RTS
```

Daten auf Abruf

Ein großer Vorteil des Computereinsatzes an Schulen liegt im schnellen Zugriff auf große Datenbanken.

Der Zugriff auf eine „elektronische Bibliothek" bietet für Schüler viele Vorteile: Man kann sofort nachschlagen, muß das Klassenzimmer nicht verlassen, nicht einmal aufstehen, um Informationen zu bekommen. Zudem können Daten auf Computerbasis leicht aktualisiert werden, wogegen in Büchern enthaltene Daten statisch sind und nur durch Neudruck aktualisiert werden können. Hohe Druckkosten stellen ein ernstes Problem für die Versorgung mit Schulbüchern dar. Deshalb wird oftmals veraltetes Lehrmaterial eingesetzt. In einer mit Computern ausgestatteten Klasse hingegen wäre lediglich der Austausch der Master-Diskette erforderlich.

Die Vorteile werden in der Kombination von Computer und Telekommunikation offensichtlich. Für den Zugriff zur „Enzyclopaedia Britannica" (in Deutschland vergleichbar dem „Großen Brockhaus") wäre das ideal. Natürlich wäre es ebenso einfach, ein Buch aus dem Regal zu nehmen. Wollte man aber alle entscheidenden Faktoren haben, um etwa herauszufinden, wie hoch der Ausbildungs-Etat in den verschiedenen Ländern liegt, wäre das ohne computerisierten Datenzugriff sehr mühsam.

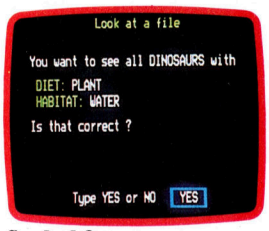
Die Enzyclopaedia Britannica umfaßt 26 Bände, und für geeignete Such- und Zugriffsroutinen wäre extrem viel Speicherkapazität erforderlich. Hier werden die Vorteile von Tele-Informationsdiensten deutlich. Zugang erfolgt mit einem standardisierten 300- oder 1200-Baud-Modem und einem normalen Telefonanschluß. Damit hat man Zugriff auf eine über-

Trotz der Ängste, daß die Anwendung von Computern an Schulen dazu führen könnte, Kinder in ihrer Entwicklung einzuengen, hat die Praxis bewiesen, daß speziell Datenbank-Software zu vielseitigen Aktivitäten führt. Vor der Eingabe müssen Informationen gesammelt werden. Dies erfolgt auf unterschiedliche Art. Eine Methode ist der „Gang durch die Natur", womit Schüler aufgefordert werden, die Umgebung zu beobachten und entsprechend zu klassifizieren.

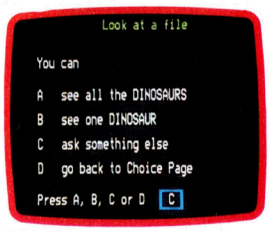
Dino-Daten

DATEINAME	FELD	BIS 10 MERKMALE		
DINO	DINOSAURIER	MERKMAL 1	MERKMAL 2	MERKMAL 3
RANG-NUMMER		GROESSE	ERNAEHRUNG	LEBENSRAUM
1	WINNOPTERUS	1	PFLANZEN	LAND
2	HOLGOTAURUS	12	PFLANZEN	LAND
3	JOJOFERIX	3	PFLANZEN	LAND
4	RAMAZOTTUS	4	PFLANZEN	LAND
5	ELKOSAURUS	5	PFLANZEN	LAND
6	PETROFISILUS	3	PFLANZEN	LAND
7	PLESIOSAUR	2	PFLANZEN	LAND
8	QUETSCHENTUS	28	INSEKTEN	LAND
9	BRACHASIOR	19	INSEKTEN	LAND
10	COMPUTERUS	4	INSEKTEN	LAND
11	POLACANTUS	7	INSEKTEN	LAND
12	HYPSILIOPHODON	9	INSEKTEN	LAND
13	UTALASAURUS	11	FLEISCH	LUFT
14	RICADERUS	10	FLEISCH	LUFT
15	NEUHAUSAUR	15	FLEISCH	LUFT
16	SEIDELENSI OR	6	FISCHE	LUFT
17	PETERANODON	7	FISCHE	LUFT
18	DEINONYCHTUS	7	FISCHE	LUFT
19	IODILODRIUS	9	FISCHE	WASSER
20	QETZALOCATULUS	10	FISCHE	WASSER
				WASSER
				WASSER
				WASSER

„Factfile" ist ein komplettes Datenbank-System für jüngere Schüler, das eine hervorragende Einführung in den Umgang mit Datenbanken gibt. Das Programm enthält ein „Dino" betiteltes Beispiel, das Informationen über Dinosaurier enthält.



Suchabfragen



Dino-Menü

wältigende Informationsmenge. Die Inspec-Datenbank beispielsweise, zusammengestellt vom Institut der Elektronik-Ingenieure, bietet allein über zwei Millionen Eintragungen aus dem Gesamtbereich Elektrik und Elektronik. Die Royal Society of Chemists bietet ausgedruckte Forschungsunterlagen, Artikel und anderes Referenzmaterial an.

Datenbank für Ausbildung

Die Firma Dialog in den USA ist die weltgrößte Datenbank und wird vor allem in der Ausbildung intensiv genutzt. Sie bietet 200 verschiedene Banken, in denen über 100 Publikationen mit 100 Millionen Einzeldaten gespeichert sind, die von der Kunstgeschichte bis zur Zoologie reichen. Bei der Dialog-Abfrage wird man nicht nur auf Autoren und Titel aller wichtigen Schriften und Artikel verwiesen, sondern bekommt eine Zusammenfassung, die alle benötigten Informationen enthält. Diese Daten können selbstverständlich auch auf Wunsch ausgedruckt werden.

Eine andere Art der Zugriffsmöglichkeit bietet Prestel mit seinem „Schools Link". Hiermit ist Schulen die Möglichkeit gegeben, Informationen zum Thema Computer zu erlangen, womit ein neues Ideen- und Kontakt-Forum geschaffen wurde. Es wendet sich mehr an Lehrer denn an Schüler und bietet Rezensionen von Software, Lern-Robotern, Büchern sowie Projektvorschläge. Das Angebot kann in den Schulcomputer geladen werden.

