Einsteigen - Verstehen - Beherrschen

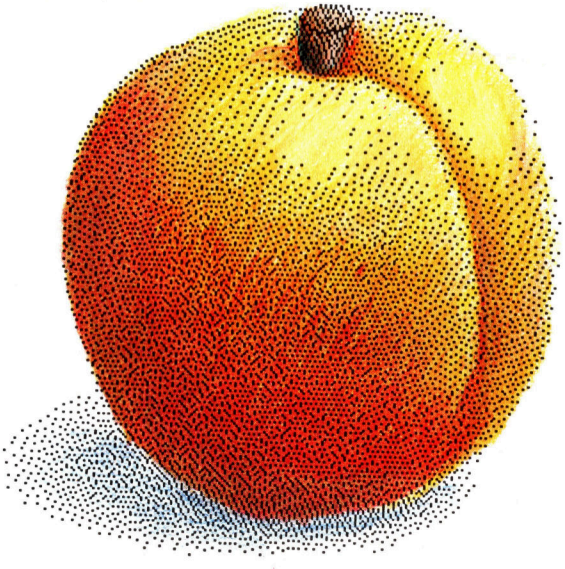
DM 3,80 öS 30 sfr 3,80

computer kurs

Heft 53

Ein wöchentliches Sammelwerk

Aa



F1e

Firmenportrait: Olivetti

Spectrum User Port

Software für Landwirte

PC Apricot F1e

computer kurs

Heft 53

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

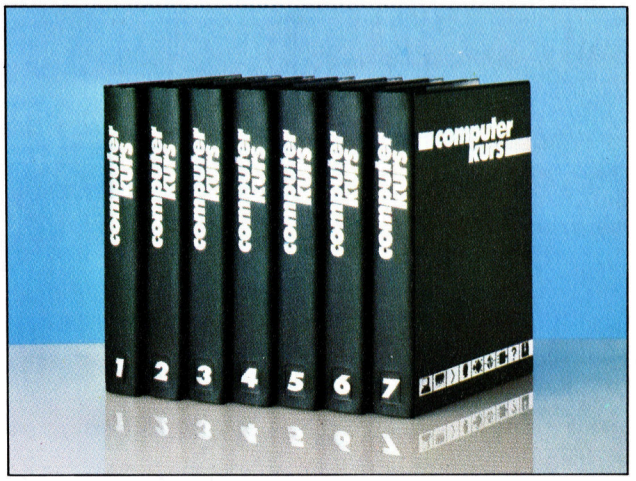
Schweiz: Der Sammelordner kostet sfr 15. Bitte wenden Sie sich an Ihren Kiosk; dort werden Sie jederzeit die gewünschten Exemplare erhalten.

INHALTSVERZEICHNIS

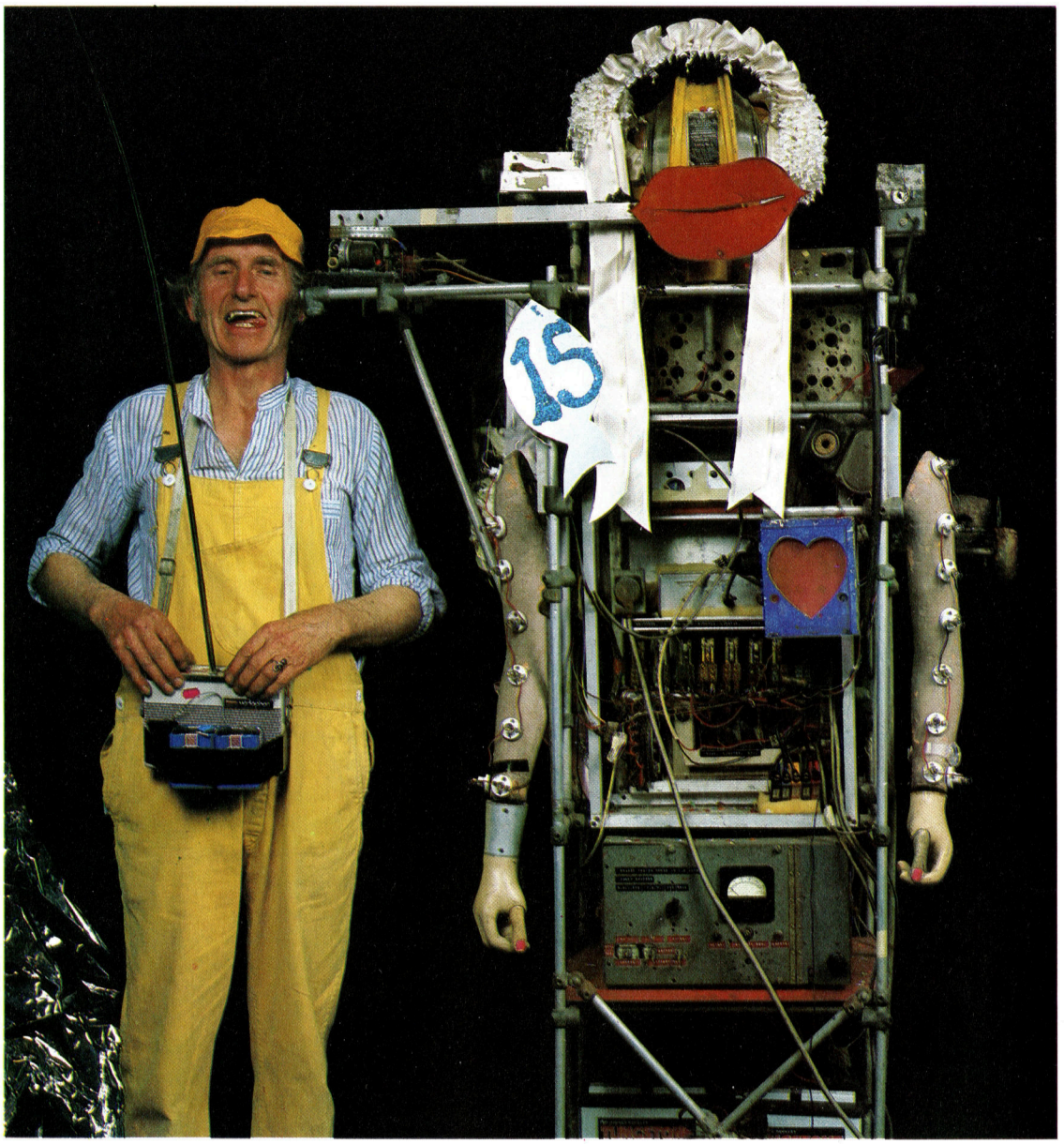
Alle 12 Hefte erscheint ein Teilindex. Die letzte Ausgabe von Computer Kurs enthält den Gesamtindex — darin einbezogen sind Kreuzverweise auf die Artikel, die mit dem gesuchten Stichwort in Verbindung stehen.

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

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



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



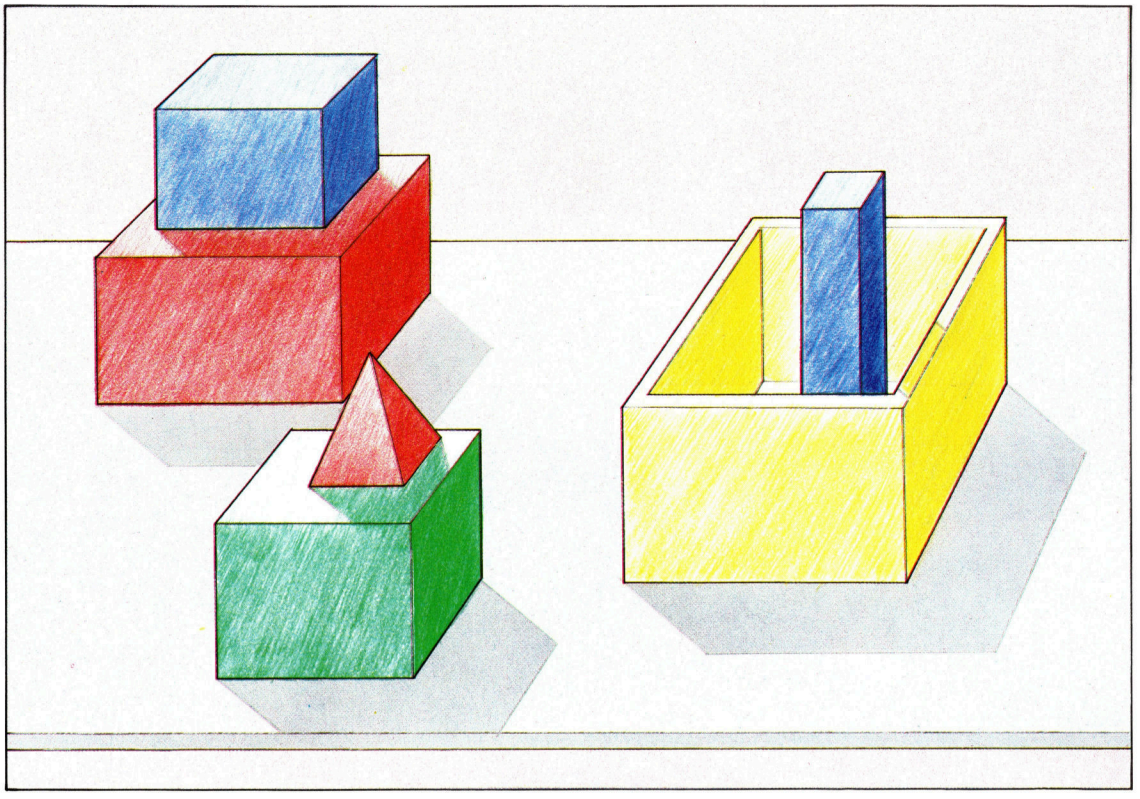
Man glaubt, daß Roboter eine sehr neue Erfindung seien, was die Umsetzung anbelangt. Doch Rosa Bosom (Radio-Operated Simulated Actress Battery Or Stand-by Operated Mains), die hier mit ihrem Erfinder Bruce Lacey im Bild zu sehen ist, wurde bereits 1966 gebaut, um die Königin von Frankreich in dem Bühnenstück „Die drei Musketiere" des Royal Court Theatre, London, zu spielen. Sie wurde aus Abfallteilen (Relais und Motoren) aus Regierungsbeständen gebaut. Rosa hört auf Befehle, die in Notenform gegeben werden, und setzt sie in Bewegungen um. Sie ist ferner mit Ultraschall-Sensoren ausgestattet, die das Erkennen und Umgehen von Hindernissen ermöglichen. Als Schauspielerin brachte Rosa ihre Worte von „Band" in den Raum, dabei wurde die Lippenbewegung synchron ferngesteuert. Besonders interessant ist der Umstand, daß Rosa mit einem zweiten Roboter namens Mate zusammenarbeiten kann. Rosa trat bei mehreren Veranstaltungen auf, unter anderem bei der Cybernetics Serendipity auf der ICA, und siegte bei der „alternativen" Miß-Welt-Wahl in London.

Geistesblitze

Wie unserer Serie über Künstliche Intelligenz unschwer zu entnehmen ist, gibt es keinen Mangel an Ideen, was ein Roboter tun sollte und möglicherweise auch tun wird. Die Probleme bei der Entwicklung intelligenter Roboter sind allerdings vielfältig.

In wohl keinem anderen Computerbereich liegen Theorie und Praxis so weit auseinander wie in der Robotik. Hollywoods Science-fiction-Roboter können gehen, sprechen und haben die Welt „im Griff". Doch nur wenige Meilen von Hollywood entfernt kann das jüngste Kind der Roboter-Entwicklung an der Stanford Universität nicht einmal ohne anzustoßen durch einen Raum gehen, in dem sich Menschen aufhalten. Natürlich spielen Roboter produktive Rollen in Autofabriken und anderen Zweigen der Industrie. Doch solche Systeme sind lediglich dazu da, Farbflächen zu sprühen oder zu schweißen, also nichts weiter zu tun, als vorprogrammierte Sequenzen ständig zu wiederholen. Sie sind so „dumm", daß sie etwa bei einem Stillstand des Förderbands Farbe in den leeren Raum sprühen würden.

Die Gründe für das Auseinanderklaffen zwischen Roboter-Theorie und -Wirklichkeit sind vielfältig. Sie lassen sich aber unter den Begriff „Wahrnehmung" und „Planung" einordnen. Zunächst einmal können die Roboter von heute mit ihren „Sensoren" nichts Sinnvolles



Blockwelten sind stark abstrahierte Modelle der Realwelt, die es KI-Forschern ermöglichen, unterschiedliche Methoden von Problemlösungen zu evaluieren. Die Umsetzung auf die Realwelt ist immer noch sehr schwer.

anstellen. Deshalb ist die Robotik ein ideales Experimentierfeld für Künstliche Intelligenz.

Die Robotik beansprucht alle Aspekte des Arbeitsbereichs der Künstlichen Intelligenz, da es ihr Ziel ist, ein Produkt zu schaffen, das mit der realen Welt fertig wird. Deshalb muß ein Roboter seine Umgebung sehr leicht erkennen können.

Es ist kein Problem, einen Computer an eine Reihe von Eingabe-Einheiten anzuschließen – zum Beispiel Fernsehkameras, Hitzesensoren, Ultraschalltaster, Druckmesser und dergleichen mehr –, die Zugriffsmöglichkeiten zu Informationen bieten, die der Mensch direkt nicht erreichen kann, so etwa Infrarot oder ultraviolettes Licht. Doch wenn der Roboter nicht über eine sehr genau definierte und gleichbleibende Umgebung verfügt, wird er nicht erkennen, was seine Sensoren ihm mitteilen.

Micromaus

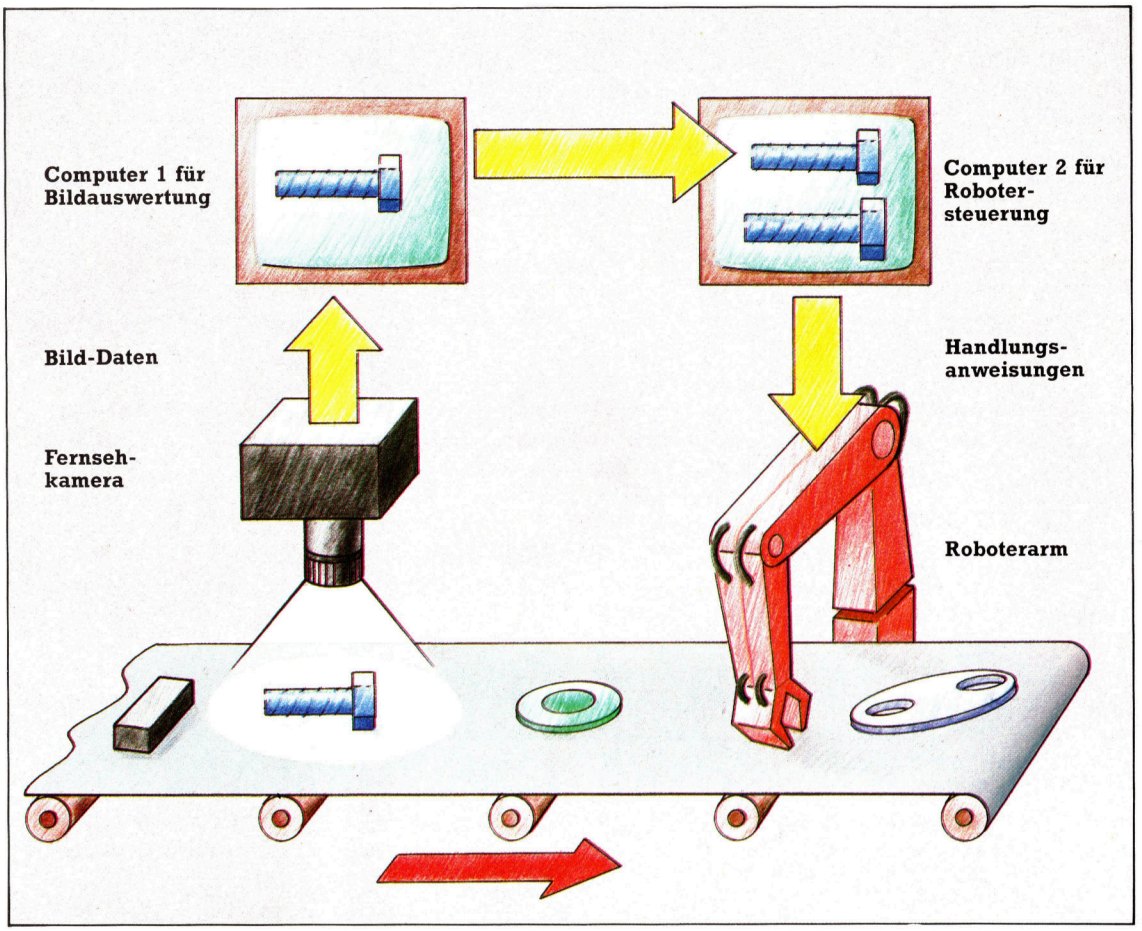
Es ist eine Sache, einen umfassenden Algorithmus zu schreiben, der einen Weg durch ein im Computerspeicher enthaltenes Labyrinth findet und diesen auf dem Bildschirm darstellt. Es ist jedoch weitaus schwieriger, mit demselben Algorithmus einen Roboter durch eine Stadt zu steuern, in der es Menschen, Tiere, Autos, weggeworfene Bierdosen, üppig wuchernde Hecken und andere unberechenbare Hindernisse gibt.

Darum sind „Micromaus-Wettbewerbe" so populär. Bei diesen Wettbewerben muß eine computergesteuerte „Maus" den Weg zum Mittelpunkt eines großen Labyrinths finden. Es gibt seit Jahren Methoden, die Maus ans Ziel zu führen – zumindest theoretisch. In der Praxis jedoch sind die Wände nicht völlig gerade, es gibt schlüpfrige Seitenstraßen, auf denen Vorläufer vielleicht Öl verloren haben, und dergleichen mehr. Unter solchen Voraussetzungen sind Robustheit und Anpassungsfähigkeit wichtiger als die Eleganz des Algorithmus.

In den meisten anderen Bereichen der KI können sich die Programmierer in selbstgemachte Microwelten flüchten, so zum Beispiel in ein Schachprogramm, das in völlig abstrakter Umgebung abläuft. Ein Expertensystem hat sich nur mit Fakten auseinanderzusetzen, nicht mit Dingen. Das erklärt, warum Forscher der Künstlichen Intelligenz so vernarrt in „Block-Welten" sind.

Eine Block-Welt ist eine vereinfachte, schematisierte Darstellung der Umgebung, die sich aus farbigen Bausteinen zusammensetzt. – Natürlich nicht aus echten Bauklötzen, sondern aus formalisierten Darstellungen dieser Steine. Denn den Wissenschaftlern ist daran gelegen, Probleme lösen und Objekte manipulieren zu können, ohne ihren Arbeitsplatz – also den Computer – zu verlassen.

Der Robotik-Wissenschaftler hat diese Möglichkeit nicht. Roboter werden mit den Widrigkeiten der Wirklichkeit konfrontiert und müssen damit fertig werden. Ein „intelligenter" Roboter – etwa einer, der sich aus eigener Initiative umherbewegen kann – muß ein Modell seiner Umgebung erstellen können, und er muß auf gesprochene Befehle reagieren können. Da aber nicht alle Reaktionen auf dem Vorwege planbar sind, ist eine gewisse Lern-



Ein sehender Roboter kann zur Erkennung und physischen Erfassung von Gegenständen auf einem Fließband eingesetzt werden. In diesem Fall sind für die Robotersteuerung zwei Computer erforderlich – einer, um die Eingangsdaten zu interpretieren, die über die Fernsehkamera geliefert werden, der andere, um den Greifmechanismus zu steuern.

fähigkeit wünschenswert. Das ist der Grund, warum es noch keine wirklich intelligenten Roboter gibt. Bevor diese gebaut werden können, müssen folgende Probleme der KI noch erst einmal gelöst werden:

- Optische Wahrnehmung,
- Spracherkennung,
- Problemlösung in sich verändernder Umgebung,
- Verarbeiten von Erfahrungen.

Das ist nur eine Liste der Mindestvoraussetzungen für einen Roboter.

Moderne Industrieroboter werden für klar strukturierte Umgebungen entwickelt. Typisch ist der Roboter, der nichts anderes als ein vom Computer gesteuerter Arm und Bestandteil einer Werkbank oder industriellen Fertigungsstraße ist. Seine „Hand" kann mit Werkzeugen versehen sein. Ein menschlicher Operator „lehrt" ihn die Schrittfolge bzw. Sequenz der auszuführenden Bewegungen, die er dann immer wiederholt. Er erinnert sich an diese Sequenzen, kann sie aber nicht abwandeln.