Ergänzend zur „elektronischen Bibliothek" werden Datenbanken im Klassenzimmer immer populärer und spielen dort eine aktive Rolle. Es gibt bereits mehrere Programme, mit denen Schüler Informationen nach eigenen Wünschen abfragen und strukturieren können.

Anläßlich einer Pädagogik-Konferenz an der Universität von Cambridge im Jahre 1981 wurde der Einsatz von Datenbanken diskutiert. Diskussionsergebnis war die Erstellung eines Programms mit Namen „Factfile", das es selbst Kleinkindern ermöglichte, Dateien nach eigener Wahl zu erstellen.

Die Bedeutung einer Datenbank wie Factfile beim Erforschen und Lernen wird schnell offensichtlich. Angenommen, ein Kind will eine Datenbank erstellen, die Informationen über seine Lehrer enthält. Zunächst muß es darüber nachdenken, wie die Daten zu kategorisieren und zu organisieren sind. Dazu gehören Fakten wie Alter, Größe, Name, Geschlecht und „Art des Lehrers". Ist die Strukturierung erfolgt, müssen Informationen für die entsprechenden Kategorien gesammelt werden. In dieser Phase entwickelt sich die Übung zu einem ganz anderen Lernvorgang: Zumindest muß das Kind Informationen sammeln. Werden Parameter benötigt, so sind diese sowie die darin enthaltenen Daten zu bestimmen. Der Vorgang wird noch komplexer, wenn das Kind zu verstehen beginnt, was die selbst geschaffenen Kategorien bedeuten. Das Feld „Art des Lehrers" etwa setzt eine Diskussion voraus. Bedeutet das „In welchem Fach unterrichtet er?" oder „Ist der Lehrer fair oder ungerecht?"

Ist der Aufbau der Datenbank beendet, üben andere Kinder Kritik daran und bringen ergänzende Informationen oder nehmen Korrekturen vor. Beispielsweise sollen alle Lehrerinnen, die als „fair" kategorisiert worden sind, aufgelistet werden. Der Lernprozeß schreitet weiter fort, wenn die Kinder beginnen, die Ergebnisse zu analysieren. Factfile erlaubt den Kindern, die Datenbanken auf Diskette oder Cassette zu speichern und zu einem späteren Zeitpunkt weitere Informationen hinzuzufügen.

Einer der Kritikpunkte an der Ausbildung mit dem Computer war, daß Kinder vor dem Bildschirm sitzen und Dinge tun, die keinen Zusammenhang mit der Realwelt haben. Doch neben der unmittelbaren Erfahrung mit Datenbanken gibt Factfile Kindern die Möglichkeit, unzählige andere Dinge zu tun und Ideen umzusetzen. Wurde beispielswiese eine Datei mit der örtlichen Flora angelegt, muß das Kind die Details zu jeder Pflanze sorgfältig ermitteln und dann ein eigenes Klassifizierungssystem schaffen, um eine Datenbank aufbauen zu können. Eine Londoner Schule entwickelte eine Datenbank als Teil des Sozialkundeunterrichts und benutzte dabei die Ergebnisse einer großangelegten Umfrage.

Programmpaket „Your Facts"

Ein anderes populäres Programmpaket, für kleinere Kinder entwickelt, heißt „Your Facts". Die Kinder werden nach Namen und Geschlecht gefragt, ob sie ein Haustier haben, eine Uhr, ein Fahrrad oder Geschwister. Ist die Information für ein Kind eingegeben, fragt das

Programm, ob der nächste „Kandidat" seine Informationen eingeben möchte. Es ist Platz für bis zu 40 Kinder-„Fakten" vorhanden, so daß eine ganze Klasse mitmachen kann. Nach Eingabe aller Informationen und einem „NEIN" als Antwort auf die Frage „WILL SONST NOCH JEMAND EINE EINGABE MACHEN?", werden die Kinder zum Menü zurückgeführt, aus dem alle Aufzeichnungen ersichtlich sind und feststellbar ist, welches Kind in welche Kategorie fällt – beispielsweise die Besitzer einer Uhr oder eines Fahrrads – oder die Kinder können ein Spiel starten: Der Computer errät den Namen eines Kindes, nachdem er ihm einige Fragen gestellt hat. So etwa „BIST DU EIN JUNGE?", „HAST DU EINE UHR?" und so weiter. Dann fragt er zum Beispiel „IST DEIN NAME MICHAEL?" „Your Facts" ist eine ideale Einführung für kleine Kinder, um sie mit dem Prinzip einer Datenbank vertraut zu machen. Es fördert das Lesen und Schreiben, stellt persönlichen Bezug her und ist zudem auf lehrreiche Weise unterhaltsam.

Erdkunde-Programme

Das australische Software-Haus „Active Learning Systems" hat ein sehr umfangreiches Datenbank-Paket entwickelt, das Informationen über fast jedes Land der Welt enthält. Das Programm heißt „One World" und verfügt auch über Anmerkungen für Lehrer, sowie Arbeitsblätter für die Klasse. Zu jedem Land sind 30 Dateneinheiten vorhanden, so unter anderem Fakten über Regierungsform, Import, Export, Sprachen, Nachbarländer, Religion, Bildungsstand und Geschichte. Man erfährt den Anteil der Land- und Stadtbevölkerung ebenso wieden Prozentsatz der Arbeiter, die in Grundindustrien arbeiten, ferner die Anzahl derer, die in der Schwerindustrie und Dienstleitungsunternehmen tätig sind. Dazu wurden zahlreiche Angaben über Ödland, Wald und Kulturland aufgezeichnet.

Im Hauptmenü stehen folgende Optionen:
1. EIN LAND DARSTELLEN
2. SUCHE UNTER VERWENDUNG SPEZIELLER INFORMATION
3. ANALYSIERE UNTER VERWENDUNG VERSCHIEDENER KRITERIEN
4. HILFE — REFERENZ
5. BEENDEN

Die ANALYSE-Option ist der interessanteste Teil des Programms. Kinder haben damit die Möglichkeit, unter Verwendung von bis zu drei Optionen Informationen zu analysieren. Sie könnten zum Beispiel feststellen, in welchem europäischen Land der Großteil der Bevölkerung auf dem Land lebt. Dazu wären nur zwei Optionen erforderlich: Region und Prozentsatz der Bevölkerung. Ist beispielsweise bekannt, daß die Bauxit-Bergarbeiter in Surinam streiken, könnte man schnell feststellen, wo sich das Land befindet, daß Bauxit das Hauptexportgut ist und daß der Arbeitskampf verhängnisvolle Auswirkungen auf die Gesamtwirtschaft des Landes hat.