Einige Roboter jedoch haben bereits den ersten Schritt in die Realwelt getan. Sie sind etwas weniger vom Menschen abhängig oder von anderen Maschinen, die ihnen Arbeit zuliefern. In gewissem Umfang können sie sich selbst Arbeit suchen.

Die verbreiteteste Art der „Sinneswahrnehmung" ist das „Sehvermögen" eines Roboters. Eine Videokamera zeichnet Bilder auf (etwa eines Gegenstands, den der Roboter aufnehmen soll) und gibt die Bildinformation an den Steuerungscomputer weiter (dabei kann es sich um einen separaten Rechner handeln, der nicht für die Steuerung des Roboterarms zuständig ist). Der Computer bearbeitet die erhaltene visuelle Information und bedient sich dabei der Top-Down- oder Bottom-Up-Methode. Dabei entdeckt er gewisse Muster, die für die Ausführung der Arbeit des Roboters wichtig sind. Diese Information wird an den Computer, der den Arm steuert, weitergeleitet, worauf dieser sie intern mit einer bereits gespeicherten Information vergleicht. Der Roboter verfügt nun über Bilder des Objekts und die zur Bearbeitung erforderlichen Werkzeuge, ferner über Anweisungen, was er damit zu tun hat.

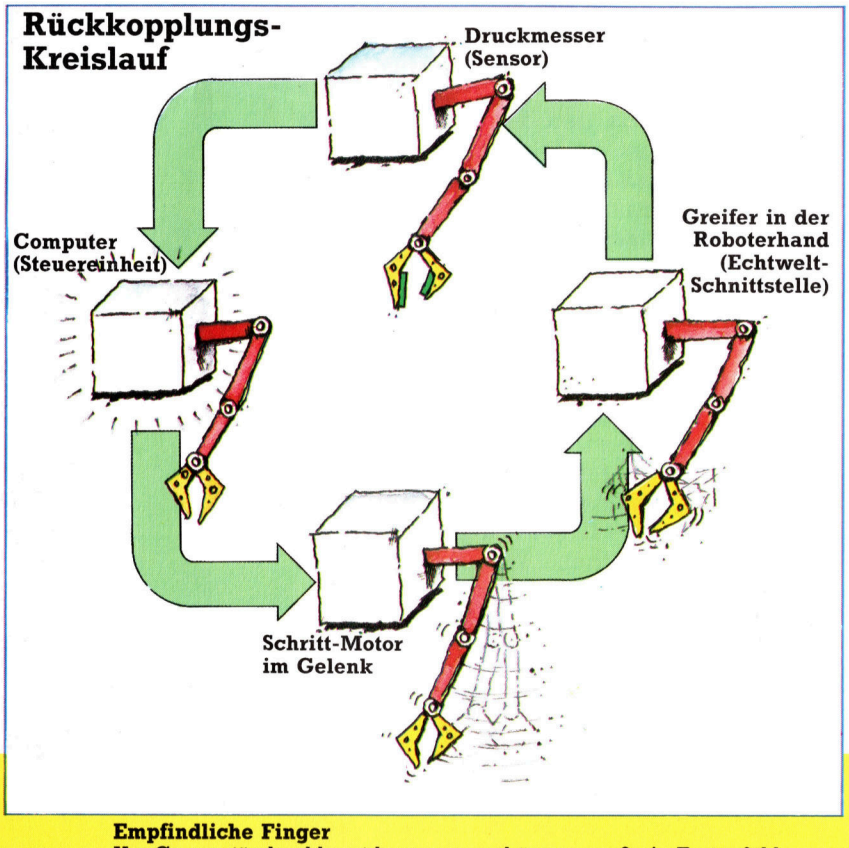
Der große Vorteil dieser Wahrnehmungsmethode liegt darin, daß dem Roboter Teile aus verschiedensten Richtungen zugeliefert werden können und auch völlig unterschiedlich geformte Teile. Er kann die Stücke, die er verarbeiten soll, erkennen und auch den Winkel, in dem sie ihm „gereicht" werden. Ein völlig „blinder" Roboter würde versuchen, jedes Stück auf ein und dieselbe Art zu greifen. Die Ergebnisse könnten katastrophal sein.

Eine andere Möglichkeit, Roboter „empfindsamer" zu machen, ist die Ausstattung mit Sensoren. Stößt ein Roboter gegen ein Objekt, wird diese Information an den Steuercomputer geleitet. Stößt der Roboter gegen eine Wand,

übermittelt der Kraftsensor diese Information, woraufhin eine Modifikation der Handbewegung erfolgt und ein neuerlicher Zusammenstoß verhindert wird.

Sehvermögen und Tastsinn, wenngleich bei den derzeitigen Robotern noch in einem relativ primitiven Stadium, demonstrieren die Wichtigkeit der Rückkopplung. Sollen Roboter die angestrebte Rolle wirklich einmal einnehmen, ist ihre Ausstattung mit Rückkopplungseinheiten unabdingbar. Natürlich ist es schwer, Informationen über die Außenwelt in eine Form zu bringen, die der den Roboter steuernde Computer verarbeiten kann. Eines dieser Probleme ist die Umwandlung der Daten in „Echtzeit"-Daten. Nur wenige Roboter sind mit solchen Sensoren ausgestattet oder haben gar die Fähigkeit, die daraus gewonnenen Erkenntnisse zu interpretieren.

In der Folge, in der wir uns mit „natürlicher Sprache" befaßten, zeigten wir, daß das ständige Verstehen von Sprache das größte Problem ist. Dennoch kann man in gewissem Umfang einen Roboter mit „Spracherkennung" ausstatten (was nichts mit dem „Verstehen" von Sprache zu tun hat). Das bedeutet: Der Roboter wird dahingehend geschult, daß er auf eine begrenzte Anzahl von Sätzen oder Wörtern in irgendeiner Form reagiert.



Empfindliche Finger
Um Gegenstände akkurat bewegen zu können, muß ein Tastgefühl vorhanden sein. Wenn wir einen Gegenstand aufnehmen, schätzen wir das Gewicht des Objekts, um bemessen zu können, wie groß der Kraftaufwand für das Anheben sein muß. Bei diesem Vorgang, der mit unterschiedlichem Erfolg bei Robotern ausprobiert wurde, haben wir es mit einem Rückkopplungs-Zyklus zu tun, in dem Tastsensoren Daten an die Kontrolleinheit geben, die den Griff um den Gegenstand intensivieren oder lockern.

Einige der interessantesten Entwicklungen der Robotik gab es im Zuge der Raumforschung. NASA-Ingenieure haben einen Mars-Rover gebaut, ein Fahrzeug, das auf der Oberfläche des Planeten Mars fahren kann. Dieses Roboterfahrzeug ist mit Sensoren und TV-Kamera ausgestattet, um Informationen über seine Umgebung zu sammeln. Und es verarbeitet die Information selbst. Denn aufgrund der großen Zeitversetzung wäre es unmöglich, auf gefunkte Handlungsanweisungen zu warten. Die Signale brauchten hin und zurück je eine halbe Stunde.

Der Einsatz von Robotern statt Menschen in der Raumforschung spart zudem Geld. Jüngeren amerikanischen Berechnungen zufolge kostet der Transport eines Menschen in den Raum und wieder zurück stündlich rund 40 000 Mark. Die Entwicklungskosten für Raumroboter mögen zwar hoch sein, doch langfristig sind die Roboter preiswerter als ihre menschlichen „Kollegen".

Eine etwas exotische Möglichkeit für die Zukunft ist eine Idee, die John von Neumann Ende der 40er Jahre entwickelte. Neben den theoretischen Grundlagen für die Entwicklung digitaler Computer arbeitete von Neumann an anderen Projekten, so unter anderem auch an der Entwicklung einer Theorie des sich selbst reproduzierenden Automaten. Er sagte voraus, daß sich Maschinen unter Befolgung einer einfachen Gesetzesreihe selbst reproduzieren könnten.

Vier Komponenten

Nach von Neumann benötigt ein solches Robotersystem vier Komponenten. Da ist zunächst eine automatische Fabrik, die Rohstoffe sammelt und diese nach gegebenen Anweisungen in Produkte umsetzt. Zweite Komponente ist ein Duplikator, der diese Anweisungen kopiert. Drittens gehört ein Kontrolleur dazu, der die Anweisungen an den Duplikator zum Kopieren überprüft. Der vierte und letzte Bestandteil ist das Instruktionsschema selbst, das dem System sagt, wie eine völlig neue automatische Fabrik aus den von ihr erzeugten Produkten gebaut werden kann.

Diese Theorie ist nun über 40 Jahre alt und – bisher – Theorie geblieben. Doch NASA-Techniker haben sie aufgegriffen und arbeiten an einem solchen sich selbst vervielfältigenden System für den Mond. Eine solche Raumfabrik würde einen „universalen Konstrukteur" beinhalten, der die von der Produktionseinheit gefertigten Teile dazu benutzt, neue Fabriken zu bauen und natürlich auch einen weiteren universellen Konstrukteur. Die Fabrik würde die Rohstoffe des Mondes verwenden und ohne irdische Kontrolle auskommen. Und schließlich könnte sich die Fabrik ständig selbst mit Roboterarbeitern versorgen.

Königliches Spiel

In der letzten Folge unserer LISP-Serie beschäftigen wir uns mit weiteren Standardfunktionen der Sprache und führen sie am Beispiel von zwei komplexen Problemen vor. Zunächst untersuchen wir jedoch, wie LISP Datenstrukturen darstellt.

LISP verwendet „Pointer" (Zeiger) und bezieht sich so mit einer einheitlichen Schreibweise auf Namen, Zeichen, Zahlen und Listen. Pointer lassen sich in jeder Sprache – auch in BASIC – leicht simulieren.

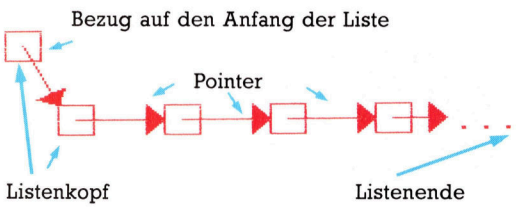
```
POKE 100,55 : POKE 99,100 (fast alle Micros)
```
oder
```
?100 = 55 : ?99 = 100 (Acorn B/Electron)
```

halten in den Speicherstellen 100 und 99 die angegebenen Werte. Der Wert in Adresse 99 läßt sich als Pointer auf Speicherstelle 100 verstehen. Bei dem Befehl

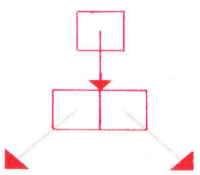
```
PRINT PEEK(PEEK(99))
```
oder
```
PRINT ?(?99)
```

erhalten wir die Antwort 55. Beachten Sie, daß sich die Speicherstellen 99 und 100 auf einigen Micros im ROM befinden und sich daher nicht ändern.

Natürlich können Pointer auf jeden Speicherbereich zeigen. Über 255 liegende Adressen brauchen jedoch zwei Bytes, um den Bereich zwischen 0 und 65535 umfassen zu können. Wenn wir den Wert 55 als Pointer auf eine weitere Speicherstelle ansehen und so weiter, erhalten wir folgende Listenstruktur:

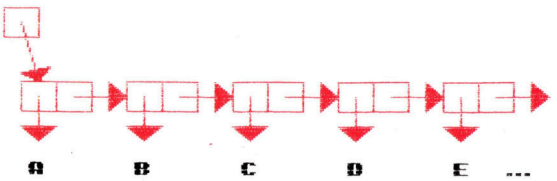


Diese Liste enthält noch keine Informationen. Wenn man jedoch zwei Adressen zusammenfügt, entstehen an jedem Knotenpunkt zwei Pointer:



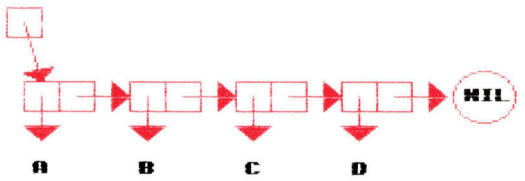
Dieses Verfahren läßt sich nun in ein Listenformat umsetzen:

```
(A B C D E. . .)
```



Nun fehlt nur noch ein Element, das das Listenende anzeigt. Dafür wird der letzte Pointer auf die leere Liste (oder NIL) gesetzt. Hier die Liste mit fester Länge:

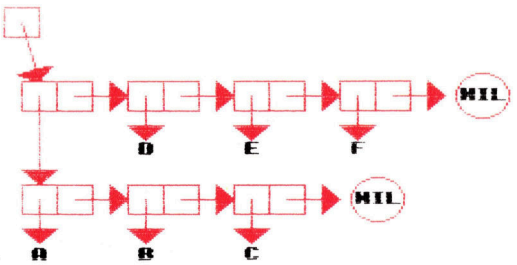
```
(A B C D)
```



Bei einer Liste im Innern einer anderen Liste zeigt nur einer der linken Pointer auf die zweite Liste statt auf ein Atom.

```
((A B C) D E F)
```

wird daher folgendermaßen dargestellt:



An dieser Stelle wird auch deutlich, wie die Funktionen CAR und CDR arbeiten. CAR (der Kopf des Argumentes) ist der linksstehende Pointer des ersten Argumentelements, während CDR (alle Elemente außer dem Kopf) der rechtsstehende Pointer des ersten Elements ist. Der Aufruf

```
CONS((CAR L) (CDR L))
```

liefert daher die ganze Liste.

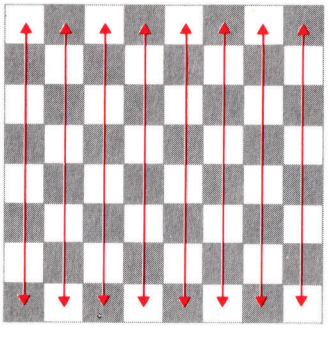
Mit CONS lassen sich Listen anlegen. Die Funktion baut aus seinen beiden Argumenten eine weitere Liste auf und liefert als Ergebnis einen Pointer auf das erste Element der neuen Liste. CONS kann jede Art von Liste aufbauen. Wenn die Zahl der CONS-Aufrufe zu groß wird, kann die Kurzfunktion LIST verwandt werden.

Neue Informationen sollten immer mit CONS an den Kopf einer Liste angefügt werden, da beim Anfügen an das Listenende zuerst die ursprüngliche Liste kopiert werden muß, damit der Endpointer auf das neue Element zeigt.

Einige LISP-Versionen führen diese Aufgabe mit APPEND automatisch aus. Diese Funktion ist jedoch nicht so schnell wie CONS. Aus Geschwindigkeitsgründen wird manchmal nur der Pointer der ursprünglichen Liste umgesetzt und keine Kopie angelegt.

Zum Abschluß dieser Serie stellen wir die Lösung für ein bekanntes Problem vor: Auf einem Schachbrett sollen acht Königinnen aufgestellt werden, ohne sich gegenseitig zu bedrohen. Offensichtlich muß in jeder Reihe und Spalte genau eine Königin stehen. Die Schwierigkeit liegt darin, die Reihen und Spalten so anzuordnen, daß sich die Reichweiten der Figuren in der Diagonale nicht überschneiden.

Jede Spalte des Schachbretts enthält eine Königin, die gemäß ihrem Platz von 1 bis 8 in einer Liste namens KOENIGINNEN gespeichert sind. Die Spalte jeder Königin (der Wert von KOENIGINNEN) ist eine Variable und wird von LISP berechnet.



KOENIGINNEN (K1 K2 K3 K4 K5 K6 K7 K8)

Beim Programmanfang ist die Liste KOENIGINNEN noch leer. Bei jedem Schritt wird eine neue Königin in Reihe 1 am Anfang der Liste eingesetzt. Die Position dieser Königin (und möglicherweise auch die der anderen) wird mit der Funktion AENDERN so lange verändert, bis alle Königinnen auf sicheren Positionen stehen. Die neue Liste wird dann wiederum zu KOENIGINNEN, an deren Anfang eine weitere Königin angefügt wird, und so weiter.

Dieser Vorgang setzt sich fort, bis die Liste acht Königinnen enthält. An diesem Punkt wird das Ergebnis mit PR_SCHACHBRETT gedruckt. Die Anfangsfunktion nennen wir LOESUNG. Sie sieht folgendermaßen aus:

```
(DEFUN LOESUNG ()
 (SETQ KOENIGINNEN ())
 (LOOP (UNTIL EQ (LAENGE (KOENI-
   GINNEN) 8)) (SETQ KOENIGINNEN (ALTER
   (CONS 1 KOENIGINNEN))))
 (PR_SCHACHBRETT ' (LT LS LL LK RK RL
   RS RT) KOENIGINNEN))
```

Hier ist leicht zu erkennen, wie LOESUNG anfangs die leere Liste KOENIGINNEN anlegt. Die Funktion LOOP setzt KOENIGINNEN auf den neuen Wert der mit AENDERN aktualisierten Liste. Dabei steht eine Königin in Reihe 1 gefolgt von der zuvor angelegten Liste. Die Schleife LOOP wird so lange ausgeführt, bis (UNTIL) die LAENGE von KOENIGINNEN dem Wert 8 gleicht (EQ). Danach wird die fertige Liste gedruckt.

Das Programm enthält drei neue Funktionen. LAENGE liefert die Anzahl der Elemente eines Arguments und ist folgendermaßen definiert:

```
(DEFUN LAENGE (L (N))
 (SETQ N 0)
 (LOOP (WHILE L N)
   (SETQ N (ADD1 N))
   (SETQ L (CDR L)) ))
```

Mit dem in Klammern gestellten Argument N stellt Acornsoft-LISP mögliche Argumente dar. In unserem Fall wird N nicht an die Funktion übergeben – es ist nur eine lokale Variable.

Die Funktion PR_SCHACHBRETT kombiniert beide Argumentenlisten zu einem Ausdruck:

```
(DEFUN PR_SCHACHBRETT (ZEILEN
 SPALTEN) (COND ((NULL ZEILEN) T)
   (T (PRINTC (CAR ZEILEN) (CAR SPALTEN))
    (PR_SCHACHBRETT (CDR ZEILEN) (CDR
      SPALTEN)) )))
```

Da beide Listen die gleiche Größe haben, wird die Funktion beendet, wenn die erste Bedingung eintritt (eine Liste ist leer). Falls nicht, wird das erste Element jeder Liste gedruckt (mit Return), während sich die Funktion für den Rest der Liste immer wieder selbst aufruft.

Die dritte Funktion AENDERN schließlich erledigt die Hauptarbeit:

```
(DEFUN AENDERN (KOENIGINNEN)
  (COND ((EQ (CAR KOENIGINNEN) 9)
    (AENDERN (CONS (ADD1
      (CADR KOENIGINNEN))
      (CDDR KOENIGINNEN) )))
    ((SICHER KOENIGINNEN) KOENI-
      GINNEN)
    (T (AENDERN (CONS (ADD1
      (CAR KOENIGINNEN))
      (CDR KOENIGINNEN) )))))
```

AENDERN überprüft, ob die zuletzt eingesetzte Königin vom Spielbrett „gefallen" ist (Zeile 9). Ist dies der Fall, müssen die Positionen der anderen Königinnen verändert werden.

Da die „gefallene" Königin alle Zeilen von 1 bis 8 überprüft hat (die dritte Bedingung von AENDERN), gibt es keine Möglichkeit, sie bei den augenblicklichen Positionen der anderen Königinnen auf dem Brett aufzustellen. AENDERN ruft sich daher für den Rest der Liste (alle Königinnen außer der ersten) selbst auf.

Dabei inkrementiert die Funktion die Zeile der letzten Königin. Dieser Vorgang wird „Backtracking" genannt. Er ist bei dieser Art von Computeralgorithmen weit verbreitet. Ist die zweite Bedingung TRUE, dann stehen alle Königinnen auf sicheren Positionen und die Liste kann zurückgegeben werden. Ist dies

nicht der Fall, wird die Position der zuletzt eingesetzten Königin inkrementiert und ein weiterer Versuch unternommen.

Nun fehlt nur noch die Funktion SICHER, die überprüft, ob alle Königinnen auf unangreifbaren Positionen stehen. Da in der Liste KOENIGINNEN die Spalten bereits festgelegt sind und zwei Königinnen nie die gleiche Position einnehmen können, brauchen nur die Zeilen und Diagonalen überprüft zu werden. Die Funktion KEINGARDEZ ergibt TRUE, wenn keine Königin eine andere bedroht. SICHER wird folgendermaßen definiert:

```
(DEFUN SICHER (KOENIGINNEN)
  (COND ((NULL KOENIGINNEN) T)
    (T (AND (KEINGARDEZ
      (CAR KOENIGINNEN)
      (CDR KOENIGINNEN) 1)
      (SICHER (CDR KOENIGINNEN)) ))))
```

Dieser Ablauf läßt sich in normalen Worten leichter verstehen:

„Die erste Königin der Liste KOENIGINNEN bedroht keine andere Königin der Liste UND alle anderen Königinnen der restlichen Liste sind sicher."