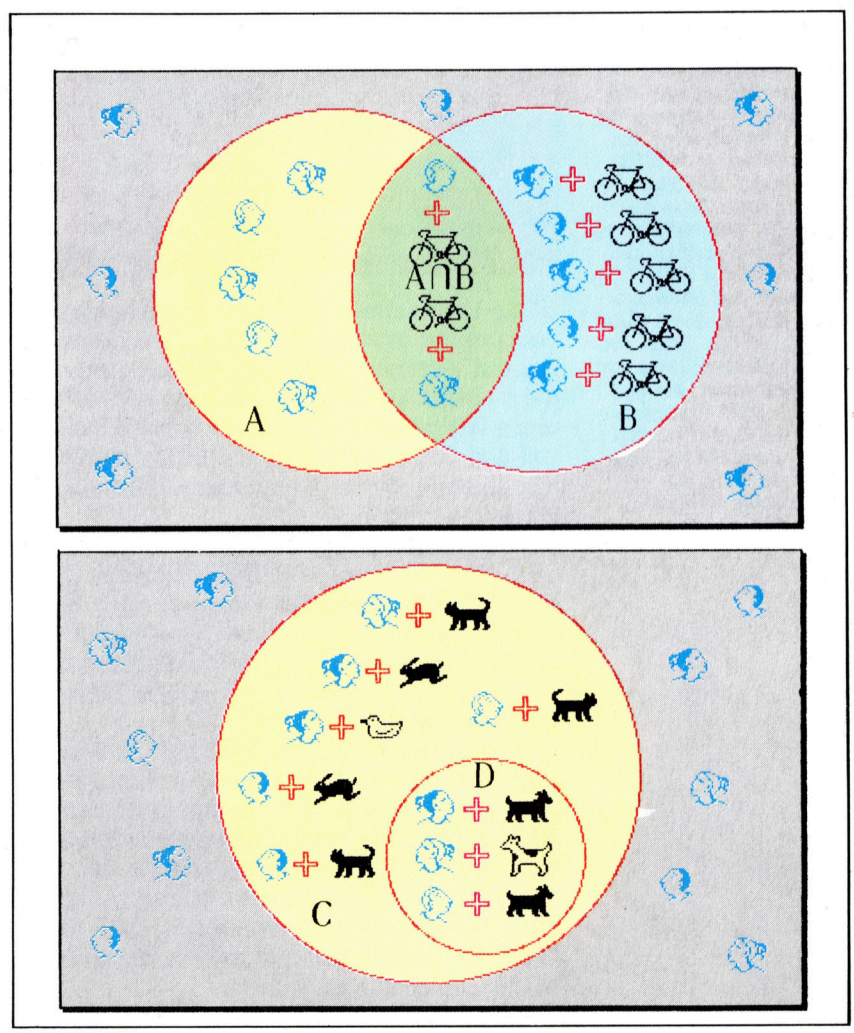
„One World" lehrt Forschen und fördert die Fähigkeiten der Datenauswertung. Er ermutigt Kinder, Schlüsse aus ihrer Datenanalyse zu ziehen. Gerade beim Geschichts-, Erdkunde- und Sozialkunde-Unterricht bietet es eine große Hilfe.

Eine Datenbank bietet Kindern eine Fülle von Lernmöglichkeiten, ermutigt sie, nach Antworten zu suchen. Sie können so sehen, wie ein Computer Informationen verarbeitet und verstehen die Wichtigkeit der richtigen Fragestellung und die genaue Eingabe-Organisation. Sind aber erst einmal Dateien erstellt, so ist es leicht, auf Dateien fußende, veränderliche Systeme im Unterricht zu behandeln. Es können Simulationen angewandt werden, die sonst nicht so leicht vorstellbare Verknüpfungen erklären.

Ein solches System wäre beispielsweise das ökologische Gleichgewicht einer Landschaft, auf das man im Computer versuchsweise „eingreifen" kann. So lernen Schüler am Rechner viel über ihre Umwelt.

Aufbau und Abfrage von Datenbanken ist anderen Lernmethoden durchaus ähnlich. Moderne Mathematik-Lehrmethoden (Mengenlehre) führen Kinder schon in frühem Alter an diese Arbeitsart heran. Dabei geht es um die Klassifizierung von Daten und die Verbindung zwischen Klassifizierungen.

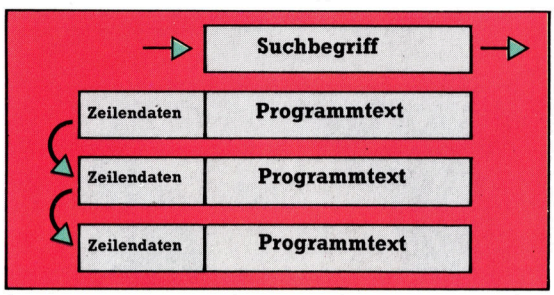
„Intersektion" und Untermengen sind bei der Datenbankabfrage besonders wichtig. Intersektion ist die Überlappung zweier Mengen, so etwa die Menge blonder Kinder und die Menge der Kinder, die Fahrräder besitzen. Die Intersektion stellt die Menge blonder Kinder, die ein Fahrrad besitzen, dar. Ein spezieller Fall der Intersektion ist die Untermenge, die in einer anderen Menge enthalten ist (zweites Beispiel).



Vom Nützlichsten

Programme, die Editiermöglichkeiten oder andere Hilfen anbieten, nennt man „Utilities". Wir beginnen eine Serie, in der wir diverse Hilfsprogramme für die bekannteren Heimcomputer untersuchen und zusätzliche BASIC-Utilities entwickeln.

Das Angebot an Utilities für Heimcomputer variiert erheblich: Einige verfügen nur über einen einfachen Editor, wie der Sinclair Spectrum, andere dagegen über komfortable Hilfen. Der Acorn B bietet zum Beispiel den TRACE und RENUMber-Befehl: TRACE bewirkt, daß die Zeilennummer jeder in Ausführung befindlichen BASIC-Anweisung dargestellt wird, und RENUM ermöglicht das automatische Neunumerieren von BASIC-Programmzeilen. Beide Befehle sind bei der Programmentwicklung sehr nützlich. Doch gleichgültig, welche Fähigkeiten in Ihrem Computer bereits vorhanden sind, ist es ausgesprochen hilfreich, zusätzlich weitere gute Utilities zur Verfügung zu haben.



Beim Acorn B und Spectrum beginnt eine Programmzeile mit drei oder vier Bytes, die der Zeilennummer und der Zeilenlänge zugeordnet sind. Danach folgt der verschlüsselte BASIC-Text. Wenn das Programm eine Zeile untersucht, verzeichnet es die Zeilennummer und berechnet die Startadressse der nächsten Zeile mittels der Länge der aktuellen Zeile. Danach durchsucht es den Programmtext, bis das Zeilen- oder Programmende (nach der letzten Zeile) erreicht ist, oder ein REM bzw. ein erfolgreicher Vergleich gefunden wurde.

Utility-Programme werden meistens in Maschinensprache geschrieben. Trotzdem beginnen wir mit einigen einfachen Hilfsprogrammen, die in BASIC geschrieben werden können. Auf diese Art können wir unsere Aufmerksamkeit voll auf das richten, was ein Utility eigentlich macht, ohne uns mit komplizierten Details befassen zu müssen.

Wir werden ein BASIC-Programm entwikkeln, das ein anderes BASIC-Programm bearbeitet. Das hier gezeigte Listing für den Spectrum, den C 64 und den Acorn B durchsucht ein BASIC-Programm nach einem Variablennamen und gibt die Zeilennummern aus, in denen er gefunden wurde.

Wegen der großen Ähnlichkeit stellen wir zunächst die Suchprogramme für Acorn B und Spectrum vor. Beide Programme beginnen damit, den Start des Programmtextes im Speicher zu suchen. Danach untersuchen sie das Programm Zeile für Zeile und extrahieren alle Namen. Abschließend wird jeder gefundene Name mit dem Namen verglichen, der gesucht werden sollte.

Wenn das Programm seine Suche mit einer neuen Zeile fortsetzt, merkt es sich zuerst die Zeilennummer, die jeweils in zwei Bytes gespeichert wird, sowie die Zeilenlänge (die Anzahl der durch sie belegten Bytes). Beim Acorn B wird die Zeilenlänge in einem Byte gespeichert und entspricht der Gesamtzahl an Bytes von der Zeilennummer bis zur Endmarkierung (ASCII-Code 13). Beim Spectrum wird die Zeilenlänge in zwei Bytes abgelegt und repräsentiert die Anzahl der Bytes vom Zeichen, das dem Längenbyte folgt, bis zur Endmarkierung (Längenbyte und Zeilennummer nicht eingeschlossen).

In beiden Programmversionen ignorieren wir alle REM-Anweisungen und alles, was in Anführungszeichen steht, da normalerweise in Strings keine Programmvariablen vorkommen. Der Acorn B gestattet ihnen, Hexadezimalzahlen zu verwenden, denen das Zeichen & vorangestellt wird. Um sicherzustellen, daß das Programm nicht irrtümlich diese Zahlen als Variablennamen interpretiert, müssen alle Zeichenfolgen mit vorangestelltem & übersprungen werden. Sonst könnte zum Beispiel die Hexzahl &A0 als Variablenname A0 mißdeutet und genutzt werden.

Beim Spectrum werden Zahlen in einem Programm als ASCII-Zeichen für die Stellen der Zahl, gefolgt von ASCII-Code 14 und fünf Bytes mit dem binären Äquivalent der Zahl, gespeichert. Unser Programm muß in der Lage sein, den Zahlencode und die fünf Binär-Bytes zu überspringen.

String-Variablen

Nach Überprüfung dieser Bedingungen fährt das Programm fort, die aktuelle Zeile nach Namen zu untersuchen. In beiden Programmen ist ein Name so definiert, daß er mit einem Buchstaben beginnt, gefolgt von einem anderen Buchstaben oder einer Zahl. Die Acorn-B-Version erlaubt Integer-Variablen (zu unterscheiden durch % nach dem Namen) und das Unterstreichzeichen. Beide Versionen erlauben String-Variablen, die durch ein $-Zeichen gekennzeichnet sind.

Der Name eines Arrays, einer Funktion oder (beim Acorn B) einer Prozedur wird von einer offenen Klammer „(" gefolgt. Sie ist zwar kein direkter Bestandteil des Namens, doch dient sie der Unterscheidung von Variablen.

Beim Spectrum gibt es weitere Probleme. Beispielsweise unterscheidet das Spectrum-BASIC nicht zwischen Groß- und Kleinbuchstaben in Variablennamen, so daß FRED, Fred oder FRed identisch behandelt werden. Daher muß das Spectrum-Programm vor dem Vergleich der Namen alle Buchstaben in Großbuchstaben umwandeln. Der Spectrum erlaubt auch die Verwendung von Leerstellen in Variablennamen, doch kann das zu Problemen führen, so daß wir von dieser Methode abraten.

Spectrum-BASIC macht im Gegensatz zu den meisten BASIC-Dialekten nicht die strikte Unterscheidung zwischen String-Variablen und String-Arrays, so daß eine String-Variable an sich ein Zeichen-Array ist. So ist es etwa möglich, T$ und T$(i) zu verwenden, wo einerseits ein String und andererseits ein Teil desselben Strings angesprochen wird. Daher unterscheidet unser Programm nicht zwischen einfachen String-Variablen und String-Arrays. Das ist jedoch keine ernsthafte Einschränkung, da Spectrum-BASIC es nicht gestattet, eine String-Variable und ein Array mit demselben Namen zu verwenden.

Geben Sie das Hilfsprogramm ein und speichern Sie es. Verbinden Sie es dann beim Spectrum mittels MERGE mit dem Programm, das Sie bearbeiten wollen, und beim Acorn B mit der im Handbuch beschriebenen Methode.