Nun muß nur noch die Funktion KEINGARDEZ definiert werden. Sie soll TRUE ergeben, wenn es in den Zeilen und Spalten keine Kreuzungspunkte gibt:

```
(DEFUN KEINGARDEZ (NEW REST SPALTE)
  (COND ((NULL REST) T)
    (T (AND (NOT (EQ NEW (CAR REST)))
      (NOT (EQ SPALTE (ABS (DIFFERENCE
        NEW (CAR REST)))))
      (KEINGARDEZ NEW (CDR REST)
        (ADD1 SPALTE)) ))))
```

Hier ist NEW die zuletzt eingesetzte Königin und REST die Liste aller anderen Königinnen des Brettes. Angenommen, andere Königinnen sind auf dem Brett vorhanden (NULL REST ist FALSE), so ergibt die AND-Funktion TRUE, wenn:

1) die augenblickliche Zeile der Königin nicht die gleiche ist wie die Zeile der ersten Königin der restlichen Liste;

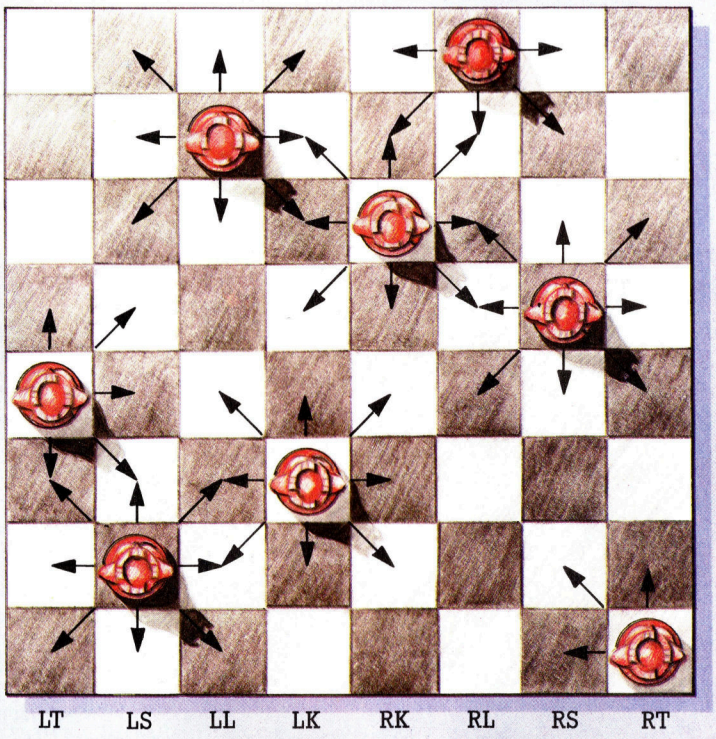
2) die Königinnen nicht auf der gleichen Diagonalen liegen. Bei der Überprüfung wird sichergestellt, daß die absolute Zeilendifferenz nicht die gleiche ist wie die Spaltendifferenz. SPALTE enthält dabei die Zahl der Spalten, die zwischen der augenblicklichen Königin und der nächsten Königin der Liste liegen (anfangs 1). ABS wurde schon in einer früheren Folge definiert:

```
(DEFUN ABS (ZAHL)
  (COND ((MINUSP ZAHL) (MINUS ZAHL))
    (T ZAHL) ))
```

3) Punkt 1 und 2 für die augenblickliche Königin und auch die nächste Königin der Liste zutreffen. Dies wird überprüft, indem die Funktion sich selbst aufruft, dabei aber nur die restlichen Königinnen mit Ausnahme des ersten Elements anspricht und die Spaltendifferenz inkrementiert.

Jetzt sind alle Definitionen vorhanden. Das Bild zeigt die Lösung.

Wahrhaft königlich
Das „Problem der acht Königinnen" liefert ein perfektes Beispiel für Rätsel, die sich mit den Listen und recursiven Funktionen von LISP schnell lösen lassen. Probieren Sie es selbst einmal, bevor Sie sich die nebenstehende Lösung ansehen. Setzen Sie acht Königinnen derart auf ein Schachbrett, daß sie sich nicht gegenseitig bedrohen. Sie brauchen Schach nicht zu kennen, sondern müssen nur wissen, daß Königinnen sich in gerader Linie beliebig weit diagonal, vertikal und horizontal bewegen können und dabei jede im Weg liegenden Figur schlagen.

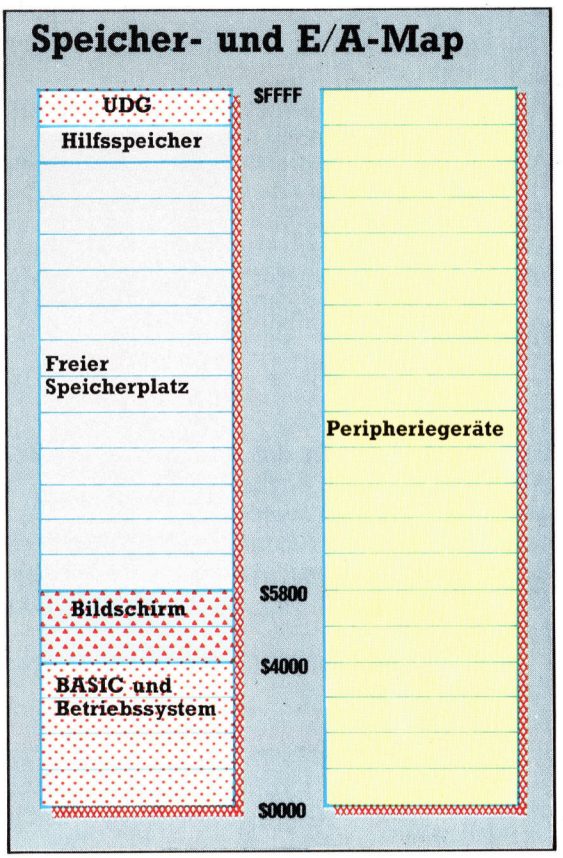


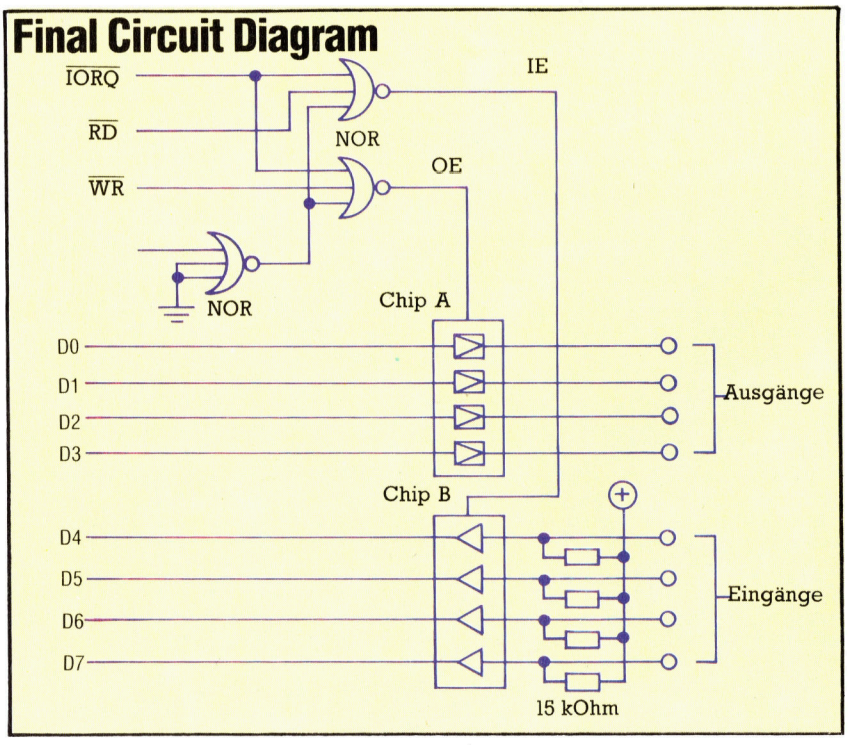
Umbaumaßnahmen

Jetzt können auch Spectrum-Besitzer an unserem Roboter-Projekt teilnehmen: Dazu wird der Spectrum mit einem Interface ausgerüstet.

Der Roboter wird über eine der 65535 vorhandenen Ein- und Ausgabepositionen der E/A-Map mit dem Spectrum verbunden. Mit den Befehlen IN und OUT zum Schreiben und Lesen von Daten können diese Positionen angesprochen werden.

Durch die Verbindung von der IORQ-Leitung mit den E/A-Steuerleitungen sowie dem Anschluß von Adreßbit 5 über einige NOR-Gatter können wir zwei Ausgangssignale erzeugen. IE und OE aktivieren die vier Buffer-Schaltungen für die Ein- und Ausgabe. Dadurch können die Steuersignale vom Computer zu den Schrittmotoren im Roboter gelangen. Umgekehrt werden auch die Sensorsignale für die Programmsteuerung weitergeleitet. Bei einem High auf $\overline{WR}$ kann eingegeben werden, die Ausgabe wird durch High auf $\overline{RD}$ aufgerufen.





Acorn B und Commodore 64 haben einen eingebauten User Port zum Anschluß unterschiedlicher Peripheriegeräte. In unserem Selbstbau-Kurs haben wir bereits verschiedene Geräte darüber gesteuert. Leider fehlt eine solche Schnittstelle beim Sinclair Spectrum. Neben dem Cassettenrecorder-Anschluß hat dieser Rechner kein weiteres Interface.

Der Spectrum ist stattdessen mit einem Erweiterungsport ausgerüstet, über den alle internen Anschlüsse – Datenbus, Adreßbus, Steuerleitungen, Stromversorgung und noch einige andere Signale – geleitet werden. Wir können also eine User-Port-Imitation zum Aufstecken an den Erweiterungsport nachrüsten.

User Ports basieren auf einem VIA (Versatile Interface Adaptor) genannten Chip, der eine Vielfalt unterschiedlicher Aufgaben zu bewältigen hat. Der VIA stellt acht Ein/Ausgangsleitungen zur Verfügung, von denen jede softwaregesteuert auf Ein- und Ausgabe geschaltet werden kann. Außerdem verfügt der VIA noch über diverse Steuerleitungen für den „Handshaking"-Betrieb sowie über eine interne Uhr. Unser Spectrum-Interface ist nicht ganz so kompliziert, für die Robotersteuerung reicht es aber aus. Die vier Eingangs- und vier Ausgangsleitungen sind fest definiert. Sie lassen sich somit zwar nicht mehr über das Programm umdefinieren, ersparen uns aber den Nachbau der Kombination aus Datenrichtungs- und Datenregister im VIA.

Bei den meisten Rechnern mit 6502-Prozessor sind User Port und andere Schnittstellen mit dem Rechner ähnlich wie ein Byte im Speicher gekoppelt – sie bilden also einen Teil der Memory-Map. Der Z80-Prozessor des Spectrum arbeitet anders, bei ihm stehen die Adressen für Ein- und Ausgabe in einer parallelen zweiten Memory-Map. Sie kann vom BASIC aus durch IN und OUT angesprochen werden, ähnlich wie PEEK und POKE den Speicherinhalt ansprechen.

Kontrollbit

Der Z80 unterscheidet Hauptspeicher- und Ein/Ausgabe-Adressen durch ein separates Adreßbit. Neben den 16 Bits zur Adressierung eines 64-K-Hauptspeichers kontrolliert dieses Bit auch, ob gerade die Speicher- oder die Ein/Ausgabeadresse angesprochen wird. Für dieses Extra-Bit stellt der Prozessor zwei Steuerleitungen bereit, $\overline{IORQ}$ (Input/Output ReQuest) und $\overline{MREQ}$ (Memory ReQuest). Zum

	A10	A8	RFSH	M1	-12v	+12v	WAIT	-5v	WR	RD	IORQ	MREQ	HALT	NMI	INT	D4	D3	D5	D6	D2	D1	D0			D7	A13	A15
A11	A9	BUSACK	ROMCS	A4	A5	A6	A7	RESET	BUSRQ	U	V	Y	VIDEO	0v	IOROGE	A3	A2	A1	A0	CK	0V	0V		9V	5V	A12	A14

Adressieren des Speichers geht die MREQ-Leitung auf Low, wenn dagegen IORQ Low wird, soll eine Ein- bzw. Ausgabe zur Prozessor-Peripherie erfolgen. Alle 16 Adreßleitungen und auch die beiden Steuerleitungen sind auf dem Erweiterungsanschluß des Spectrum verfügbar. Unser Interface muß also auf eine bestimmte Adresse und den Zustand der IORQ-Leitung reagieren.

Belegte E/A-Ports

Einige der 65535 möglichen E/A-Ports sind beim Spectrum bereits mit der Tastaturabfrage, dem Cassetteninterface und dem Lautsprecher belegt. Die E/A-Ports mit den Adreßleitungen 0 bis 4 werden vom Spectrum genutzt, wir verwenden ausschließlich Leitung 5. Dadurch „erscheint" das Interface zwar mehrfach in der Ein/Ausgabe-Map (an jeder Adresse, die Bit 5 verwendet), die Elektronik wird dadurch aber sehr viel unkomplizierter.

Weil Daten sowohl ein- als auch ausgegeben werden müssen, brauchen wir neben den Adreß- und Steuerleitungen auch die Schreib- und Leseleitungen des Z80 (Read=RD, Write=WR). Das Interface muß auf ein Low der IORQ- und der A5-Leitung zusammen mit Low auf der RD- oder der WR-Leitung reagieren. In diesem Fall schaltet das Interface den Datenbus auf die Steuerleitungen des Roboters. Rechts haben wir die dazugehörige Wahrheitstabelle abgedruckt.

IE und OE sind die Einschaltsignale für Ein- und Ausgabe des Interface. Natürlich kann der Prozessor nicht einen Speicherplatz gleichzeitig auslesen und beschreiben, es werden also nie alle Leitungen einschließlich RD und WR gleichzeitig Low. Die Wahrheitstabelle läßt sich sehr leicht in die dargestellte Schaltung umsetzen.

Sind die IORQ-, RD- und A5-Leitungen alle Low (etwa, wenn der Prozessor den Ein/Ausgabe-Port 31 lesen will), wird die Leitung IE im Interface High. Dann werden die vier Eingangsleitungen über den Buffer-Chip A mit den vier oberen Datenbus-Bits des Computers verbunden. Bei einem Low auf IORQ, A5 und WR wird OE High – die vier Ausgabeleitungen des Interface sind dann mit den vier unteren Datenbus-Bits gekoppelt. Arbeitet der Prozessor danach mit anderen Adressen – OE wird wieder Low – speichert Chip B das ausgegebene Bit-Muster bis zum nächsten Prozessor-Zugriff (Latching).

Bei allen anderen Signalkombinationen sind die Interface-Anschlüsse vom Datenbus isoliert und beeinflussen den Computer nicht. Die gesamte Schaltung wird mit drei ICs aufgebaut, eines mit vier OR-Gattern (nur drei sind benutzt), einem Vierfach-Buffer-IC (Chip A) und einem Vierfach-Latch (Chip B).

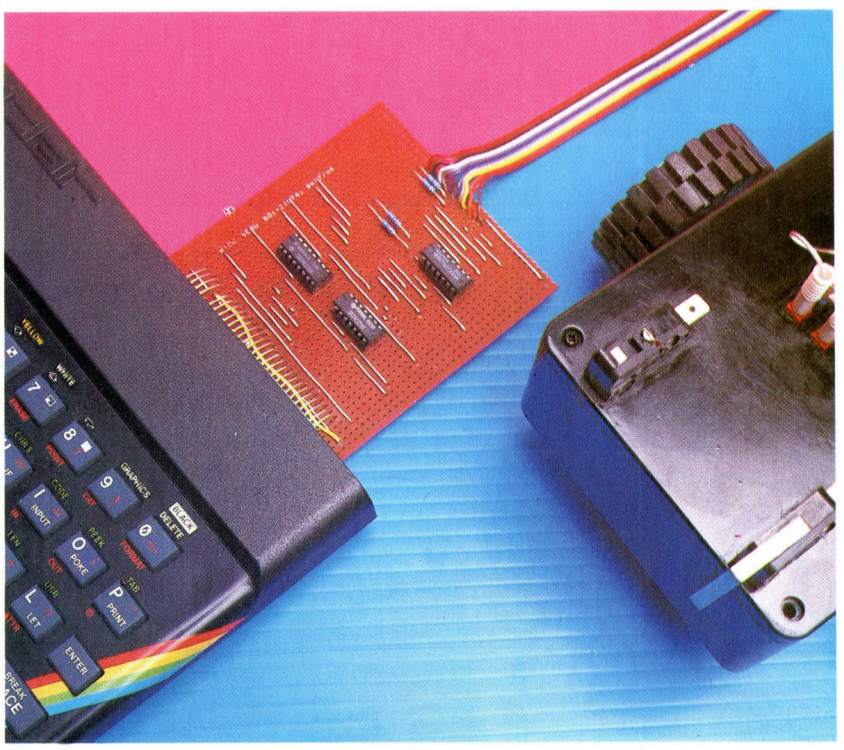
Wahrheitstabelle für Spectrum-Interface

IORG	WR	RD	A5	IE	OE
0	0	0	0	—	—
0	0	0	1	—	—
0	0	1	0	0	1
0	0	1	1	0	0
0	1	0	0	1	0
0	1	0	1	0	0
0	1	1	0	0	0
0	1	1	1	0	0
1	0	0	0	—	—
1	0	0	1	—	—
1	0	1	0	0	0
1	0	1	1	0	0
1	1	0	0	0	0
1	1	0	1	0	0
1	1	1	0	0	0
1	1	1	1	0	0

Die Wertetabelle stellt den Zustand der Steuerleitungen für alle Kombinationen der vier Eingangssignale dar. Natürlich sind in Wirklichkeit die Leitungen WR und RD nie gleichzeitig Null. Eingabe ist gefordert, wenn IORQ, RD und A5 alle auf Null liegen. IE wird durch Einspeisen aller drei Signale in ein NOR-Gatter hergestellt. Ähnlich bei der Datenausgabe, die mit Low von IORQ, WR und A5 eingeleitet wird. Ein zweites NOR-Gatter mit diesen drei Leitungen erzeugt OE.

Über den Erweiterungsport auf der Rückseite des Spectrum können Peripheriegeräte mit der Rechnerelektronik gekoppelt werden. Es stehen alle 16 Adreßleitungen und die acht Datenleitungen zur Verfügung. Außerdem sind zusätzliche Prozessoranschlüsse herausgeführt, etwa RD, WR und IORQ. Mit einer Hilfsschaltung am Erweiterungsanschluß kann auch der Sinclair Spectrum unseren Roboter steuern.

Der Acorn B und der Commodore 64 verfügen über spezielle Ein- und Ausgabechips. Dadurch läßt sich unser Selbstbau-Roboter direkt mit den beiden Rechnern koppeln. Damit auch der Sinclair Spectrum den Roboter steuern kann, müssen wir eine Zusatzplatine bauen, die sich auf den Erweiterungsport des Spectrum stecken läßt.



Schlußszene

Jetzt geben wir dem Minenspiel für den Acorn B den letzten Schliff. Dabei betrachten wir speziell den Teletext-Modus, den wir in der Spielende-Routine verwenden.

Der Grafikmodus 7 des Acorn B, auch bekannt als Teletext-Modus, hat im Gegensatz zu den anderen Modi einige Besonderheiten. Diese dienen zur Darstellung von externen Rechnern übertragener Informationen, auf die über eine Telefonleitung Zugriff genommen werden kann. Diese zusätzlichen Grafikmöglichkeiten produzieren mit wenigen einfachen Anweisungen attraktive Textdarstellungen, so daß dieser Modus ideal für die „Spielende"-Darstellung geeignet ist.