Spectrum

```
9000 INPUT "Name to search for? "; LINE T$
9010 FOR i=1 TO LEN (T$)
9020 IF T$(i)>="a" AND T$(i)<="z" THEN
LET T$(i)=CHR$ (CODE (T$(i))-32)
9030 NEXT i
9040 LET TokenforREM=234
9050 LET Quote=34
9060 LET Newline=13
9070 LET Underscore=95
9080 LET Number=14
9090 LET PROG=23635
9100 LET Textpointer=PEEK (PROG)+256*PEE
K (PROG+1)
9110 LET Lineno=256*PEEK (Textpointer)+P
EEK (Textpointer+1)
9120 IF Lineno>=9000 THEN  STOP
9130 LET Textpointer=Textpointer+2
9140 LET Textlength=PEEK (Textpointer)+2
56*PEEK (Textpointer+1)
9150 LET Textpointer=Textpointer+2
9160 LET Nextline=Textpointer+Textlength
9170 IF PEEK (Textpointer)=Newline THEN
 LET Textpointer=Textpointer+1: GO TO 91
10
9180 IF PEEK (Textpointer)<>TokenforREM
THEN  GO TO 9220
9190 REM Skip over REM line
9200  LET Textpointer=Nextline
9210 GO TO 9110
9220 IF PEEK (Textpointer)<>Quote THEN
GO TO 9280
9230 REM Skip anything between quotes, b
ut stop at end of line in case of unmatc
hed quote
9240 LET Textpointer=Textpointer+1
9250 IF PEEK (Textpointer)=Newline THEN
 LET Textpointer=Textpointer+1: GO TO 91
10
9260 IF PEEK (Textpointer)<>Quote THEN
GO TO 9240
9270 LET Textpointer=Textpointer+1
9275 GO TO 9170
9280 IF PEEK (Textpointer)<>Number THEN
 GO TO 9320
9290 REM Skip 5-byte binary number
9300 LET Textpointer=Textpointer+6
9305 GO TO 9170
9310 REM First character of name must be
 upper or lower case letter
9320 IF PEEK (Textpointer)>=CODE ("A") A
ND PEEK (Textpointer)<=CODE ("Z") THEN
LET c$=CHR$ (PEEK (Textpointer)): GO TO
9370
9330 REM Use upper case instead of lower
 case
9340 IF PEEK (Textpointer)>=CODE ("a") A
ND PEEK (Textpointer)<=CODE ("z") THEN
LET c$=CHR$ (PEEK (Textpointer)-32): GO
TO 9370
9350 LET Textpointer=Textpointer+1
9360 GO TO 9170
9370 LET n$=""
9380 LET n$=n$+c$
9390 LET Textpointer=Textpointer+1
9400 REM Letter, digit or underscore aft
er first character of name
9410 IF PEEK (Textpointer)>=CODE ("A") A
ND PEEK (Textpointer)<=CODE ("Z") THEN
LET c$=CHR$ (PEEK (Textpointer)): GO TO
9380
9420 REM Use upper case instead of lower
 case
9430 IF PEEK (Textpointer)>=CODE ("a") A
ND PEEK (Textpointer)<=CODE ("z") THEN
LET c$=CHR$ (PEEK (Textpointer)-32): GO
TO 9380
9440 IF PEEK (Textpointer)>=CODE ("0") A
ND PEEK (Textpointer)<=CODE ("9") THEN
LET c$=CHR$ (PEEK (Textpointer)): GO TO
9380
9450 IF PEEK (Textpointer)=Underscore TH
EN  GO TO 9380
9460 REM End with $ fo. string variable
9470 IF PEEK (Textpointer)=CODE ("$") TH
EN  LET n$=n$+"$": LET Textpointer=Textp
ointer+1: GO TO 9500
9480 REM ( if array or function
9490 IF PEEK (Textpointer)=CODE ("(") TH
EN  LET n$=n$+CHR$ (PEEK (Textpointer)):
 LET Textpointer=Textpointer+1
9500 IF n$=t$ THEN  PRINT n$;" IN LINE "
;Lineno
9520 GO TO 9170
```

Acorn B

```
30000 INPUT "Name to search for";
TARGET$
30010 TokenforREM=244
30020 Quote=34
30030 Hex=38
30040 Newline=13
30050 Underscore=95
30060 Textpointer=PAGE
30070 Textpointer=Textpointer+1
30080 Lineno=256*?Textpointer+?(Textpoin
ter+1)