Durch Verwendung von CHR$-Kontroll-Codes in PRINT-Anweisungen können wir den Text und die Hintergrundfarben kontrollieren, um so einen blinkenden Text mit doppelt hohen Zeichen zu produzieren. Zur Positionierung des Textes kann wie gewohnt die TAB-Funktion verwendet werden. Insgesamt stehen sieben Farben zur Verfügung, die mit den folgenden Codes gewählt werden können:

129	Rot
130	Grün
131	Gelb
132	Blau
133	Magenta
134	Cyan
135	Weiß

Wird der Modus 7 eingesetzt, so erscheint der Text in Weiß auf schwarzem Hintergrund. Die Textfarbe kann jederzeit durch einen Kontrollcode geändert werden. So werden beispielsweise die drei folgenden Wörter in den Farben Rot, Weiß und Grün dargestellt:

```
PRINT CHR$(129)"SPIEL";CHR$(135);"OHNE";
CHR$(132);"GRENZEN"
```

Dabei ist jedoch zu beachten, daß bei der Darstellung einer neuen Zeile wieder die Ursprungsfarbe (Weiß) verwendet wird. Somit müssen für jede neue Zeile Kontrollcodes angegeben werden, selbst wenn man den Text in der gerade benutzten Farbe darstellen will.

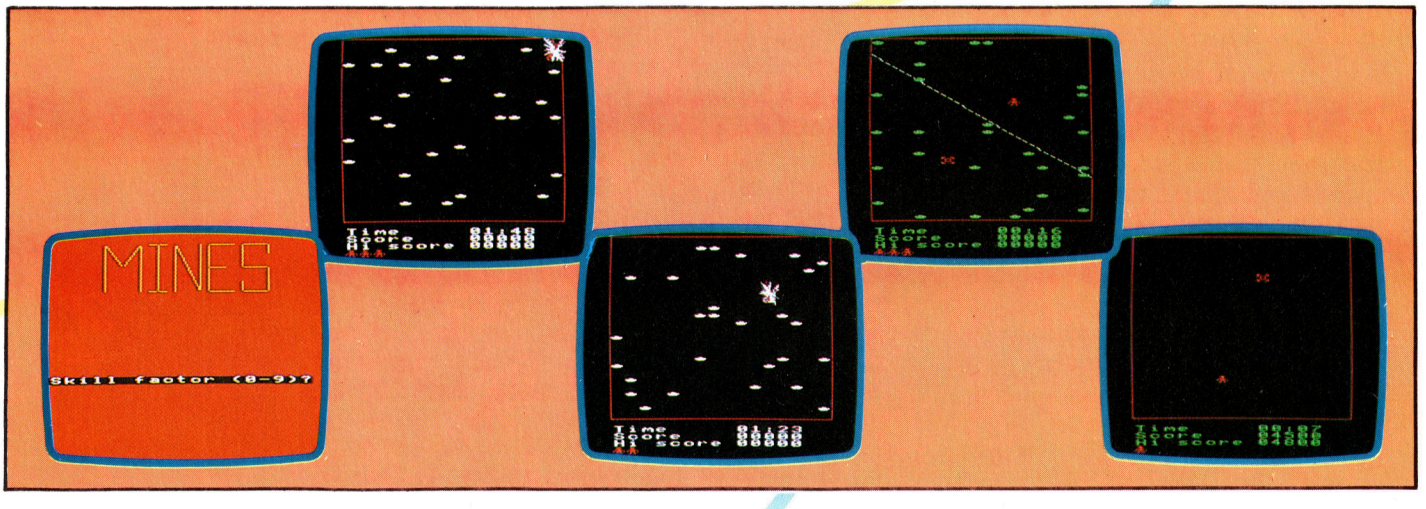
Außer der Textfarbe kann auch die Hintergrundfarbe geändert werden. Dies geschieht mit CHR$(157), gefolgt vom gewünschten Farbwert. Um zum Beispiel blaue Buchstaben auf weißem Hintergrund darzustellen, werden folgende Kontrollcodes verwendet:

```
PRINT CHR$(132);CHR$(157);CHR$(135);"BEI
DER MARINE"
```

Der erste Wert gibt die Textfarbe an, der zweite und dritte die Hintergrundfarbe. Sowohl Text als auch Hintergrund blinken, wenn direkt vor dem Farbcode CHR$(136) verwendet wird. Mit CHR$(137) wird dieser Effekt wieder abgeschaltet. Im folgenden Beispiel blinkt die blaue Schrift:

```
PRINT CHR$(136);CHR$(132);CHR$(157);
CHR$(135);"BEI DER MARINE"
```

Die eindrucksvollste Möglichkeit von Modus 7 ist, Zeichen in doppelter Höhe darzustellen. Dazu wird CHR$(141) verwendet, wobei jedoch dieselbe Zeile zweimal gePRINTet werden muß. Sämtliche anderen Effekte können auch mit den doppelt hohen Zeichen verwendet werden. Um beispielsweise doppelt hohe, blinkende, blaue Zeichen auf weißem Hintergrund zu produzieren, geben Sie nunmehr folgende Zeilen ein:



Diese Bilder von verschiedenen Programmläufen des Spieles zeigen die Minen, den Assistenten, „Heckenschützen"-Feuer, eine Explosion und die Punktzahl sowie den Titelbildschirm.

```
10 MODE 7
20 PRINT CHR$(141);CHR$(136);CHR$(132);
   CHR$(157);CHR$(135);"BEI DER MARINE"
30 PRINT CHR$(141);CHR$(136);CHR$(132);
   CHR$(157);CHR$(135);"BEI DER MARINE"
```

Diese Kontrollcodes belegen viel Programmspeicher und Zeit zur Eingabe. Eine alternative Methode ist, mehrere Codes in einem einzelnen String zu verbinden, der dann in den PRINT-Anweisungen verwendet wird. Braucht man viele Texte in roter Schrift auf gelbem Hintergrund, so kann man einen String (rot$) definieren, der dann mit den entsprechenden PRINT-Anweisungen benutzt wird:

```
10 MODE 7
20 rot$=CHR$(129)+CHR$(157)+CHR$(131)
30 PRINTrot$,"SPIELENDE"
40 PRINTrot$,"IHRE PUNKTE"
```

„GAME OVER"-Prozedur

Diese Effekte sollen nun in unsere Spielende-Prozedur integriert werden:

```
2110DEF PROCend_game
2120 IF score$>hi_score$ THEN hi_score$=score$
2130red$=CHR$(129)+CHR$(157)+CHR$(131)
2140game$="G A M E  O V E R "
2150PRINTTAB(0,5)red$;CHR$(141);CHR$(136);TAB(12);game$
2160PRINTred$;CHR$(141);CHR$(136);TAB(12);game$
2170PRINT:PRINTred$;"Your Score";TAB(30);score$
2180PRINT:PRINTred$;"Hi score";TAB(30);hi_score$
2190PRINT:PRINTred$;"Time";TAB(30);time$
2200blue$=CHR$(132)+CHR$(157)+CHR$(134)
2210go$="A N O T H E R  G O    Y / N ?"
2220PRINT:PRINT
2230PRINTblue$;CHR$(141);CHR$(136);TAB(5);go$
2240PRINTblue$;CHR$(141);CHR$(136);TAB(5);go$
2250REM ** REPLY ? **
2260*FX 15,1
2270answer$=INKEY$(0)
2280IF GET$="N" THEN finish_flag=1
2290ENDPROC
```

In Zeile 2120 wird überprüft, ob die erreichte Punktzahl eine neue Bestpunktzahl ist.

Die Meldung „GAME OVER" wird dann in doppelt hohen, rot blinkenden Zeichen auf gelbem Hintergrund dargestellt (Zeilen 2130 bis 2160) sowie Angaben über Punktzahl und Zeit (Zeilen 2170 bis 2190). Danach wird der Spieler gefragt, ob er noch einmal spielen will. Ist die Antwort N, wird eine Variable (finish flag) auf Eins gesetzt.

Beachten Sie, daß Modus 7 in dieser Routine noch nicht aktiviert wurde. Dies liegt daran, daß der Acorn B innerhalb einer Prozedur keine Änderung des Modus gestattet. Versucht man es trotzdem, erhält man eine BAD-MODE-Fehlermeldung. Statt dessen muß Modus 7 in einem kurzen Programm gesetzt werden, das dann die Prozedur aufruft. Die folgenden Zeilen vervollständigen das Programm. Beachten Sie, daß sich das aufrufende Programm in einer REPEAT . . .UNTIL-Schleife befindet, die so lange ausgeführt wird, bis das finish flag auf Eins gesetzt ist.

```
1100REPEAT

1200MODE7
1210REM ** TURN OFF CURSOR **
1220VDU23,1,0;0;0;0;
1230PROCend_game
1240UNTIL finish_flag=1
1250CLS
1260END
```

Besitzer eines Electron konnten mit unseren Abhandlungen über Modus 7 sicher nicht viel anfangen, da der Electron nicht über diesen Modus verfügt. Daher haben wir alternativ eine andere Prozedur entwickelt, die Modus 5 verwendet. Entfernen Sie Zeile 1200 des aufrufenden Programms und geben Sie die folgende Prozedur ein:

```
>L.2100,2300
 2100 DEF PROCend_game
 2110 IF score$>hi_score$ THEN hi_score$=score$
 2120 REM ENSURE BACKGROUND YELLOW
 2130 VDU19,130,3,0,0,0
 2140 GCOL0,130:CLG:REM COLOUR SCREEN
 2150 COLOUR1:COLOUR130:REM SET TEXT COLOURS
 2160 game$="G A M E  O V E R "
 2170 PRINTTAB(2,4);game$
 2180 COLOUR0
 2190 PRINTTAB(0,8);"Your Score";TAB(15);score$
 2200 PRINT:PRINT"Hi score";TAB(15);hi_score$
 2210 PRINT:PRINT"Time";TAB(15);time$
 2220 go$="ANOTHER GO Y/N ?"
 2230 REM CHANGE COL3 TO FLASH YELL/BLUE
 2240 VDU19,3,11,0,0,0
 2250 COLOUR3
 2260 PRINT:PRINT
 2265 PRINTTAB(2)go$
 2270 REM ** REPLY ? **
 2275 *FX 15,1
 2280 answer$=INKEY$(0)
 2285 IF GET$="N" THEN finish_flag=1
 2290 VDU 20:REM RESET DEFAULT COLOURS
 2300 ENDPROC
```

Das Minenfeldspiel (vollständig)

```
1000REM ********************
1010REM *                  **
1020REM *    MINES         **
1030REM *                  **
1040REM ********************
1050:
1060hi_score$="00000"
1070finish_flag=0
1080:
1090REM **** MAIN PROGRAM ****
1100REPEAT
1110MODE5
1120REM ** TURN OFF CURSOR **
1130VDU23;8202;0;0;0;
1140PROCtitle_page
1150CLS
1160PROCsetup
1170:
1180PROCloop
1190:
1200MODE7
1210REM ** TURN OFF CURSOR **
1220VDU23,1,0;0;0;0;
1230PROCend_game
1240UNTIL finish_flag=1
1250CLS
1260END
1270:
1280:
1290REM **** DEFINE PROCEDURES ****
1300DEF PROCtitle_page
1310GCOL 0,129
1320CLG
1330GCOL 3,3
1340PROCmusic
1350Y=100:X=0
1360REPEAT
1370X=X+20:Y=Y+50
1380FOR I=1 TO 2
1390PROCmines
1400NEXT I
1410UNTIL Y>700
1420:
1430PROCmines
1440PRINTTAB(0,20)"Skill factor (0-9)?"
1450PROCmusic
1460REPEAT
1470skill=GET-48
1480UNTIL skill>-1 AND skill<10
1490ENDPROC
1500:
1510DEF PROCmines
1520PLOT4,X,Y
1530REM ** LETTER M **
1540PLOT1,0,200
1550PLOT1,80,-100
1560PLOT1,80,100
1570PLOT1,0,-200
1580REM ** LETTER I **
1590PLOT0,40,0
1600PLOT1,80,0
1610PLOT0,-40,0
1620PLOT1,0,200
1630PLOT0,-40,0
1640PLOT1,80,0
1650REM ** LETTER N **
1660PLOT0,40,-200
1670PLOT1,0,200
1680PLOT1,120,-200
1690PLOT1,0,200
1700REM ** LETTER E **
1710PLOT0,160,0
1720PLOT1,-120,0
1730PLOT1,0,-200
```

Das Listing zeigt die Syntax der Variablen- und Prozedurennamen (wie zum Beispiel hi_score$ und PROCend_game), bei denen das Unterstreichungs-Zeichen „_“ verwendet wird. Acorn-B-Programmierer werden mit seiner Verwendung als Leerstellenzeichen vertraut sein. Es darf nicht mit dem Bindestrich verwechselt werden.

```
1740PLOT1,120,0
1750PLOT0,-40,100
1760PLOT1,-80,0
1770REM ** LETTER S **
1780PLOT0,280,60
1790PLOT1,0,40
1800PLOT1,-120,0
1810PLOT1,0,-100
1820PLOT1,120,0
1830PLOT1,0,-100
1840PLOT1,-120,0
1850PLOT1,0,40
1860ENDPROC
1870:
1880DEF PROCsetup
1890COLOUR 2
1900end_flag=0
1910PROCinitialise_variables
1920PROCdefine_characters
1930factor=skill*3+30
1940PROClay_mines(factor)
1950PROCdraw_border
1960PROCset_time
1970PROCset_score
1980PROCset_men
1990PROCposition_chars
2000ENDPROC
2010:
2020DEF PROCloop
2030REPEAT
2040PROCupdate_time
2050PROCtest_keyboard
2060rand=RND(50-skill)
2070IF rand=1 THEN PROCsnipe
2080 UNTIL TIME>12099 OR end_flag=1
2090ENDPROC
2100:
2110DEF PROCend_game
2120 IF score$>hi_score$ THEN hi_score$=score$
2130red$=CHR$(129)+CHR$(157)+CHR$(131)
2140game$="G A M E  O V E R "
2150PRINTTAB(0,5)red$;CHR$(141);CHR$(136);TAB(12);game$
2160PRINTred$;CHR$(141);CHR$(136);TAB(12);game$
2170PRINT:PRINTred$;"Your Score";TAB(30);score$
2180PRINT:PRINTred$;"Hi score";TAB(30);hi_score$
2190PRINT:PRINTred$;"Time";TAB(30);time$
2200blue$=CHR$(132)+CHR$(157)+CHR$(134)
2210go$="A N O T H E R  G O    Y / N ?"
2220PRINT:PRINT
2230PRINTblue$;CHR$(141);CHR$(136);TAB(5);go$
2240PRINTblue$;CHR$(141);CHR$(136);TAB(5);go$
2250REM ** REPLY ? **
2260*FX 15,1
2270answer$=INKEY$(0)
2280IF GET$="N" THEN finish_flag=1
2290ENDPROC
2300:
2310REM **** LEVEL 2 PROCEDURES ****
2320DEF PROCinitialise_variables
2330xdet=2:ydet=25:xman=17:yman=1
2340xstart=120:xfinish=1144
2350score$="000000"
2360ENDPROC
2370:
2380DEF PROCdefine_characters
2390REM ** MINES **
2400VDU23,224,0,0,56,254,254,124,0,0
2410REM ** MINE DETECTOR **
2420VDU23,225,231,195,129,0,36,36,189,195,231
2430REM ** ASSISTANT **
2440VDU23,226,56,56,56,16,124,186,170,40,108
2450ENDPROC
2460:
2470DEF PROCdraw_border
2480GCOL 0,1
2490MOVE 120,188
2500DRAW 120,992
2510DRAW 1152,992
2520DRAW 1152,188
2530DRAW 120,188
2540ENDPROC
2550:
2560DEF PROClay_mines(number_mines)
2570REM ** CHANGE COLOUR 2 TO GREEN **
2580VDU19,2,2,0,0,0
2590FOR I=1 TO number_mines
2600PRINTTAB(RND(16)+1,RND(25));CHR$(224)
2610NEXT I
2620ENDPROC
2630:
2640DEF PROCset_time
2650PRINTTAB(2,27)"Time      02:00"
2660TIME=0
2670ENDPROC
2680:
2690DEF PROCset_men
2700men$=CHR$(226)+CHR$(226)+CHR$(226)
2710count=1
2720COLOUR 1
2730PRINTTAB(2,30);men$
2740COLOUR 2
2750ENDPROC
2760:
2770DEF PROCset_score
2780score=0:score$="00000"
2790PRINTTAB(2,28)"Score     00000"
2800PRINTTAB(2,29)"Hi score ";hi_score$
2810ENDPROC
2820:
2830DEF PROCposition_chars
2840COLOUR 1
2850PRINTTAB(xdet,ydet);CHR$(225)
2860PRINTTAB(xman,yman);CHR$(226)
2870COLOUR 2
2880ENDPROC
2890:
2900DEF PROCupdate_time
2910sec$=STR$(((12100-TIME) DIV 100)MOD 60)
2920min$=STR$(((12100-TIME) DIV 6000)MOD 60)
2930REM ** ADD LEADING ZEROS **
2940sec$=LEFT$(zero$,2-LEN(sec$))+sec$
2950min$=LEFT$(zero$,2-LEN(min$))+min$
2960time$=min$+":"+sec$
2970PRINTTAB(11,27);time$
2980ENDPROC
2990:
3000DEF PROCtest_keyboard
3010 REM ** UP ? **
3020IF INKEY(-58)=-1 THEN PROCmove(0,-1)
3030REM ** DOWN ? **
3040IF INKEY(-42)=-1 THEN PROCmove(0,1)
3050REM ** RIGHT ? **
3060IF INKEY(-122)=-1 THEN PROCmove(1,0)
3070REM ** LEFT ? **
3080IF INKEY(-26)=-1 THEN PROCmove(-1,0)
3090ENDPROC
3100:
3110DEF PROCsnipe
3120xstart=RND(750)+220
3130yfinish=RND(750)+220
3140dx=32:dy=(yfinish-ystart)/32
3150GCOL 3,3
3160PROCline
3170IF POINT(x,y)=1 THEN PROCexplode(x,y) ELSE PROCline
3180ENDPROC
3190:
3200REM **** LEVEL 3 PROCEDURES ****
3210:
3220 DEF PROCmove(delta_x,delta_y)
3230REM ** RUB OUT OLD POSITIONS **
3240COLOUR 1
3250PRINTTAB(xdet,ydet);" "
3260PRINTTAB(xman,yman);" "
3270REM ** MOVE DETECTOR **
3280xdet=xdet+delta_x
3290ydet=ydet+delta_y
3300REM ** TEST FOR LIMITS **
3310IF xdet>17 THEN xdet=17
3320IF ydet>25 THEN ydet=25
3330IF xdet<2  THEN xdet=2
3340IF ydet<1  THEN ydet=1
3350REM ** CALCULATE MAN'S COORDS **
3360xman=19-xdet
3370yman=26-ydet
3380PROCconvert(xman,yman)
3390IF POINT(xgraph,ygraph)=2 THEN PROCexplode(xgraph,ygraph)
3400PROCconvert(xdet,ydet)
3410IF POINT(xgraph,ygraph)=2 THEN PROCfound_mine
3420PROCposition_chars
3430ENDPROC
3440:
3450DEF PROCline
3460SOUND0,-8,4,5
3470x=xstart:y=ystart
3480MOVE x,y
3490REPEAT
3500DRAW x,y
3510x=x+dx:y=y+dy
3520UNTIL x>xfinish OR POINT(x,y)=1
3530ENDPROC
3540:
3550DEF PROCexplode(x_explode,y_explode)
3560REM ** SOUND EFFECT **
3570SOUND 0,-15,6,50
3580REM ** SET FLASH RATE **
3590*FX9,20
3600*FX10,50
3610FOR I=1 TO 100
3620MOVE x_explode,y_explode
3630VDU19,2,RND(15),0,0,0
3640GCOL 0,RND(3)
3650PLOT 1,RND(100)-50,RND(100)-50
3660NEXT I
3670PROCreset
3680ENDPROC
3690:
3700REM **** LEVEL 4 PROCEDURES ****
3710:
3720 DEF PROCconvert(xchar,ychar)
3730xgraph=64*xchar+32
3740ygraph=1023-(32*ychar+16)
3750ENDPROC
3760:
3770DEF PROCfound_mine
3780REM ** SOUND EFFECT **
3790SOUND 2,-15,170,3
3800REM ** INCREMENT SCORE **
3810COLOUR 2
3820score=score+150
3830score$=STR$(score)
3840 score$=LEFT$(zero$,5-LEN(score$))+score$
3850PRINTTAB(11,28);score$
3860ENDPROC
3870:
3880DEF PROCreset
3890count=count+1
3900IF count>4 THEN end_flag=1:ENDPROC
3910CLS
3920VDU19,2,2,0,0,0
3930COLOUR 2
3940PROCinitialise_variables
3950mines_left=factor*score/150
3960PROClay_mines(mines_left)
3970PROCdraw_border
3980PRINTTAB(2,27);"Time"
3990PRINTTAB(2,28)"Score"
4000PRINTTAB(11,28)score$
4010PRINTTAB(2,29)"Hi score"
4020PRINTTAB(11,29)hi_score$
4030remaining_men$=LEFT$(men$,4-count)
4040COLOUR 1
4050PRINTTAB(2,30);remaining_men$;"    "
4060COLOUR 2
4070PROCposition_chars
4080ENDPROC
4090DEF PROCmusic
4100REM ** 1ST BAR **
4110SOUND1,-8,213,5
4120SOUND1,-8,209,5
4130SOUND1,-8,213,5
4140SOUND1,-8,209,5
4150SOUND1,-8,213,5
4160SOUND1,-8,193,5
4170SOUND1,-8,205,5
4180SOUND1,-8,197,5
4190REM ** 2ND BAR **
4200SOUND1,-8,185,20
4210SOUND1,-8,165,5
4220SOUND1,-8,185,5
4230SOUND1,-8,193,20
4240REM ** 3RD BAR **
4250SOUND1,-8,165,5
4260SOUND1,-8,193,5
4270SOUND1,-8,197,20
4280ENDPROC
```

Chic in Schale

Obwohl der Apricot F1e vorwiegend für kommerzielle Zwecke gedacht ist, erscheint er auch für Ausbildungszwecke und einen semiprofessionellen Einsatz sehr interessant, zumal er zu einem günstigen Preis angeboten wird.

Das Grundkonzept des F1e ist jedem vertraut, der die anderen Apricot-Rechner kennt: Die Hardware umfaßt eine F1-Tastatur, die Rechnereinheit mit eingebautem Diskettenlaufwerk (einseitig, einfache Dichte) und einen 8-Zoll-Monochrommonitor. Wie beim Apricot Portable werden die Eingaben der Tastatur und auch der auf Wunsch lieferbaren Maus drahtlos über Infrarotsignale an den Rechner übermittelt. Dem Trend zu immer kleineren Abmessungen folgend ist der F1e mit nur 200 mm Breite bemerkenswert schmal. Dafür nimmt er auf dem Schreibtisch eine Tiefe von 425 mm in Anspruch.

Die Tastatur des F1e entspricht dem gewohnten Apricot-Design. Die Tasten sind wie beim Sinclair QL wenig gegeneinander abgesetzt, was für das Blindschreiben und damit für die Textverarbeitung nicht sehr günstig ist. Das Anschlaggefühl ist einem Bürorechner angemessen, obwohl das dezente Klappern der Tasten ängstliche Gemüter vielleicht über die Lebensdauer der Tastatur nachdenken läßt. Wie beim Apricot Portable finden sich über dem Tastenfeld noch vier unscheinbare Druckknöpfe mit den Beschriftungen „Reset", „Repeat Rate" (Wahl der automatischen Tasten-Wiederholfrequenz), „Set Time" und „Keyboard Lock" (Tastatursperre).

In das Rechnergehäuse ist vorn ein 3½-Zoll-Sony-Einzellaufwerk eingebaut, wie es zunehmend auch von anderen Herstellern verwendet wird. Zudem besteht die Möglichkeit, ein 5¼-Zoll-Laufwerk anzuschließen. Wer den F1e ins Auge faßt, kann daher wegen des Softwareangebots unbesorgt sein. Auf die Microfloppy passen trotz einfacher Dichte und einseitiger Aufzeichnung 315 KByte, also mehr als auf die meisten 5¼-Zoll-Disketten.

Vier Leuchtdioden

Die vier Leuchtdioden links vorne am Rechner zeigen Betriebsspannung, Tastatur-Umschaltung, Scroll (Bildschirm-Rollmodus) und Laufwerkbereitschaft an. Etwas tiefer sind die Infrarotdetektoren angebracht.

Auf der rechten Gehäuseseite des Apricot-Rechners befindet sich ein 60poliger Buserweiterungsstecker, an den diverse Erweiterungskarten oder auch ein Zusatzlaufwerk angeschlossen werden können. Mit dem MSD-Erweiterungssystem, das eine 10-Megabyte-Festplatte beinhaltet, wird sogar die Ausbaustufe des Apricot XI erreicht.

Die übrigen Interfaces liegen auf der Rückseite des Geräts, die ausreichend viele Peripherieanschlüsse ermöglichen: ganz links ein 25poliger D-Stecker für serielle Geräte mit RS232-Schnittstelle, daneben zwei Monitorbuchsen, die eine für einen RGB-Farbmonitor (neunpolig, die auch für die ACT-Monochromgeräte geeignet ist), die andere mit Composite-Video-Signal und ganz rechts eine Centronics-Schnittstelle.

Der Apricot F1e wird wahlweise mit oder ohne Monitor geliefert. Wegen des fehlenden TV-Modulators braucht man zum Anschluß eines normalen Fernsehapparats jedoch noch

Der Apricot F1e zeichnet sich rein äußerlich durch sein ansprechendes Design aus. Er wird mit integrierter 3½-Zoll-Floppy und MS-DOS-Betriebssystem zu einem akzeptablen Preis geliefert.

einen zusätzlichen passenden Adapter.

Weil der Monitor nicht zur Grundausstattung gehört und die meisten Fremdgeräte ein eigenes Netzteil haben, hat man sich die Monitor-Spannungsversorgung gespart – für den Betrieb der ACT-Bildschirme (ohne Netzteil) brauchen Sie daher eine externe 17-Volt-Spannungsquelle.

Das ist um so kurioser, weil der Hersteller zur Vermeidung von Kabelsalat gerade die Infrarotkommunikation eingeführt hat und vergleichbare Rechner (u. a. auch ACT's Flaggschiff Apricot) ohne diese Zusatzkästchen auskommen.

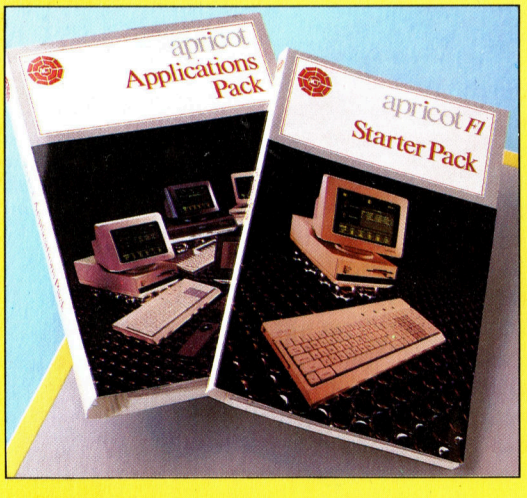
Im Innern des Rechnergehäuses ist ein weiterer Steckplatz für Zusatzkarten vorgesehen. Die F1e-Platine sieht wie bei anderen Rechnern dieser Preisklasse recht elegant aus und ist gegen das Laufwerk und das Netzteil (das einige Temperatur entwickeln kann) durch ein wärmeableitendes Blech abgeschirmt.

16-Bit-Prozessor 8086

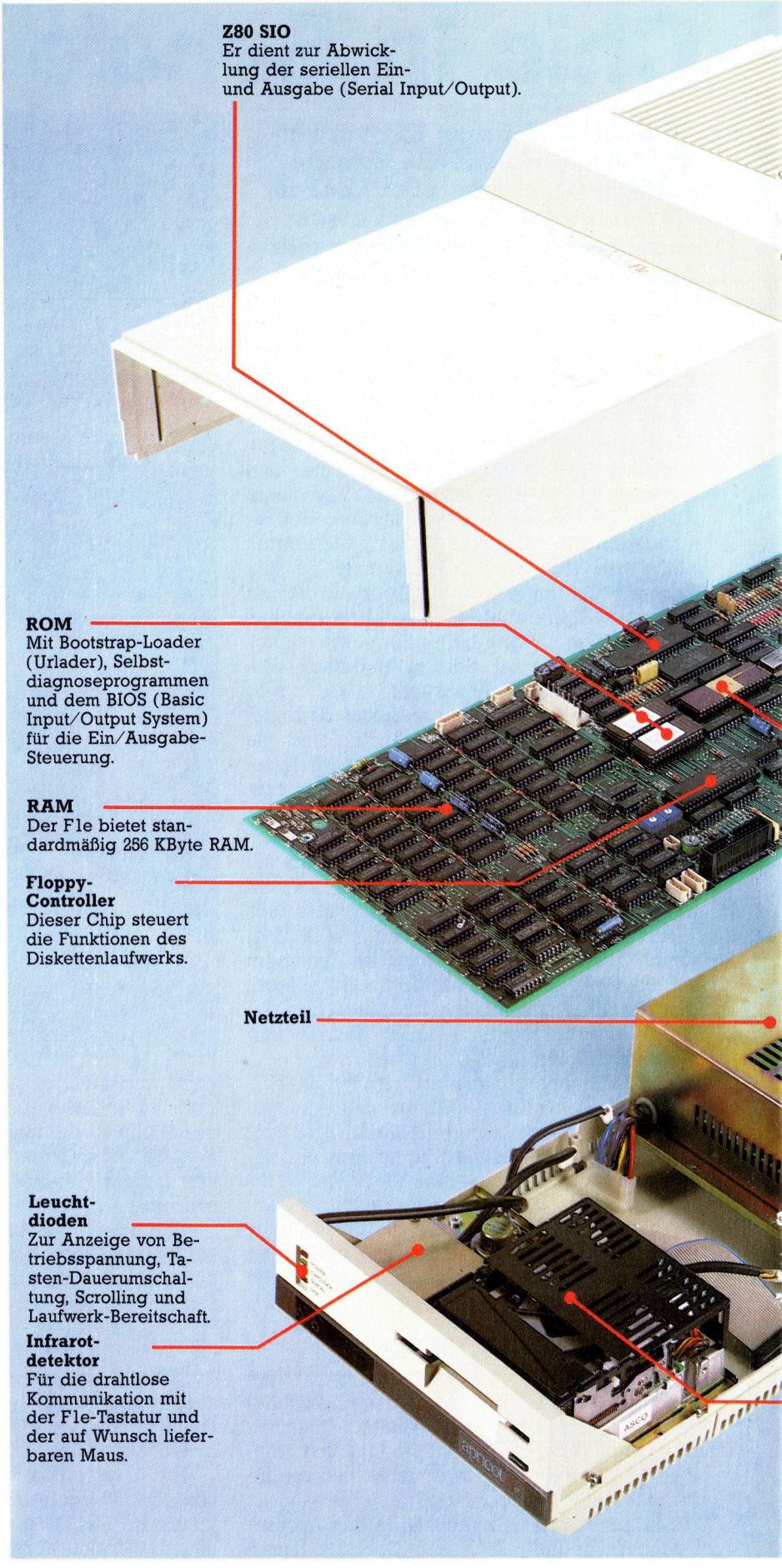
Der F1e arbeitet mit dem 16-Bit-Prozessor 8086 von Intel und dem bekannten MS-DOS-Betriebssystem. Aufgrund der Piktogramm-Steuerung wird die Arbeit mit diesem Betriebssystem erheblich erleichtert. Das zugehörige Steuerprogramm heißt „Activity". Damit sind Sie als Benutzer bald vertraut.

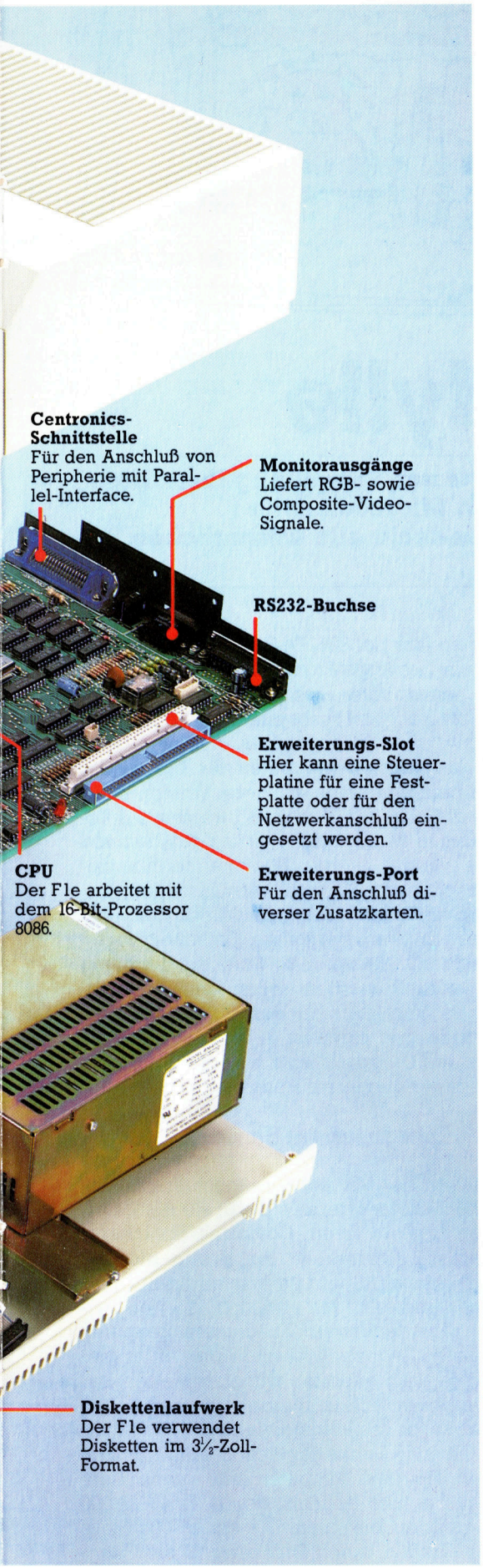
Das Activity-System ist ein „objektorientiertes" Programm. Der Anwender muß also keine Befehlsfolgen eintippen. Zum Laden von Dateien beispielsweise wählt man nur das entsprechende Piktogramm aus dem Menüangebot auf dem Schirm – den Rest erledigt der Computer allein.

Bei den meisten Systemen mit Piktogramm-Menüs erfolgt die Cursorsteuerung aus-



Super-Software
Im Preis des F1e ist ein Softwarepaket enthalten, das ein Textverarbeitungssystem, ein „elektronisches Notizbuch" und ein Tabellenkalkulationsprogramm mit vielen Möglichkeiten (z. B. Fenstertechnik) umfaßt.





schließlich über die Maus. Beim F1e läßt sich jedoch auch der Zehnerblock zur Führung des Cursors einsetzen: Die Zifferntasten 1 bis 9 bilden einen 3×3-Block, den man sich für die Vorgabe der Cursor-Laufrichtung als eine Art Kompaßrose (bei Aussparung der mittleren Taste) vorstellen kann. Die Betätigung der mittleren Taste in der oberen Reihe (das ist die Acht) führt den Cursor beispielsweise nach oben, wogegen ein Druck auf die Taste rechts unten (3) ihn in die rechte untere Bildschirmecke sendet. Zur Auswahl eines Piktogramms drückt man auf die ENTER-Taste.

Unter den Programmen auf der mitgelieferten Activity-Systemdiskette ist auch eins, das Sie in die Benutzung von Activity einweist. Es erläutert unter anderem den Gebrauch des Piktogramm- und des Zeichen-Editors (mit dem Sie ihren eigenen Zeichensatz generieren können) sowie des „Konfigurators", der die Inbetriebnahme des Rechners in Verbindung mit beliebigen Peripheriegeräten erlaubt.

Wie es sich für einen Rechner gehört, der einen Anspruch auf kommerziellen Einsatz erhebt, wird der F1e mit einem Paket von Anwendungsprogrammen geliefert. Der „Super Writer" ist ein Textverarbeitungsprogramm auf WordStar-Basis, aber ohne dessen Freizügigkeit in der Formatierung.

„Super Planner"

Der „Super Planner" soll ein „elektronisches Notizbuch" sein, das Ihnen eine weitsichtige Planung Ihrer Aktivitäten erlaubt. Dazu gehören ein Adreßbuch, ein Kalender, ein Tages-Vormerkbuch und ein kleines Datei-Organisationsprogramm. Der Super Planner erinnert entfernt an ein Datenbanksystem, verfügt aber nicht über die spezifischen Suchalgorithmen, die eine echte Datenbank auszeichnen.

Das dritte Programm heißt „Super Calc" – ein Tabellenkalkulationsprogramm für Finanzierung und Rechnungswesen. Es ist umfassender als der Writer und der Planner und bietet zahlreiche Möglichkeiten – unter anderem für die Fenstertechnik und die Textberichtigung –, Fähigkeiten also, die man von einem professionellen Tabellenkalkulationsprogramm erwartet.

Mit dem scharf kalkulierten Preis möchte ACT offensichtlich gezielt in den Ausbildungsbereich vordringen. In England dürfte der F1e daher vor allem dem Acorn B das Leben schwer machen, der dort bis vor kurzem noch den Markt der Schul- und der kleinen Bürorechner beherrschte.

Im Heimcomputerbereich wird der Apricot F1e wohl kaum ein solcher Verkaufsschlager wie der Sinclair Spectrum werden. Dieser Rechner ist jedoch all den Anwendern zu empfehlen, die einen preisgünstigen Computer mit der Ausbaumöglichkeit zu einem kompletten Bürosystem suchen.

Apricot F1e

Abmessungen
425×200×105 mm

Zentraleinheit
16-Bit-Prozessor Intel 8086 mit 4,7 MHz Taktfrequenz

Speicherkapazität
Standardmäßig 256 KByte RAM

Bildschirmformat
Textdarstellung: 25 Zeilen zu 80 oder 50 Zeilen zu 132 Zeichen; Grafikauflösung maximal 800×400 Punkte.

Schnittstellen
RS232-Anschluß, Centronics-Parallelinterface, RGB- und Composite-Video-Monitorbuchsen.

Diskettenlaufwerk
3½-Zoll-Einzellaufwerk, Kapazität 315 KByte.

Betriebssystem
MS-DOS, CP/M-86 und Concurrent CP/M.

Handbücher
Die Anleitung für die Anwendungs-Software ist ausführlich und entspricht dem gewohnten ACT-Niveau. Das "Starter Manual" könnte dagegen mehr Informationen über die Maschine selbst enthalten.

Stärken
Der F1e bietet ein ausgewogenes Preis/Leistungsverhältnis.

Schwächen
Das wenig griffige Tastenfeld bleibt deutlich hinter dem sonst üblichen Apricot-Standard zurück, und trotz seines schnellen Prozessors erreicht der F1e nicht die Rechengeschwindigkeit ausstattungsmäßig vergleichbarer Maschinen.

Ländliche Idylle

Schon seit einigen Jahren setzen Landwirte zur Verwaltung ihrer Höfe Computersysteme ein. Die auf den Dragon Microcomputer ausgerichtete Programmreihe von Farmfax deckt alle wesentlichen Aspekte eines modernen Bauernhofs ab.

Bei dem breiten Einsatzspektrum für Computer ist es kaum eine Überraschung, wenn am Markt eine Programmreihe zur Verwaltung von Bauernhöfen auftaucht. Es erregt jedoch Aufsehen, daß für diese Aufgabe der zuvor fast gescheiterte Dragon-Rechner eingesetzt wird, dessen Herstellung inzwischen nach Spanien verlagert wurde. Im gesamten europäischen Bereich erfreut sich dieses Gerät bei Landwirten immer größerer Beliebtheit.

Die Farmfax-Serie wurde von dem Landwirt Geoffrey Paterson entwickelt. Schon der Aufbau zeigt, daß hier Experten am Werk waren: Jedes Paket ist auf eine bestimmte landwirtschaftliche Aufgabe ausgerichtet. Die Programme werden als ROM-Cartridges geliefert, die sofort nach dem Anschluß benutzt werden können und für Computerneulinge gedacht sind. Einige Cartridges haben zusätzliche RAM-Speicher für schnellen Zugriff.

Ein gutes Beispiel für die Steuerung eines Hofes per Computer liefert Robin Dunkerley, der im Umland von Oxford 500 Morgen Land bearbeitet. Etwa die Hälfte dient als Weide für 150 Holsteiner und friesische Rinder, während das restliche Land für den Anbau eingesetzt ist. Ein Problem, das das Farmfax-Programm lösen konnte, ist der „Calving Index" – die Anzahl der Tage, an denen eine Kuh zwischen zwei Trageperioden Milch geben kann. Dazu Robin Dunkerley: „Nach einer Faustregel kostet jeder Tag, den eine Kuh über 365 Tage hinaus nicht trächtig ist, etwa acht Mark. Früher hatten wir Zyklen mit bis zu 390 Tagen, wobei der Durchschnitt bei 375 Tagen lag. Seit der Computer uns jedoch mitteilt, welche Kühe noch nicht gedeckt wurden, haben wir keine einzige vergessen. Unser ‚Calving Index' lag im letzten Jahr bei 366. Es läßt sich, glaube ich, kaum ein besseres Ergebnis erzielen."