30090 IF Lineno>=30000 THEN END
30100 Textpointer=Textpointer+2
30110 Linelength=?Textpointer
30120 Endline=Textpointer+Linelength-3
30130 Textpointer=Textpointer+1
30140 IF ?Textpointer=Newline THEN GOTO
30070
30150 IF ?Textpointer<>TokenforREM THEN
GOTO 30180
30160    REM Skip over REM line
30170    Textpointer=Endline:GOTO 30070
30180 IF ?Textpointer<>Quote THEN GOTO 3
0240
30190    REM SKip antything between quot
es, but stop at end of line in case of u
nmatched quote
30200    Textpointer=Textpointer+1
30210    IF ?Textpointer=Newline THEN GO
TO 30070
30220    IF ?Textpointer<>Quote THEN GOT
O 30200
30230    Textpointer=Textpointer+1
30235 GOTO 30140
30240 IF ?Textpointer<>Hex THEN GOTO 303
00
30250    REM Skip hex number, to avoid c
onfusion with variable names
30260    Textpointer=Textpointer+1
30270    IF ?Textpointer>=ASC("0") AND ?
Textpointer<=ASC("9") THEN GOTO 30260
30280    IF ?Textpointer>=ASC("A") AND ?
Textpointer<=ASC("F") THEN GOTO 30260
30285    GOTO 30140
30290 REM First character of name must b
e upper or lower case letter
30300 IF ?Textpointer>=ASC("A") AND ?Tex
tpointer<=ASC("Z") THEN GOTO 30330
30310 IF ?Textpointer>=ASC("a") AND ?Tex
tpointer<=ASC("z") THEN GOTO 30330
30320 GOTO 30130
30330 Name$=""
30340 Name$=Name$+CHR$(?Textpointer)
30350 Textpointer=Textpointer+1
30360 REM Letter, digit or underscore af
ter first character
30370 IF ?Textpointer>=ASC("A") AND ?Tex
tpointer<=ASC("Z") THEN GOTO 30340
30380 IF ?Textpointer>=ASC("a") AND ?Tex
tpointer<=ASC("z") THEN GOTO 30340
30390 IF ?Textpointer>=ASC("0") AND ?Tex
tpointer<=ASC("9") THEN GOTO 30340
30400 IF ?Textpointer=Underscore THEN GO
TO 30340
30410 REM End with $ for string variable
, % for integer variable
30420 IF ?Textpointer=ASC("$") THEN Name
$=Name$+CHR$(?Textpointer):Textpointer=T
extpointer+1:GOTO 30450
30430 IF ?Textpointer=ASC("%") THEN Name
$=Name$+CHR$(?Textpointer):Textpointer=T
extpointer+1
30440 REM ( if array, procedure or funct
ion
30450 IF ?Textpointer=ASC("(") THEN Name
$=Name$+CHR$(?Textpointer):Textpointer=T
extpointer+1
30460 IF Name$=TARGET$ THEN PRINT Name$;
" IN LINE ";Lineno
30470 GOTO 30140
30480 END
```

Starten Sie das Programm mit RUN 9000 (Spectrum) oder GOTO 30000 (Acorn B), und geben Sie den gesuchten Variablennamen ein. Handelt es sich um einen Array-Namen, tippen Sie am Ende eine offene Klammer.

Das Programm zum Suchen von Variablennamen, das wir für den Acorn B und den Spectrum entwickelt haben, kann auch problemlos auf den C64 konvertiert werden. Das C64-Programm ist sogar etwas einfacher, da nicht so viele spezielle Fälle beachtet werden müssen.

NEWLINE geht nicht

Viele der Variablennamen in der Commodore-Version des Programmes müssen geändert werden, damit sie nicht BASIC-Schlüsselwörter beinhalten. So kann beispielsweise NEWLINE nicht verwendet werden, da der Name mit dem Wort NEW beginnt, und TEXTPOINTER beinhaltet das Wort INT.

Die Änderungen am Programmanfang sind aufgrund der Unterschiede beim Speichern einer BASIC-Programmzeile notwendig. Beim Acorn B und Spectrum beginnt eine BASIC-Zeile im internen Format mit einer Zwei-Byte Zeilennummer (höherwertiges Byte zuerst), gefolgt von einem oder zwei Bytes für die Zeilenlänge. Beim Commodore 64 dagegen beginnt eine BASIC-Zeile mit einem Zwei-Byte-Zeiger auf den Beginn der nächsten Zeile, und einer Zwei-Byte-Zeilennummer. In beiden Fällen ist das erste Byte das niederwertige.

Nach wie vor müssen REM-Zeilen und Strings innerhalb von Anführungszeichen übersprungen werden, doch entfallen solche Besonderheiten wie die Hexadezimalzahlen auf dem Acorn B oder die Binärdarstellung von Zahlen beim Spectrum.

Der Programmabschnitt, der die Variablennamen heraussucht, schaut zuerst nach Buchstaben, dann nach Zeichen oder Zahlen, und zuletzt nach den Zeichen $ oder %, die String- oder Integervariablen indizieren, sowie nach der „(", die eine Funktion oder ein Array kennzeichnet. Der C64 gestattet nicht das Unterstreichzeichen, das beim Acorn B und Spectrum in Variablennamen so einfach verwendet werden kann.

Obwohl der C64 sowohl Groß-, als auch Kleinbuchstaben darstellen kann, bedeuten sie nur einen Unterschied auf dem Bildschirm, nicht jedoch im internen Code für ein Zeichen. Somit braucht man nur nach Großbuchstaben in einem Variablennamen zu suchen.

„RUN 30000"

Die C64-Version des Programms wird in derselben Art verwendet wie die Acorn B und Spectrum-Versionen. Geben Sie das Programm ein und speichern Sie es. Laden Sie dann mit LOAD das zu bearbeitende Programm, und hängen Sie das Suchprogramm an. Zum Start des Programms geben Sie „RUN 30000" und den gesuchten Variablennamen ein (bei einem Array-Namen geben Sie am Ende eine offene Klammer ein).

Beim C64 gibt es eine einfache Methode, zwei gespeicherte Programme zu verbinden, vorausgesetzt, die Zeilennummern des ersten Programms sind alle niedriger als die des zweiten. Die Methode benutzt zwei Zeiger der Zero Page: TXTTAB bei den Adressen 43 und 44 gibt die Adresse an, bei der das BASIC-Programm startet, sowie VARTAB bei den Adressen 45 und 46. Ein BASIC-Programm endet mit einem Byte mit dem Wert 0, der das Ende der letzten Zeile des Programms angibt, gefolgt von zwei weiteren Null-Bytes, die das Programmende markieren. Die Adresse von VARTAB ist normalerweise des Byte hinter diesen letzten zwei Bytes. Zum Verbinden zweier Programme laden Sie zuerst das Programm mit den niedrigeren Zeilennummern und geben dann ein:

```
PRINT PEEK(45), PEEK(46)
```

Ist die erste Zahl ein Wert zwischen 2 und 255, subtrahieren Sie 2 und POKEn das Ergebnis in Adresse 43. Ist der Wert 0 oder 1, POKEn Sie 254 oder 255 in Adresse 43, sowie einen um 1 geringeren Wert als das Ergebnis von PEEK(46) in Adresse 44. Sie können nun schon das zweite Programm laden und eingeben:

```
POKE 43,1:POKE 44,8
```

Dadurch werden wieder die Ursprungswerte gesetzt, und die Programme sind nun verbunden wie Sie es wünschten.