Die sogenannte Einzel-Kuh-Datei („Individual Cow Records") gibt einen Einblick in die Arbeitsweise des Programms. Auf einer Cartridge lassen sich die Daten für bis zu 240 Kühe speichern (eine billigere Version speichert auf Cassette). Alle Daten werden zur Bearbeitung in den Arbeitsspeicher geladen und danach wieder in das Speichermedium zurückgeschrieben. Obwohl die Cassettenspeicherung langsam arbeitet, fällt diese Zeit nicht ins Gewicht, wenn wichtige Entscheidungen anstehen, zum Beispiel, zu welchem Zeitpunkt Kühe verkauft werden sollen, deren Milchertrag im Verhältnis zu den Futterkosten zu stark zurückgegangen ist. In jedem Fall ist der Einsatz des Dragon billiger als jede auf einem Großrechner gemietete Rechenzeit.

Tierdaten im Speicher

„Wir haben der Milchgenossenschaft pro Monat etwa 160 Mark gezahlt, damit sie die Daten unserer Kühe in ihrem Computer speichert", erinnert sich Dunkerley. „An jedem Monatsende gaben wir ihnen die notwendigen Informationen und erhielten etwa drei Wochen später die Auswertungen. Zu diesem Zeitpunkt konnten wir jedoch nicht mehr darauf reagieren. Außerdem erhielten wir eine Menge Daten, die bei einer Zuchtherde interessant sind, nicht aber für die kommerzielle Milchproduktion." Obwohl Dunkerley schon lange wußte, daß ein eigener Computer eine große Hilfe sein würde, war er nicht bereit, rund 10 000 Mark dafür zu bezahlen. Die etwa 3000 Mark für den Dragon, einen Drucker und die Farm-

fax-Programme waren jedoch vertretbar. Die Investition machte sich schnell bezahlt, da er nun jederzeit die gewünschten Informationen abrufen kann, statt viel Geld für Daten zu bezahlen, die teils unwichtig waren bzw. oft viel zu spät kamen.

Monatliche Auswertung

An jedem Monatsende sitzt Dunkerley etwa eine Stunde an seinem Computer und organisiert und bewertet seine Daten. Das Programm für Milchwirtschaft („Dairy Management") gibt ihm die Spanne zwischen den Kosten für das eingesetzte Futter und dem Einkommen aus dem Verkauf der Milch an, dazu noch sein Bruttoeinkommen, wieviel Futter zur Erzeugung eines Liters Milch nötig war, die Gesamtmenge der Milch und andere wichtige Daten. Danach hilft ihm das Programm „Dairy Ration Formulation" sicherzustellen, daß die optimale Milchmenge produziert wird. Dieses Modul erstellt eine Nährwertberechnung des Futters. Eingegeben wird die Energiemenge in Mega-Joule, die Menge verdaulichen Rohproteins und der Ballaststoffe pro Kilo Futtermittel. Durch Eingabe des Preises und die Angabe, ob das Futter gekauft oder selbst produziert wurde, kann Dunkerley für jede Kuh problemlos die Gesamtkosten und die Unterhaltskosten pro Tag feststellen.

Wenn für jede Kuh die tägliche Milchmenge eingegeben wird, liefert das Programm zur Bearbeitung der Einzel-Kuh-Datei am Ende jedes Monats eine exakte Bewertung. „Bei der Änderung nur eines Faktors berechnet das Programm alle Werte neu", erklärt Dunkerley. „Mit dem Computer läßt sich diese Aufgabe in weniger als einem Drittel der sonst nötigen Zeit erledigen. Das Programm liefert mir außerdem Daten, die ich immer schon haben wollte, beispielsweise die Überziehung meines Bankkontos. Da ich Pächter bin, muß ich zweimal im Jahr auf den Tag hinarbeiten, an dem die Pacht fällig ist. Mit dem Gerät kann ich mein Budget vorausplanen und brauche meinen Kredit nicht mehr über Gebühr zu strapazieren."

Die Farmfaxmodule sind sehr benutzerfreundlich. Mitgeliefert wird eine einfache Anleitung, die sich in wenigen Minuten durcharbeiten läßt. Jedes Programm beginnt mit einem einfachen Menü. So gibt „Cullcow" – eine Entscheidungshilfe für den Verkauf von Kühen, deren Ertrag unter einen bestimmten Punkt gefallen ist – die Wahl zwischen zwei Bearbeitungsmöglichkeiten:

SEITE 1 — PREISINFORMATIONEN
SEITE 2 — BESTER KAUF/VERKAUF

Auf Seite 1 gibt der Landwirt zunächst den Monat und das Jahr ein, dann den aktuellen Zinssatz, den Milchpreis, den täglichen Milchertrag pro Kuh in Litern und die monatlichen Kosten. Um Eingabezeit zu sparen, werden die letzten vier Daten auf alle Einträge kopiert. Sie lassen sich jedoch jederzeit ändern.

Auf Seite 2 kann der Landwirt entscheiden, ob er Kühe kaufen oder verkaufen will. Er gibt dann die Anzahl Kühe pro Einheit, den angestrebten Preis und den Verkaufsmonat an. Der Computer vergleicht alle Zahlen und setzt einen Stern hinter die Kuh mit dem besten Preis. Mit der BREAK-Taste kann der Anwender jederzeit in den Index zurückkehren und dann (falls notwendig) Daten verändern.

Die anderen Programme arbeiten nach dem gleichen Schema, wobei umfangreichere Programme natürlich auch im Hauptmenü mehr Möglichkeiten bieten. Das „Dairy Ration Formulation" enthält beispielsweise folgende Eröffnungsseite (hier übersetzt):

SEITE 1 — FUTTERTABELLE
2 — MINERALIEN
3 — VERGLEICHBARKEIT
4 — GRUPPENBESCHREIBUNGEN
5 — FUTTERZUSAMMENSETZUNG
6 — FUTTERQUANTITÄTEN
7 — SPEICHERN/EINLESEN

Eine Cartridge ohne Speicher ruft beim Einschalten automatisch Seite 7 auf, damit zuvor gespeicherte Daten von der Cassette geladen werden können. In allen anderen Fällen blendet das Hauptmenü auf. Jedes Futtermittel, dessen Preis auf Seite 1 eingetragen wurde, erscheint automatisch auf Seite 2, auf der der Anwender die darin enthaltenen Mineralien angeben kann. Seite 3 berechnet die Daten und liefert eine Übersicht, wieviel Energie, Ballaststoffe und Protein die einzelnen Futterarten pro Geldeinheit bieten. Wenn der Landwirt auf Seite 4 die Ertragsziele eingibt – Körpergewicht der Kühe, Ausfalltage für das Kalben, Milch in Litern und deren Butterfettgehalt – stellt das Programm die Futterration für die einzelnen Kühe zusammen und errechnet die Gesamtfutterkosten für eine bestimmte Anzahl Kühe über einen festgelegten Zeitraum.

Die Programme wurden von Experten für Experten geschrieben. Landwirte wie Robin Dunkerley schätzen die Hilfe, die das System ihnen in einer mehr und mehr industrialisierten Landwirtschaft mit vielfachen Behördenauflagen bietet: „Bei den Milchquoten der EG und dem ständig wachsenden Druck müssen wir unsere Höfe so produktiv wie möglich führen. Das ist inzwischen nur noch mit dem Computer möglich. Ich bin zwar nicht sicher, ob wir dadurch mehr Geld verdienen, ich weiß aber, daß wir damit eine Menge Geld sparen."

Die Farmfax-Reihe: Für den Dragon
Vertrieb: Farmfax Computer Systems, PO Box No. 2, Stockbridge, Hampshire SO20 6LE
Format: Cartridge

Keine Hexerei

Bisher haben wir uns im Kurs auf die Techniken der strukturierten Programmierung beschränkt. Strukturierte Programme sind einfach zu entwickeln, und die Fehlersuche ist leicht. Jetzt werden Methoden zur Erhöhung der Verarbeitungsgeschwindigkeit vorgestellt.

Die strukturierte Programmierung und eine gute Programmauslegung sind Methoden, die ein Computerprogramm zwar anwenderfreundlich machen, seine Geschwindigkeit aber nicht unbedingt erhöhen.

Die Arbeitsweise von Interpreter-Programmiersprachen wie BASIC führt oft zu einem wahren Schneckentempo beim Programmablauf. Am besten lassen sich BASIC-Programme durch Compilierung beschleunigen. – Allerdings verfügen nur die wenigsten Heimcomputer über einen leistungsfähigen Compiler. Compiler sind sowohl in Cassetten- wie Diskettenversion erhältlich, die meisten verarbeiten aber nur speziell formatierte Programme. Die Compilierung während der Programmentwicklung ist ein zeitraubender Prozeß. Der Compiler belegt – je nach Qualität – mehr oder minder große Teile des Arbeitsspeichers.

Am stärksten wird die Ablaufgeschwindigkeit jedoch durch den Zugriff auf Files verlangsamt. Verzögerungen sind unvermeidlich, wenn ein Programm – etwa eine Datenbank – wiederholt die Cassette oder Diskette liest bzw. beschreibt. Der Zugriff auf ein Record in einem Random-Access-File dauert im allgemeinen etwa eine Viertelsekunde. Je nach Dateiumfang ist der Vorgang bei seriellen Files und bei Cassetten-Speicherung noch zeitraubender. Durch das Einlesen einer größeren Datenmenge ins RAM und das Zwischenspeichern der veränderten Daten mehrerer Files läßt sich die Geschwindigkeit wesentlich erhöhen.

Bei Dialogprogrammen wartet der Benutzer oft sekundenlang vor einem leeren Bildschirm. Dagegen hilft eine Programmänderung: Die Filezugriffe werden auf einen Zeitpunkt verschoben, an dem der Anwender beschäftigt ist (etwa mit dem Lesen der auf dem Bildschirm dargestellten Anweisungen).

Eine weitere Ursache für den langsamen Programmablauf ist das Berechnen von Realzahlen (Zahlen mit Nachkommastellen). Zur Ausführung einer Rechenoperation braucht der Prozessor bei einer realen Zahl sehr viel mehr Taktzyklen als bei einer ganzen Zahl (Integer). Wird in einem Programm viel gerechnet, kann sich das Ersetzen aller verwendeten Variablen durch ganzzahlige Variablen lohnen (zum Beispiel wird SUM durch SUM% ersetzt). Schon bei Programmen mit einem bescheidenen mathematischen Anteil können bis zu 20 Prozent Zeit gespart werden, bei reinen Rechenprogrammen sogar bis zu 50 Prozent.

Schnelle Algorithmen

Die Entwicklung schneller Algorithmen ist die beste Methode zur Beschleunigung eines Programms – einige davon haben wir in diesem Kurs bereits vorgestellt. BASIC verfügt über eine Vielfalt von Funktionen (wie INSTR, SGN, LOG), die mit optimalen Algorithmen in Maschinensprache geschrieben und dadurch sehr schnell sind. Bevor Sie also Ihre eigenen Funktionen programmieren, sollten Sie noch einmal einen Blick ins Handbuch werfen. Auch die mit DEF FN selbstdefinierten Funktionen sind schnell und besonders in Programmen mit wiederholten Berechnungen oder mehrfachen String-Manipulationen nützlich. Durch den Einsatz dieser Funktionen kann man Verzögerungen durch den Sprung zu einem langsameren Unterprogramm vermeiden.

Programmierung in Maschinensprache sorgt immer für eine höhere Verarbeitungsgeschwindigkeit, weil die zeilenweise Übersetzung des BASIC-Programms in Maschinenbefehle dabei wegfällt. Unglücklicherweise ist das Schreiben eines Assembler-Programms sehr viel schwieriger als zum Beispiel das Programmieren in BASIC.

Compiler für Heimcomputer

Acorn B Turbo Compiler Salamander, 17 Norfolk Road, Brighton BN 1 3AA, GB	Cassette
Commodore 64 DTL-BASIC Compiler Dataview Wordcraft Ltd., Radix House, East Street, Colchester C01 2XB, GB	Cassette/ Diskette
Sinclair Spectrum Softek FP Softek IS FP und IS zusammen Softek International, 12/13 Henrietta St, London WC2, GB	Cassette Cassette Cassette

Es gibt eine Vielzahl kleiner Kniffe, mit denen sich die Verarbeitungsgeschwindigkeit steigern läßt, etwa der Gebrauch einer Variablen anstelle einer Zahl. Besonders bei Schleifen steht der Wert dann schneller bereit. Anfangsbuchstaben von Variablennamen sollten verschieden und gleichmäßig über das Alphabet verteilt sein. Möglichst viele Befehle in einer Zeile und ein fester Abstand zwischen den Zeilennummern (etwa 10) beschleunigen die Programme ebenfalls. Wenn der Interpreter es erlaubt, sollte bei FOR . . . NEXT-Schleifen auch die Variable des Schleifenzählers weggelassen werden. Innerhalb von Schleifen sollte die wiederholte Berechnung gleicher Werte vermieden werden. Ermitteln Sie diese außerhalb der Schleife und fügen Sie sie als Variablen ein.

Ganzzahlige Arithmetik spart nicht nur Zeit, sondern auch Speicherplatz. Eine ganze Zahl belegt nur zwei Byte, während für eine Realzahl vier bis fünf Byte gebraucht werden. Speziell in großen Arrays kann also viel Platz gespart werden. Auch die Verbindung von schnellerer Programmausführung mit der Einsparung von Speicherplatz ist möglich, indem man eingebaute bzw. selbstdefinierte Funktionen sowie Maschinensprache einsetzt und viele Befehle in einer Zelle unterbringt. All dies spart neben Zeit auch Platz. Compilierung dagegen macht kleine Programme größer. Nur bei umfangreichen Programmen wird dadurch der Platzbedarf reduziert.

Das Entfernen der REM-Zeilen und gekürzte Textanweisungen sparen ebenfalls Speicherplatz. Große Textblöcke werden in externen Files gespeichert (Anweisungen und „Help"-Files belegen oftmals sehr viel Platz). Befehlszeilen sollten keine überflüssigen Leerzeichen enthalten. Variablennamen wählt man genau wie die Zeilennumerierung möglichst kurz. Bei der Dimensionierung eines Datenfeldes empfiehlt es sich, präzise vorzugehen. Kennt man den genauen Platzbedarf vorher noch nicht, sollte man mit einer Variablen dimensionieren. Dazu ein Beispiel:

```
10 INPUT„Wieviele Faelle gehoeren in diese
   Kategorie?";FAELLE%
20 DIM ARRAY%(FAELLE%)
```

Über diese Technik der „dynamischen Dimensionierung" verfügt neben BASIC kaum eine andere Computersprache – man sollte sie also auch nutzen!

Mit Befehlen wie etwa HIMEM kann die RAM-Speicherverwaltung in BASIC geändert werden. Normalerweise wird damit der durch das BASIC verwaltete RAM-Bereich verschoben. Oft soll mit dem Befehl ein Maschinenprogramm an eine vor Überschreibung sichere Stelle verschoben werden, man kann sich damit aber auch Zugang zu weiterem Speicherplatz erschließen, der im Normalfall als Bildschirm-Speicher dient. Wenn es auf die Bildschirmdarstellung nicht ankommt, läßt sich so ein zusätzliches KiloByte gewinnen. Auch ohne HIMEM können Bildschirmspeicherplätze oft mit PEEK- bzw. POKE-Befehlen direkt angesprochen werden.

Paßt ein Programm nach Ausreizen aller Möglichkeiten immer noch nicht in den Speicher, lassen sich bei einigen BASIC-Versionen mit dem CHAIN-Befehl Programme aneinanderkoppeln. Manche Rechner können mit COMMON bestimmte Variablenwerte an das zweite Programm übergeben. Durch CHAIN – falls überhaupt vorhanden – werden alle oder keine Variablen übergeben.

Strukturierte Programmteile sollten einzeln geschrieben, getestet und auch auf ihren Zeitbedarf geprüft werden. Dazu dient ein einfaches „Stoppuhr-Programm":

```
100 REM   Verwenden Sie diesen ersten
105 REM   Teil zum Setzen aller vom
110 REM   Unterprogramm genutzten
115 REM   Variablen. Alle vorkommenden
115 REM   Felder dimensionieren und mit
120 REM   Daten fuellen. Dieses Pro-
125 REM   gramm laeuft auf dem Acorn B.
130 REM   TIME ist eine Pseudo-Variable,
135 REM   die einen von der System-Uhr
140 REM   erzeugten Wert von Hundert-
145 REM   stelsekunden enthält.
200       START=TIME
210       GOSUB 2000: REM Aufruf des
          zu stoppenden Unterprogramms
220       FINISH=TIME
230       PRINT "Ausfuehrung
          dauerte";(FINISH-START)/100;
          "Sekunden."
240       END
```

Mit diesem Programm können Sie unterschiedliche Algorithmen auf ihre Geschwindigkeit und Wirksamkeit prüfen.

Geschwindigkeit ist keine Hexerei!

- **Wägen Sie gut ab zwischen gutem Programmierstil und schnellen, dafür aber oft schwer verständlichen Befehlsketten.**
- **Möglichst compilieren. Wenn das nicht geht, Funktionen und Abläufe festlegen.**
- **Zugriffe auf Files vermeiden.**
- **Wenn möglich, Ganzzahlen und Variablen verwenden.**
- **Algorithmen sorgfältig entwickeln. Wenn möglich, nutzen Sie auch die Erfahrung anderer Programmierer!**
- **Vor- und Nachteile der Assembler-Programmierung gut abwägen. Assembler ist schnell, aber schwer zu schreiben.**
- **Programm so weit wie möglich komprimieren. REM-Zeilen können Sie in der Endversion weglassen.**

Mit System unterbrochen

Interrupts sind Meldungen, die den Prozessor bei seiner derzeitigen Aufgabe unterbrechen und ihm wichtige Meldungen zukommen lassen. In diesem Artikel erläutern wir ihren Mechanismus.

Beim Auftreten eines Interrupts beendet der Prozessor den laufenden Befehl und schiebt den aktuellen Inhalt des Befehlszählers auf den Stack. Die Vektoradresse des ausgelösten Interrupts wird nun in den Befehlszähler geladen und so die Steuerung an diese Adresse übergeben (normalerweise eine ROM-Adresse). Ein JMP-Befehl verursacht dann den Sprung auf die eigentliche Interruptroutine. Interrupts werden mit einem RTI-Befehl beendet, der die Steuerung über die im Stack abgelegte Rücksprungadresse an das Hauptprogramm zurückgibt. Da sich der JMP-Befehl im RAM befindet, kann er vom Programmierer abgeändert werden, so daß die Steuerung zunächst auf eine Spezialroutine und dann erst auf die normale Interruptroutine läuft.

Ein umfassender Einsatzbereich für Interrupts sind Tastatureingaben. Wenn ein Programm direkt auf die Tastatur zugreift, der Prozessor aber gerade mit anderen Aufgaben beschäftigt ist, sind keine Eingaben möglich. Selbst während der Bearbeitung von Eingaben können Zeichen verlorengehen.

Die Lösung ist einfach: Beim Drücken einer Taste wird der Prozessor bei seiner Aufgabe unterbrochen und eine Interruptroutine ausgeführt. Diese Routine nimmt das eingegebene Zeichen an und speichert es im Tastaturbuffer. Nach diesem „Umweg" macht der Prozessor mit seiner ursprünglichen Aufgabe weiter.

Nun spricht die Eingaberoutine des Betriebssystems die Tastatur nicht direkt an, sondern nimmt das nächste Zeichen aus dem Buffer (oder wartet, bis dort ein Zeichen gespeichert wird). Dieser Mechanismus stellt sicher, daß auch bei schnellem Tippen keine Zeichen verlorengehen.

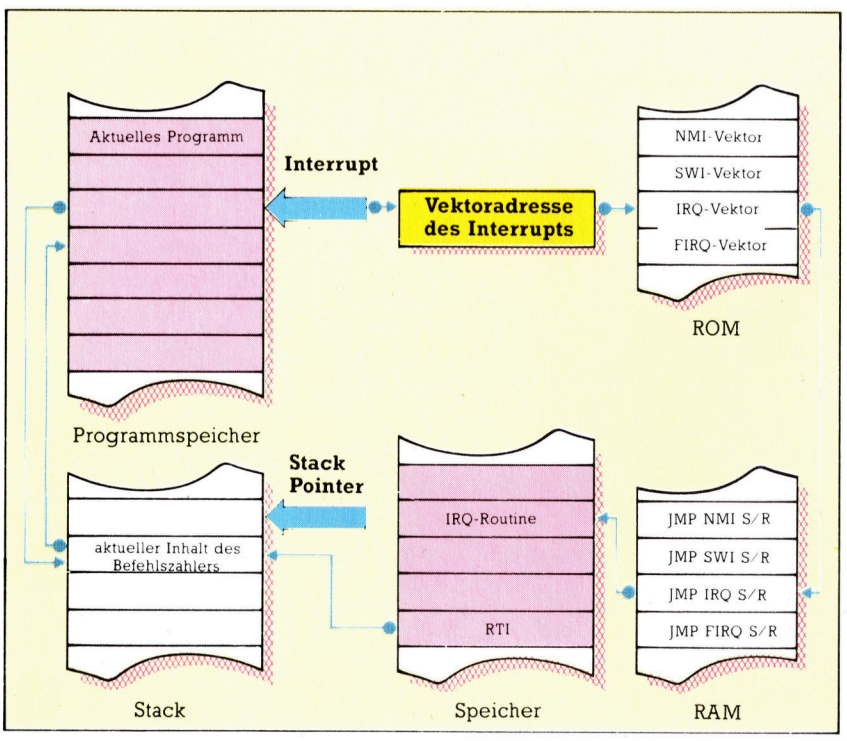
Wenn Eingaben allzu schnell erfolgen, fließt der Buffer über, bevor sie vom Programm bearbeitet werden. Hier hilft nur ein umfangreicher Buffer, der andererseits aber nicht zuviel wertvollen Speicherplatz belegen darf. Ein zweites Problem liegt beim Anwender, der erwartet, daß das eingegebene Zeichen sofort auf dem Bildschirm erscheint.

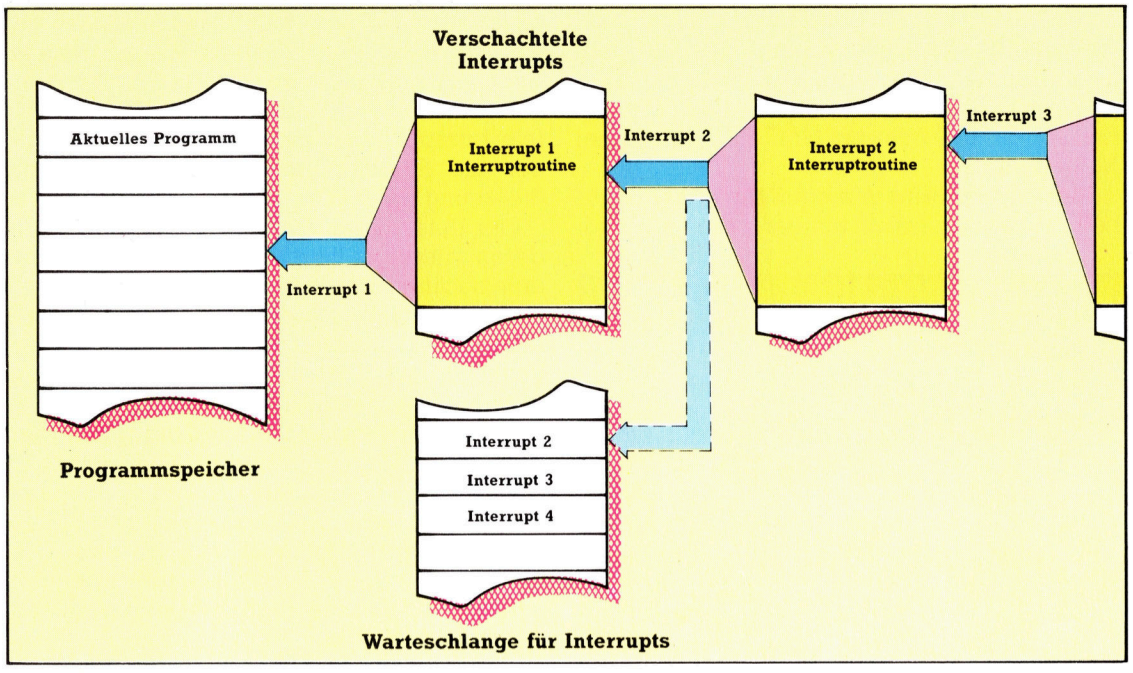
Druckerausgaben sind ein weiteres wichtiges Einsatzgebiet für Interrupts. Bei diesem zeitintensiven Vorgang arbeiten die Prozessoren oft nur 100 Millisekunden (in denen sie Zeichen zum Drucker senden) und warten dann lange Zeit, bis der Drucker die Zeichen verarbeitet hat. Hier bringt ein „Spooler" die Lösung: Alle Dateien, die gedruckt werden sollen, werden in einer Warteschlange zwischengespeichert, und ein Teil der ersten Datei wird in einen kleineren Speicherbuffer geladen. Sobald der Drucker bereit ist, ein weiteres Zeichen zu empfangen, sendet das Interface ein Signal an den Prozessor. Die Interruptroutine übermittelt das nächste im Buffer gespeicherte Zeichen oder holt sich den nächsten Teil aus der Warteschlange. Auf diese Weise läuft der Druckvorgang im Hintergrund, während der Prozessor für andere Aufgaben frei ist.

Bei einigen Abläufen – beispielsweise Diskettenzugriffen – kann eine Unterbrechung zu Datenverlusten oder anderen fatalen Fehlern führen. Durch die „Maskierung" der Interrupts hat der Prozessor jedoch die Möglichkeit, Unterbrechungen während besonders wichtiger Abläufe zu ignorieren. Hier zeigt oft eine Meldung das Auftreten eines Interrupts an, der dann später behandelt wird.

Im umgekehrten Fall – wie etwa bei interruptgesteuerten Diskettenschnittstellen – sind Interrupts nicht maskierbar und haben immer Priorität. Nicht maskierbare Interrupts können aber auch von Schaltungen ausgelöst werden, die ein Abfallen der Stromversorgung erkennen. Solange noch genügend Strom vorhanden ist, speichert eine Spezialroutine sofort die aktuelle Aufgabe.

Wenn mehrere Quellen Unterbrechungen auslösen können, muß man auch die Möglichkeit „verschachtelter" Interrupts in Betracht ziehen. Ein Interrupt, das während der Behandlung eines anderen Interrupts auftritt, läßt sich auf zwei verschiedene Arten abfangen. Zunächst kann das neue Interrupt so lange ignoriert werden, bis das laufende beendet ist.





Wenn während eines Interrupts eine neue Unterbrechung auftritt, können Interrupts verschachtelt werden. Dabei wird beim Auftreten eines Interrupts der Befehlszähler auf den Stack geschoben, der neue Interrupt ausgeführt und dann die Steuerung an die gespeicherte Adresse zurückgegeben. Verschachtelungen dieser Art sind natürlich von der Kapazität des Stacks abhängig und auch von der Fähigkeit der Interruptroutinen, Verzögerungen abfangen zu können.

Der Prozessor kann aber auch die Einzelheiten jedes Interrupts in einer speziellen Warteschlange speichern. Sobald das erste Interrupt abgeschlossen ist, untersucht der Prozessor diese Schlange und führt jedes dort gespeicherte Interrupt nacheinander aus, bis er schließlich die Steuerung wieder an das Hauptprogramm zurückgibt.

Interrupts lassen sich aber auch mit unterschiedlichen Dringlichkeitsstufen versehen, damit höhere Prioritäten niedrigere Prioritäten unterdrücken können. Das funktioniert aber nur, wenn das Betriebssystem verschachtelte Interrupts zuläßt.

Der SWI-Befehl ist eine bequeme Möglichkeit, über die Erzeugung eines eigenen Interrupts vom Programm in das Betriebssystem zurückzukehren. Diese Methode heißt „Software Interrupt" (zur Unterscheidung zu den bisher untersuchten „Hardware Interrupts"). SWI-Befehle lassen sich auch als Unterbrechungspunkte in Maschinencodeprogramme einsetzen und erleichtern so die Fehlersuche.

Registerinhalte untersuchen

Der Anwender bestimmt, an welchen Punkten die Programmausführung eine Pause machen soll. Die an diesen Punkten stehenden Befehle werden gegen SWIs ausgetauscht. Beim Ablauf des Programms erlaubt die entsprechende Interruptroutine dem Programmierer dann, Registerinhalte zu untersuchen und zu verändern. Damit läßt sich exakt feststellen, welche Auswirkung einzelne Befehle haben. Bei der Wiederaufnahme der Programmausführung setzt der Monitor (oder das Fehlersuchprogramm) die am Unterbrechungspunkt dargestellten Befehle wieder ein und läßt das Programm weiterlaufen.

Der 6809 besitzt drei unterschiedliche Interruptmechanismen: IRQ (Interrupt Request – einfache Interruptanforderung), FIRQ (First Interrupt Request – schnelle Interruptanforderung) und NMI (Non-Maskable Interrupt – nicht maskierbarer Interrupt). Alle drei werden durch den Empfang des entsprechenden Signals auf drei Kontakten des Prozessorchips aktiviert. Der Balken über dem Namen (beispielsweise in $\overline{\text{IRQ}}$) zeigt an, daß sie beim Prozessor von einem Nullsignal und nicht durch eine Eins aktiviert werden. Diese drei Kontakte sind mit dem Hauptbus verbunden, damit Peripheriechips wie der 6820 und 6850 ihre Ausgabekontakte auf die gleichen Leitungen legen können. Durch die Programmierung der Chips können die Interrupts ermöglicht werden, so daß die entsprechenden Signale automatisch gesandt werden.

Weiterhin gibt es drei Softwareinterrupts mit den Codes SWI1, SWI2 und SWI3.

Beim Auftreten eines Interrupts wird die Steuerung an eine Vektoradresse übergeben, die in einer speziellen Speicherstelle an der oberen Speichergrenze liegt. Diese Vektoradressen befinden sich normalerweise im ROM, so daß die Steuerung immer an die gleiche feste Adresse übergeben wird, die jedoch im RAM liegt. Sie enthält einen Sprungbefehl, mit dem der Anwender eine eigene Routine ansprechen kann. Die Speicherstellen sind:

Interrupttyp	Vektor
$\overline{\text{NMI}}$	$FFFC
SWI	$FFFA
$\overline{\text{IRQ}}$	$FFF8
$\overline{\text{FIRQ}}$	$FFF6
SW12	$FFF4
SW13	$FFF2

Interessant ist hierbei, daß die beiden höchsten Speicherbytes $FFFE und $FFFF den „Reset Vektor" enthalten, also die Adresse, an die die Steuerung beim Einschalten des Systems oder bei einem Hardware-Reset übergeben wird. Im Normalfall liegt dort die Anfangsadresse des ROM-Monitors.

Drei Bits des Condition Code Registers liefern Informationen über die Interrupts: Bit 4 (I),

Bit 6 (F) und Bit 7 (E). Das Setzen von Bit I maskiert den $\overline{\text{IRQ}}$ Interrupt und Bit F den $\overline{\text{FIRQ}}$. Mit dem E-Bit kann der Prozessor zwischen $\overline{\text{IRQ}}$, $\overline{\text{NMI}}$ und $\overline{\text{FIRQ}}$ unterscheiden. Steht es auf Eins, ist ein $\overline{\text{IRQ}}$ oder ein $\overline{\text{NMI}}$ eingetreten, bei Null ein $\overline{\text{FIRQ}}$.

Ein Interrupt wird wie ein Subroutinenaufruf behandelt: Der Inhalt einiger oder aller Register wird auf den Stack geschoben, so daß die Steuerung an die gleiche Stelle zurückgegeben werden kann, an der das Programm unterbrochen wurde. Ähnlich wie RTS wird die Interruptroutine mit einem RTI-Befehl beendet, der die Register vom Stack zieht und die Steuerung an das Programm zurückgibt.

Der Unterschied zwischen $\overline{\text{FIRQ}}$ und den anderen Interrupts liegt darin, daß $\overline{\text{FIRQ}}$ nur den Befehlszähler und das Condition Code Register auf den Stack schiebt und daher weitaus schneller ist. Da die Interruptroutinen alle eingesetzten Register wieder in den ursprünglichen Zustand versetzen müssen, sollte dieser Interrupttyp nicht mit Routinen eingesetzt werden, die mehr als zwei Register benötigen. Hier wird auch die Aufgabe des E-Bits deutlich: Da RTI beide Routinen – $\overline{\text{IRQ}}$ und $\overline{\text{FIRQ}}$ – beendet, kann der Prozessor mit dem E-Bit feststellen, welche Register vom Stack gezogen werden müssen. Ein Interrupt läuft folgendermaßen ab:

1) Der laufende Befehl wird ausgeführt.
2) Das I-Bit wird gesetzt, und alle $\overline{\text{IRQ}}$-Interrupts werden abgeschaltet. Bei einem $\overline{\text{FIRQ}}$ oder $\overline{\text{NMI}}$ wird weiterhin Bit F gesetzt, um auch $\overline{\text{FIRQ}}$ abzuschalten. SWI1 kann andere Interrupts maskieren, SWI2 und SWI3 jedoch nicht.
3) Bei $\overline{\text{FIRQ}}$ wird Bit E auf Null gesetzt, in allen anderen Fällen auf Eins.
4) Der Vektor in der entsprechenden Speicherstelle wird in den Befehlszähler geladen und die Steuerung an diese Adresse übergeben.

Unser erstes Programm zeigt ein weiteres Einsatzgebiet für Interrupts: die Steuerung einer Echtzeituhr. Wir setzen voraus, daß ein Zeitmechanismus – beispielsweise ein Spezialchip wie der 6840 Interval Timer, eine Unterteilung der Systemuhr oder eine Modifikation der 50-Hz-Stromversorgung – bei $5000 mit der PIA verbunden ist. Die erste Subroutine ermöglicht die Interrupts und richtet bei $50 einen 16-Bit-Zähler ein. Die Interruptroutine inkrementiert diesen Zähler, so daß wir bei jeder Untersuchung von $50 die Anzahl der Zeitsignale erhalten, die von einem bestimmten Zeitpunkt an empfangen wurden. Aus diesen Informationen und der Frequenz der Impulse kann die aktuelle Zeit berechnet werden.

Unser zweites Programm nimmt an, daß bei $E000 ein Drucker mit der gleichen PIA angeschlossen ist. Wir verwenden einen Buffer mit unbegrenzter Kapazität (bei $100), in den die Hilfsroutine eine Ausgabezeile speichert. Während des Druckes wird bei $50 ein Flag auf Null gesetzt und nach Ende des Druckes auf Eins. Dadurch weiß eine weitere Routine (auf die wir hier nicht weiter eingehen), wann sie den Buffer füllen soll. $51 und $52 enthalten einen Zeiger für die Adresse des nächsten Bufferzeichens, das gedruckt werden soll. Die erste Subroutine richtet die PIA, das Flag und den Buffer auf eine neue Zeile aus.

Erstes Programm

```
PIACR   EQU   $E001    PIA-Steuerregister
PIADR   EQU   $E000    PIA-Datenregister
INTRP   EQU   $2000
CLK1    EQU   $50
CLK2    EQU   $51
        ORG   $1000    Subroutine zur Initialisierung der Uhr
INITCK  CLR   CLK1     Uhrspeicher löschen
        CLR   CLK2
        LDA   #%00000101   PIA-Interrupts ermöglichen
        STA   PIACR
        ANDCC #%11101111   IRQ ermöglichen
WAITCK  TST   CLK2     Auf erste Inkrementierung warten
        BEQ   WAITCK
        RTS
        ORG   INTRP    Interruptroutine
        LDA   PIADA    Interrupt löschen
        LDD   CLK1     Zähler holen
        ADDD  #1       Zähler inkrementieren
        STD   CLK1
        RT1
```

Zweites Programm

```
PIACR   EQU   $E001    PIA-Steuerregister
PIADR   EQU   $E000    PIA-Datenregister
INTRP   EQU   $2000
CR      EQU   13       Wagenrücklauf
BUFFER  EQU   $100     Bufferadresse
BUFPTR  EQU   $51      Bufferzeiger
FLAG    EQU   $50      Flag für Zeilenende
        ORG   $1000    Subroutine für Initialisierung
        CLR   FLAG     Flag für Zeilenende löschen
        LDX   #BUFFER  Zum Anlegen des Buffers Bufferzeiger
                       initialisieren
        STX   BUFPTR
        CLR   PIACR    Adreßdaten-Richtungsregister
        LDA   $FF      Alle Leitungen auf Ausgabe stellen
        STA   PIADR
        LDA   #%00000101   PIA-Interrupts ermöglichen
        STA   PIACR
        ANDCC #%11101111   IRQ ermöglichen
        RTS
        ORG   INTRP    Interruptroutine
        LDX   BUFPTR   Bufferzeiger
        LDA   ,X+      Nächstes Zeichen aus Buffer holen
        STA   PIADR    Zeichen drucken
        LDB   PIADR    Interrupt löschen
        STX   BUFPTR   Bufferzeiger inkrementieren
        CMPA  #CR      Ist dies das Zeilenende?
        BNE   FINISH   Überspringen falls nicht
        INC   FLAG     Sonst Flag setzen
FINISH  RTI
```

Im Zauberwald

Im Verlauf der letzten Artikel zur Erstellung des Haunted-Forest-Programms haben wir Ihnen die Grundlagen zur Entwicklung von Abenteuerspielen gezeigt. Das nachfolgend gezeigte Listing wird Ihnen beim Zusammenfügen der einzelnen Routinen helfen.

Haunted Forest ist ein kurzes Abenteuerspiel, das Ihnen die Techniken der Programmierung derartiger Spiele zeigen sollte. Die Struktur von Haunted Forest zeigt, wie ein Adventure, basierend auf einer Karte der einzelnen Orte und deren Beziehungen zueinander, aufgebaut werden kann. Diese Karte ist immer die Basis, egal, ob es 16 oder 600 Orte gibt. Außerdem müssen die Hintergrundgeschichte und die „Hauptgefahren" vor Beginn der Programmierung ausgearbeitet werden. Der beste Weg ist oft, die Karte auf Rechenpapier zu zeichnen, um Ideen und Regeln in die Wirklichkeit umsetzen zu können. Ist das Konzept ausgearbeitet, kann mit der Programmierung begonnen werden.

Die erste Aufgabe ist normalerweise, die zweidimensionale Karte in Array-Elemente umzuwandeln. Bei Haunted Forest werden die Ortsdaten in zwei eindimensionalen Arrays gespeichert. Sie umfassen die Ortsbeschreibungen und eine codierte Liste möglicher Ausgänge für jeden Ort. Die im Spiel vorkommenden Objekte werden unter Angabe einer Beschreibung und des aktuellen Ortes in einem zweidimensionalen Array aufbewahrt. So entsteht eine einfache Datenbank, die die Abenteuerwelt repräsentiert. Jetzt können einfache Routinen entwickelt werden, die dem Spieler Bewegungen und das Aufnehmen oder Ablegen von Objekten ermöglichen. Die speziellen Regeln des Spieles sind jedoch größtenteils noch undefiniert, abgesehen von den Objekten, die dem Spieler ermöglichen, mit speziellen Aufgaben fertig zu werden.

Commodore-Version

Diese Version des Programms wurde auf einem Commodore 64 geschrieben. Sie wird jedoch auf den meisten Computern funktionieren, die mit ähnlichen BASIC-Dialekten arbeiten. Eventuell müssen in bezug auf Zufallszahlen oder Löschen des Bildschirms kleinere Anpassungen vorgenommen werden. Für den Acorn B geben wir gezielte Hinweise.

Aufgrund des speziellen Stringhandlings des Spectrum können die notwendigen Änderungen hierfür nicht dargestellt werden. Lesen Sie daher noch einmal die letzten Artikel, und schauen Sie zusätzlich in Ihr Handbuch, um Unklarheiten zu beseitigen. Nachfolgend finden Sie eine Umwandlungstabelle der verwendeten String-Variablennamen.

Da der Reiz eines Abenteuerspiels darin liegt, Gefahren zu meistern und Geheimnisse aufzuspüren, wird durch eine Analyse des Spieles dem Spieler dieser Reiz genommen. Zur Demonstration mußten jedoch die internen Vorgänge von Haunted Forest gezeigt werden. Das Programm Digitaya dagegen wurde parallel entwickelt, so daß der Spielablauf noch unbekannt ist. Im nächsten Artikel finden Sie ein komplettes Listing.