Commodore 64

```
30000 INPUT "NAME TO SEARCH FOR"; T$
30010 RE=143
30020 QUOTE=34
30030 NL=0
30040 TEXTPTR=2049
30050 NXTLINE=PEEK(TEXTPTR)+256*PEEK(TEXTPTR+1)
30070 TEXTPTR=TEXTPTR+2
30080 LINENO=PEEK(TEXTPTR)+256*PEEK(TEXTPTR+1)
30085 IF LINENO>=30000 THEN END
30090 TEXTPTR=TEXTPTR+2
30140 IF PEEK(TEXTPTR)=NL THEN TEXTPTR=TEXTPTR+1:GOTO 30050
30150 IF PEEK(TEXTPTR)<>RE THEN GOTO 30180
30160 REM SKIP OVER REM LINE
30170 TEXTPTR=NXTLINE:GOTO 30050
30180 IF PEEK(TEXTPTR)<>QUOTE THEN GOTO 30300
30190 REM SKIP ANYTHING IN QUOTES,STOP AT END OF LINE IN CASE OF UNMATCHED
      QUOTE
30200 TEXTPTR=TEXTPTR+1
30210 IF PEEK(TEXTPTR)=NL THEN TEXTPTR=TEXTPTR+1:GOTO 30500
30220 IF PEEK(TEXTPTR)<>QUOTE THEN GOTO 30200
30230 TEXTPTR=TEXTPTR+1
30235 GOTO30140
30290 REM FIRST CHARACTER OF NAME MUST BE LETTER
30300 IF PEEK(TEXTPTR)>=ASC("A") AND PEEK(TEXTPTR)<=ASC("Z") THEN GOTO 30330
30310 TEXTPTR=TEXTPTR+1
30320 GOTO30140
30330 NAME$=""
30340 NAME$=NAME$+CHR$(PEEK(TEXTPTR))
30350 TEXTPTR=TEXTPTR+1
30360 REM LETTER OR DIGIT AFTER FIRST CHARACTER
30370 IF PEEK(TEXTPTR)>=ASC("A") AND PEEK(TEXTPTR)<=ASC("Z") THEN GOTO 30340
30390 IF PEEK(TEXTPTR)>=ASC("0") AND PEEK(TEXTPTR)<=ASC("9") THEN GOTO 30340
30410 REM END WITH $ FOR STRING VARIABLE, % FOR INTEGER VARIABLE
30420 IF PEEK(TEXTPTR)=ASC("$") THEN NAME$=NAME$+"$":TEXTPTR=TEXTPTR+1
      :GOTO30450
30430 IF PEEK(TEXTPTR)=ASC("%") THEN NAME$=NAME$+"%":TEXTPTR=TEXTPTR+1
30440 REM ( IF ARRAY OR FUNCTION
30450 IF PEEK(TEXTPTR)=ASC("(") THEN NAME$=NAME$+"(":TEXTPTR=TEXTPTR+1
30460 IF NAME$=T$ THEN PRINT NAME$;" IN LINE";LINENO
30470 GOTO 30140
30480 END
```

Fachwörter von A bis Z

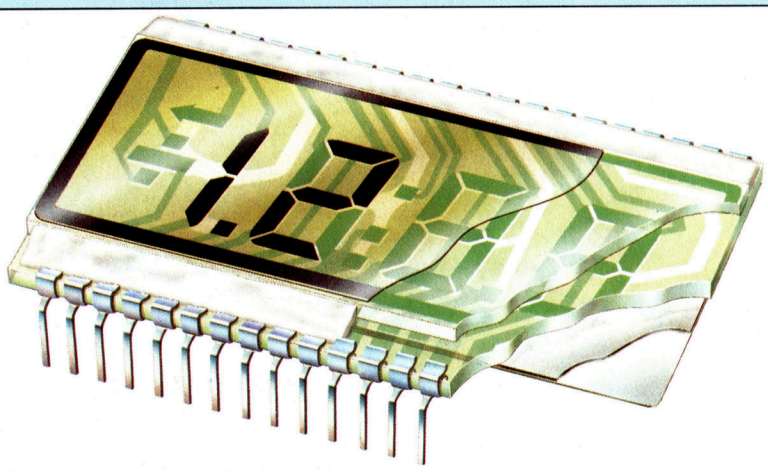
Hier werden einzelne Fachausdrücke eingehend behandelt. Da bei der Kommunikation mit dem Computer meist die englische Sprache verwendet wird, werden hier zunächst die englischen Begriffe genannt, dann die deutsche Übersetzung. In den Gesamtindex werden sowohl deutsche als auch englische Stichwörter aufgenommen,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

Language Construct = Sprachkonstrukt
Jede Programmiersprache verfügt über eine Reihe von Strukturelementen zur Realisierung häufig benötigter Funktionen. Dazu gehören „Schlüsselwörter" wie PRINT für die Datenausgabe über den Bildschirm.

Eine Programmiersprache muß aber auch kompliziertere Abläufe unterstützen können, die eine Folge von Einzelaktionen erfordern. Die entsprechenden Strukturen bezeichnet man als „Sprachkonstrukte". Ein Beispiel ist die Schleifenanweisung FOR... NEXT. Was innerhalb der Schleife geschieht, bestimmt dabei der Programmierer, aber der Aufbau der Schleife ist fest definiert. Konstrukte für Wertzuweisungen, Entscheidungen, Adressierung und Ein/Ausgabe sind wesentliche Elemente jeder Programmiersprache.

Laser Printer = Laserdrucker
Ein Laserdrucker ist ein hochwertiges Ausgabegerät, das hohe Druckgeschwindigkeiten erreicht. Die Zeichen werden als dichtes Punktmuster mit einem Laserstrahl zunächst auf die elektrisch geladene Oberfläche eines Fotohalbleiters geschrieben. Wo Licht auffällt, lädt sich der Halbleiter um; es entsteht ein „Ladungsbild". Dieses wird mit einem Tonerpulver „entwickelt", das an den belichteten Stellen haftenbleibt (ähnlich wie bei vielen Fotokopierern), und dann auf Papier übertragen und eingebrannt. Laserdrucker sind farb- und grafiktauglich und können beliebige Schriftarten erzeugen.

LCD = Flüssigkristallanzeige
LCDs (Liquid Crystal Displays) werden als Bildschirmersatz für Microcomputer eingesetzt. Alphanumerische Anzeigen sind als Siebensegment-Felder aufgebaut. Sie bestehen aus einer Sandwich-Anordnung von Elektrodenplatten und Polarisationsfiltern mit einer „kristallinen" Flüssigkeit zwischen den einzelnen Elementen und einer reflektierenden Unterlage. Bei angelegter Spannung zwischen den Elektroden wird die zuvor transparente Flüssigkeit lichtundurchlässig, und das betreffende Segment erscheint dunkel. Das Hauptargument für die Verwendung von LCDs, vor allem bei tragbaren Computern, sind ihr geringer Platz- und Leistungsbedarf. Ein wesentlicher Nachteil ist die große Trägheit: Die Anzeige kann mit dem Tempo einer geübten Schreibkraft nicht schritthalten.



Ein LCD enthält zwischen zwei Polarisationsfiltern eine dünne Flüssigkristallschicht. Die Glasplatten tragen an der Innenseite ein Elektrodenmuster, das den Zeichenkonturen entspricht. Ist Spannung an die Elektroden gelegt, werden die Flüssigkeitsmoleküle durch das elektrische Feld ausgerichtet. Dabei ändert sich die Transparenz, und die Zeichensegmente heben sich gegen den Hintergrund ab.

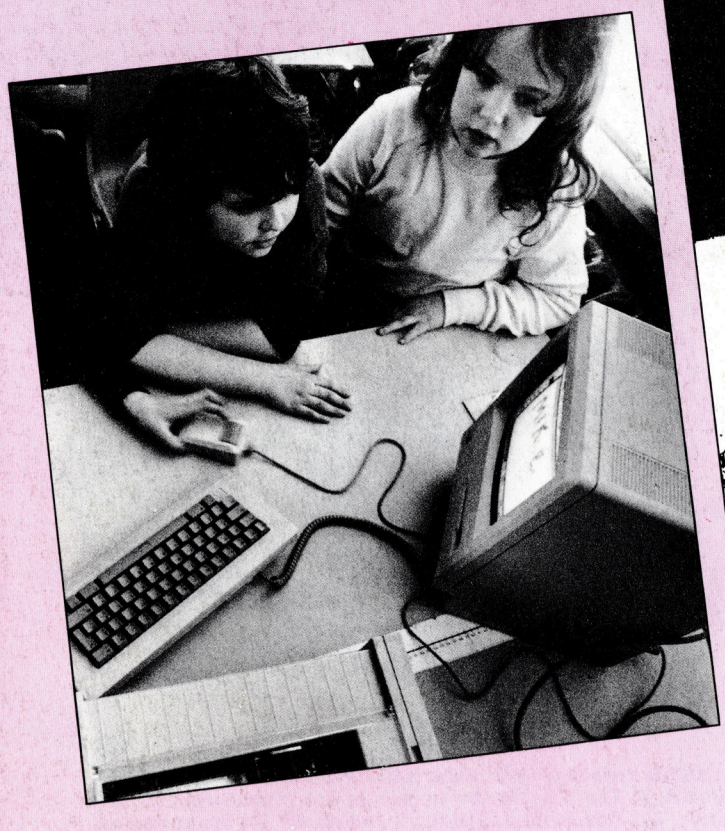
LED = Leuchtdiode
LED steht für „Light Emitting Diode", das heißt für eine Halbleiterdiode, die bei Stromdurchgang Licht emittiert. Einzelne LEDs werden bei Computern oft für Kontrolleuchten eingesetzt; außerdem lassen sich LEDs auch zu 7-Segment-Blöcken zusammensetzen und wie LCDs als alphanumerische Anzeigen verwenden. Davon wurde früher bei Taschenrechnern viel Gebrauch gemacht – in diesem Bereich sind die LEDs wegen ihres hohen Leistungsbedarfs heute weitgehend von den LCDs verdrängt worden.

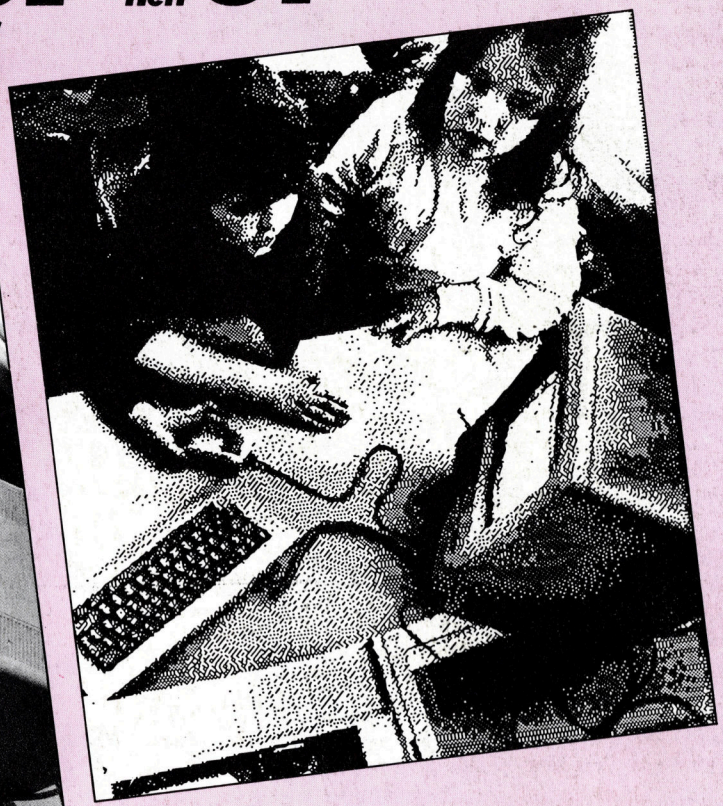
Lexical Analysis = Lexikalische Analyse
Weil die Verarbeitung von Programmen im Rechner nach strengen Regeln erfolgt, muß der eingegebene Programmtext vor dem eigentlichen Übersetzen in eine standardisierte Form gebracht werden. Das besorgt ein Teil des Compilers mit einem Satz spezieller Routinen durch die „lexikalische Analyse". Die Quellenanweisungen werden in ihre Bestandteile zerlegt, Zahlenkonstanten in die interne Darstellung gebracht, Variablennamen zusammengestellt.

Bildnachweise
1653: Marcus Wilson Smith
1655, 1671: Carolyne Clayton
1658–1660, 1666, 1667: Kevin Jones
1664, 1665, 1668, 1669, 1677: Ian McKinnell
1672, 1673: Liz Dixon
1675: Steve Cross
U4: Tony Sleep

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

computer kurs Heft 61





Die gedruckte Seite
Möglichkeiten, die Kosten im Grafikgewerbe mit Computereinsatz niedrig zu halten.

Der erste Block
Mit OSWORD lassen sich mehr als nur zwei Parameter einsetzen. So wird diese Modulgruppe eine gute Alternative.

Es ist soweit
Nach dem Zusammenbau unseres Robot-Arms geht es nun mit der Bewegung los: Die elektrischen Verbindungen.

Wortspiele
Wir simulieren Möglichkeiten der BASIC-Stringverarbeitung in FORTH.

Einkaufsbummel
Stärken und Schwächen von gängigen Computern auf einen Blick. Auswahlkriterien fürs Nachfolgemodell.