Umwandlungstabelle der String-Variablen für den Spectrum

Microsoft	Spectrum	Zweck der Variable
LN$()	L$()	Ortsbeschreibung
EX$()	E$()	Ausgänge
IC$()	I$()	Objekte beim Spieler
IV$(,)	V$(,)	Objekte im Spiel
SN$	S$	Speichert Satz zu Formatierung
OW$	O$	„Altes Wort" in Formatier-Routine
NW$	N$	„Neues Wort" in Formatier-Rout.
EX$	X$	Ausgänge von aktuellem Ort
IS$	I$	Aktuelle Anweisung
DR$	D$	Angegebene Richtung
NN$	R$	Objekt der Anweisung
VB$	B$	Präposition der Anweisung
CD$	C$	Code-Wort

BASIC-Dialekte

Acorn B
Die folgenden Zeilen sollten im Listing von Haunted Forest ausgetauscht werden:

```
207 RND (-TIME)
210 P=RND (10)
1160 CLS
4190 REPEAT: A$=GET$:UNTIL A$="Y" OR A$="N"
4535 CR=RND (3)
6067 CC=0
```

Haunted Forest (vollständig)

```
130 REM ** HAUNTED **
140 REM ** FOREST  **
180 :
200 GOSUB6000:REM READ ARRAY DATA
205 GOSUB 1000:REM STORY SO FAR
207 R=RND(-1)
210 P=INT(RND(TI)*10+1):REM START POINT
220 :
230 REM **** MAIN LOOP STARTS HERE ****
240 MF=0:REM MOVE FLAG
245 PRINT
250 GOSUB2000:REM DESCRIBE POSITION
255 GOSUB2300:REM DESCRIBE EXITS
257 GOSUB2700:REM IS P SPECIAL ?
258 IF SF=1 THEN 300:REM NEXT INSTRUCTION
260 PRINT:INPUT"INSTRUCTIONS";IS$
270 GOSUB2500:REM SPLIT INSTRUCTION
275 IF F=0 THEN 260:REM INVALID INSTRUCTION
280 GOSUB3000:REM NORMAL COMMANDS
290 IF VF=0 THENPRINT:PRINT"I DONT UNDERSTAND"
300 IF MF=1 THEN 240:REM NEW LOCATION
310 IF MF=0 THEN 260:REM NEW INSTRUCTION
320 :
990 END
1000 REM **** STORY SO FAR S/R ****
1010 SN$="WELCOME TO THE HAUNTED FOREST"
1020 GOSUB5500:REM FORMAT
1030 PRINT
1040 SN$="AS YOU AWAKE FROM A DEEP SLEEP, THE "
1050 SN$=SN$+"FOREST FLOOR FEELS SOFT AND DRY. "
1060 SN$="YOU DO NOT KNOW HOW YOU CAME TO BE HERE "
1070 SN$=SN$+"BUT KNOW THAT YOU MUST FIND THE "
1080 SN$=SN$+"VILLAGE ON THE EDGE OF THE WOOD TO "
1090 SN$=SN$+"REACH SAFETY."
1100 GOSUB5500:REM FORMAT
1110 PRINT
1120 SN$="YOU LOOK AROUND, TRYING TO GET YOUR BEARINGS."
1130 GOSUB5500:REM FORMAT
1140 PRINT:PRINT"PRESS ANY KEY TO START"
1150 GET A$:IF A$="" THEN 1150
1160 PRINTCHR$(147):REM CLEAR SCREEN
1170 RETURN
1180 :
2000 REM **** DESCRIBE LOCATION ****
2010 SN$="YOU ARE "+LN$(P):GOSUB5500
2020 SN$="YOU SEE "
2030 REM ** CHECK INVENTORY FOR OBJ **
2040 F=0:SP$=""
2050 FOR I=1 TO 3
2060 IF VAL(IV$(I,2))<>P THEN 2080
2070 SN$=SN$+SP$+"A "+IV$(I,1):F=1:SP$=", "
2080 NEXT I
2090 IF F=0 THEN SN$=SN$+"NO OBJECTS"
2100 GOSUB5500:REM FORMAT OUTPUT
2110 RETURN
2120 :
2299 :
2300 REM **** DESCRIBE EXITS S/R ****
2310 EX$=EX$(P)
2320 NR=VAL(LEFT$(EX$,2))
2330 EA=VAL(MID$(EX$,3,2))
2340 SO=VAL(MID$(EX$,5,2))
2350 WE=VAL(RIGHT$(EX$,2))
2353 :
2355 IF(NR OR EA OR SO OR WE)=0 THEN RETURN
2360 PRINT:SN$="EXITS ARE TO THE "
2370 IF NR <>0 THEN SN$=SN$+"NORTH "
2380 IF EA <>0 THEN SN$=SN$+"EAST "
2390 IF SO <>0 THEN SN$=SN$+"SOUTH "
2400 IF WE <>0 THEN SN$=SN$+"WEST "
2410 GOSUB 5500:REM FORMAT
2415 PRINT
2420 RETURN
2430 :
2500 REM **** SPLIT COMMAND S/R ****
2510 IF IS$="LIST" OR IS$="END" THEN VB$=IS$:F=1:RETURN
2515 IF IS$="LOOK" THEN VB$=IS$:F=1:RETURN
2520 F=0
2530 LS=LEN(IS$)
2540 FOR C=1 TO LS
2550 A$=MID$(IS$,C,1)
2560 IF A$<>" " THEN 2590
2570 VB$=LEFT$(IS$,C-1):F=1
2580 NN$=RIGHT$(IS$,LS-C):C=LS
2590 NEXT C
2600 :
2610 IF F=1 THEN RETURN
2620 PRINT:PRINT"I NEED AT LEAST TWO WORDS"
2630 RETURN
2650 :
2700 REM **** IS P SPECIAL S/R ****
2705 SF=0:REM UNSET SPECIAL FLAG
2707 REM ** RANDOM GHOST **
2710 IF P>4 AND RND(1)<0.1 THEN GOSUB 4290:RETURN
2715 :
2716 REM ** OTHER SPECIAL LOCATIONS **
2720 ON P GOSUB4590,4870,5100,4590
2730 RETURN
2735 :
3000 REM **** NORMAL COMMANDS S/R ****
3010 VF=0:REM VERB FLAG
3020 IF VB$="GO" OR VB$="MOVE" THENVF=1:GOSUB3500
3030 IF VB$="TAKE" OR VB$="PICK"THEN VF=1:GOSUB3700
3040 IF VB$="DROP" OR VB$="PUT"THEN VF=1:GOSUB3900
3050 IF VB$="LIST" OR VB$="INVENTORY"THEN VF=1:GOSUB4100
3055 IF VB$="LOOK" THEN VF=1:MF=1:RETURN
3060 IF VB$="END" OR VB$="FINISH" THEN VF=1:GOSUB4170
3070 RETURN
3080 :
3500 REM **** MOVE S/R ****
3505 GOSUB3630:REM SEARCH FOR DIRECTION
3510 MF=1:REM SET MOVE FLAG
3520 DR$=LEFT$(NN$,1)
3530 IF DR$<>"N"ANDDR$<>"E"ANDDR$<>"S"ANDDR$<>"W"THENGOTO3590
3540 IF DR$="N"AND NR<>0 THEN P=NR:RETURN
3550 IF DR$="E"AND EA<>0 THEN P=EA:RETURN
3560 IF DR$="S"AND SO<>0 THEN P=SO:RETURN
3570 IF DR$="W"AND WE<>0 THEN P=WE:RETURN
3580 PRINT:PRINT"YOU CAN'T ";IS$
3585 MF=0:RETURN
3590 REM ** NOUN NOT DIRECTION **
3600 PRINT"WHAT IS ";NN$;" ?"
3610 MF=0:RETURN
3620 :
3630 REM **** SEARCH FOR DIRECTION S/R ****
3640 NN$=NN$+" ":LN=LEN(NN$):C=1
3645 FOR I=1 TO LN
3650 IF MID$(NN$,I,1)<>" " THEN NEXT I:RETURN
3655 W$=MID$(NN$,C,I-C):C=I+1
3660 IF W$="NORTH" OR W$="EAST" THEN NN$=W$:I=LN
3665 IF W$="SOUTH" OR W$="WEST" THEN NN$=W$:I=LN
3670 NEXT I
3675 RETURN
3700 REM **** TAKE S/R ****
3710 GOSUB 5300:REM IS OBJECT VALID
3720 IF F=0 THEN SN$="THERE IS NO "+W$:GOSUB5500:RETURN
3730 OV=F:GOSUB5450:REM CHECK INVENTORY
3740 IF HF=1 THEN SN$="YOU ALREADY HAVE THE "+IV$(F,1):GOSUB5500:RETURN
3750 :
3755 REM ** IS OBJECT HERE ? **
3760 IF VAL(IV$(F,2))<>P THEN SN$=IV$(F,1)+" IS NOT HERE":GOSUB5500:RETURN
3770 :
3780 REM ** ADD OBJECT TO LIST **
3790 A=0
3800 FOR J=1 TO 2
3810 IF IC$(J)="" THEN IC$(J)=IV$(F,1):AF=1:J=2
3820 NEXT J
3830 :
3840 REM ** FULL QUOTA **
3850 IF AF=0 THEN PRINT"YOU ALREADY HAVE TWO OBJECTS":RETURN
3860 :
3870 SN$="YOU TAKE THE "+IV$(F,1):GOSUB5500
3880 IV$(F,2)="-1":REM DELETE INVENTORY ENTRY
3890 RETURN
3895 :
3900 REM **** DROP S/R ****
3910 GOSUB5300:REM VALID OBJECT
3920 IF F=0 THEN SN$="THERE IS NO "+W$:GOSUB5500:RETURN
3930 :
3940 REM ** IS OBJECT IN CARRIED INVENTORY **
3950 OV=F:GOSUB5450
3960 IF HF=0 THEN SN$="YOU DO NOT HAVE THE "+IV$(F,1):GOSUB5500:RETURN
3970 :
3980 REM ** DROP OBJECT **
3990 SN$="YOU DROP THE "+IV$(F,1):GOSUB5500
4000 IV$(F,2)=STR$(P):REM MAKE ENTRY IN INVENTORY
4010 :
4020 REM ** DELETE OBJECT FROM CARRIED INVENTORY **
4030 FOR J=1TO2
4040 IF IC$(J)=IV$(F,1) THEN IC$(J)="":J=2
4050 NEXT J
4060 RETURN
4070 :
4100 REM **** LIST CARRIED INVENTORY ****
4110 PRINT"OBJECTS HELD:"
4120 FOR I=1 TO 2
4130 PRINT"  ";IC$(I)
4140 NEXT I
4150 RETURN
4160 :
4170 REM **** END GAME S/R ****
4180 PRINT:PRINT"ARE YOU SURE (Y/N) ?"
4190 GET A$:IF A$<>"Y" AND A$<>"N" THEN 4190
4200 IF A$="N" THEN RETURN
4210 END
4220 :
4290 REM **** RANDOM GHOST S/R ****
4295 SF=1:GC=0
4300 SN$="YOU FEEL A COLD SENSATION RUNNING THE LENGTH"
4305 SN$=SN$+" OF YOUR SPINE. SUDDENLY A WHITE APPARITION"
4310 SN$=SN$+" APPEARS FROM OUT OF THE TREES AND"
4315 SN$=SN$+" MOVES TOWARDS YOU":GOSUB5500:REM FORMAT
4320 :
4325 SN$="THE GHOST MOVES CLOSER":GOSUB5500
4330 GC=GC+1:IF GC>4 THEN GOSUB4455:REM
4335 PRINT:INPUT"INSTRUCTIONS";IS$
4340 GOSUB2500:REM SPLIT INSTRUCTION
4345 IF F=0 THEN 4325:REM NEXT INSTRUCTION
4350 OP=P:GOSUB3000:REM ANALYSE INSTRUCTION
4355 IF MF=1 AND VB$="GO"THEN GOSUB4400:GOTO 4325
4357 IF MF=1 AND VB$="LOOK" THEN GOSUB2000:GOSUB2300:GOTO4325
4360 IF VF=1 THEN 4325:REM NEXT INSTRUCTION
4365 REM ** NEW INSTRUCTION WORDS **
4370 IF VB$="KILL" OR VB$="FIGHT" THEN GOSUB4425:GOTO 4325
4375 :
4385 IF VB$="SING" THEN GOSUB4500:RETURN
4390 SN$="I DON'T UNDERSTAND":GOSUB5500:GOTO4325
4395 :
4400 REM ** ATTEMPT TO MOVE **
4405 SN$="YOU ARE TRANSFIXED WITH TERROR AND CANNOT"
4410 SN$=SN$+" MOVE...YET":MF=0:GOSUB5500:P=OP
4415 RETURN
4420 :
4425 REM ** FIGHT OR KILL **
4430 SN$="THE GHOST IS A BEING OF THE SUPERNATURAL"
4435 SN$=SN$+" AND LAUGHS AT YOUR FEEBLE ATTEMPTS"
4440 SN$=SN$+" TO INJURE HIM":GOSUB5500
4445 RETURN
4450 :
4455 REM ** DEATH **
4460 SN$="THE PAIN IN YOUR CHEST BECOMES UNBEARABLE"
4465 SN$=SN$+" AND YOU SLUMP ONTO THE LEAFY FOREST FLOOR.":GOSUB5500
```

```
4470 SN$="YOUR SPIRIT RISES FROM YOUR INERT BODY"
4475 SN$=SN$+" AND YOU FLOAT AWAY INTO THE MIST TO JOIN"
4480 SN$=SN$+" THE OTHER TORMENTED SOULS OF THE"
4485 SN$=SN$+" HAUNTED FOREST.":GOSUB5500
4490 END
4495 :
4500 REM ** SING **
4505 SN$="YOU KNOW THREE SONGS. WHICH ONE WILL YOU CHOOSE ?":GOSUB5500
4510 SN$="1) THE THEME FROM 'GHOSTBUSTERS'":GOSUB5500
4515 SN$="2) 'THERE'S A GHOST IN MY HOUSE'":GOSUB5500
4520 SN$="3) 'WAY DOWN UPON THE SWANEE RIVER'":GOSUB5500
4525 PRINT:INPUT"MAKE YOUR CHOICE";C$
4530 IF VAL(C$)>3 OR VAL(C$)<1 THEN PRINT:PRINT"INVALID":GOTO4525
4535 CR=INT(RND(1)*3)+1
4537 IF CR<>VAL(C$) THEN GOSUB4542:REM WRONG TUNE
4540 GOSUB4565:RETURN:REM CORRECT
4542 REM **** WRONG TUNE S/R ****
4545 SN$="THE GHOST HAS A PARTICULAR HATRED OF"
4550 SN$=SN$+" THAT TUNE AND LUNGES AT YOU.":GOSUB5500
4555 GOSUB 4455:REM DEATH
4560 :
4565 REM ** CORRECT TUNE **
4570 SN$="THE GHOST IS APPEASED BY YOUR RENDITION OF THE TUNE"
4575 SN$=SN$+" AND VAPOURISES INTO THIN AIR":GOSUB5500
4580 RETURN
4585 :
4590 REM **** TUNNEL ENTRANCE S/R ****
4600 SF=1
4605 SN$="YOU HAVE ARRIVED AT THE MOUTH OF A LARGE TUNNEL":GOSUB5500
4610 SN$="YOU CAN ENTER THE TUNNEL OR RETREAT ALONG THE PATH":GOSUB5500
4620 :
4625 PRINT:INPUT"INSTRUCTIONS";IS$
4630 GOSUB2500:REM SPLIT INSTRUCTION
4635 IF F=0 THEN 4625:REM INVALID INSTRUCTION
4637 GOSUB3000:REM NORMAL INSTRUCTIONS
4640 IF MF=1 THEN RETURN:REM PLAYER RETREATS
4645 IF VF=1 THEN 4625:REM INSTRUCTION OBEYED
4650 REM ** NEW INSTRUCTIONS **
4655 IF VB$="ENTER" THEN GOSUB 4700:RETURN
4660 IF VB$="RETREAT" AND P=4 THEN MF=1:P=6:RETURN
4665 IF VB$="RETREAT" AND P=1 THEN MF=1:P=9:RETURN
4667 SN$="I DON'T UNDERSTAND":GOSUB5500:GOTO 4625
4700 REM ** ENTER TUNNEL **
4705 SN$="YOU ENTER THE TUNNEL BUT IT IS TOO DARK TO"
4710 SN$=SN$+" FIND YOUR WAY.":GOSUB5500
4725 PRINT:INPUT"INSTRUCTIONS";IS$
4730 GOSUB2500:REM SPLIT INSTRUCTION
4732 :
4735 IF F=0 THEN 4725:REM INVALID INSTRUCTION
4740 OP=P:GOSUB3000:REM NORMAL INSTRUCTIONS
4745 IF MF=1THEN SN$="IT IS SO DARK THAT YOU CAN ONLY SEE":P=OP
4747 IF MF=1THENSN$=SN$+" THE TUNNEL ENTRANCE":GOSUB5500:MF=0:GOTO4725
4750 IF VF=1 THEN 4725:REM INSTRUCTION OBEYED
4755 IF VB$="RETREAT" AND P=4 THEN MF=1:P=6:RETURN
4760 IF VB$="RETREAT" AND P=1 THEN MF=1:P=9:RETURN
4762 IFVB$<>"USE"ANDVB$<>"LIGHT"THEN SN$="I DON'T UNDERSTAND"
4765 IFVB$<>"USE"ANDVB$<>"LIGHT"THEN GOSUB5500:GOTO4725
4777 :
4780 REM ** SEARCH FOR LAMP **
4790 GOSUB5300:REM VALID OBJECT ?
4795 OV=F:GOSUB5450:REM IS OBJECT HELD ?
4797 IF F=0 THEN SN$="THERE IS NO "+W$:GOSUB5500:GOTO4725
4800 IF HF=0 THEN SN$="YOU DO NOT HAVE THE "+IV$(F,1):GOSUB5500:GOTO4725
4810 REM ** IS OBJECT LAMP ? **
4815 IF F<>2 THEN SN$="THE "+IV$(F,1)+" IS NO USE":GOSUB5500:GOTO4725
4835 REM ** SUCCESS **
4840 SN$="YOU USE THE LAMP TO LIGHT YOUR WAY THROUGH THE TUNNEL"
4845 SN$=SN$+" AND EVENTUALLY EMERGE FROM THE EXIT.":GOSUB5500
4850 IF P=1 THEN MF=1:P=4:RETURN
4855 IF P=4 THEN MF=1:P=1:RETURN
4860 :
4870 REM **** SWAMP S/R ****
4875 SF=1
4880 SN$="YOU START TO SINK INTO THE SWAMP.":GOSUB5500
4885 PRINT:INPUT"INSTRUCTIONS";IS$
4890 GOSUB2500:REM SPLIT INSTRUCTION
4895 IF F=0 THEN 4885:REM INVALID
4900 GOSUB3000:REM NORMAL INSTRUCTIONS
4910 IF VB$="LOOK" THENGOSUB2000:GOTO4885
4915 IF VB$="DROP"THEN IV$(F,2)="-2":REM OBJ LOST FOREVER
4917 IF VF=1 THEN 4885:REM NORMAL COMMAND
4920 REM ** NEW COMMANDS **
4925 IF VB$<>"SWIM"THEN SN$="I DON'T UNDERSTAND":GOSUB5500:GOTO4885
4930 REM ** SWIM **
4932 F=0
4935 FOR I=1 TO 2
4940 IF IC$(I)<>""THEN F=F+1
4950 NEXT I
4955 IF F<2 THENGOSUB5035:RETURN:REM SWIM AWAY
4960 GOSUB 5000:RETURN:REM TWO OBJS HELD
5000 REM **** TWO OBJECTS HELD S/R ****
5010 SN$="THE OBJECTS ARE WEIGHING YOU DOWN AND YOU ARE SINKING.":GOSUB5500
5012 PRINT:INPUT"INSTRUCTIONS";IS$
5015 GOSUB2500:REM SPLIT INSTRUCTION
5020 IF VB$<>"DROP" THENGOSUB5080:REM SINK
5025 GOSUB3900:IV$(F,2)="-2":REM DROP OBJ
5030 IF HF=0 OR F=0 THEN 5080:REM SINK
5035 REM **** SWIM AWAY ****
5040 SN$="YOU CAN NOW SWIM THROUGH THE SWAMP. WHICH WAY WILL YOU GO?"
     :GOSUB5500
5050 EX$(2)="00080605":GOSUB2300:REM DEFINE AND DISPLAY EXITS
5055 PRINT:INPUT"INSTRUCTIONS";IS$
5060 GOSUB2500:REM SPLIT INSTRUCTION
5062 IF F=0 THEN 5055:REM INVALID
5065 GOSUB3500:REM MOVE
5067 EX$(2)="00000000":REM ZERO EXIT DATA
5070 RETURN
5075 :
5080 REM **** SINK S/R ****
5085 SN$="YOU SINK INTO THE SWAMP AND DROWN":GOSUB5500
5090 END
5100 REM **** VILLAGE S/R ****
5102 SF=1
5105 SN$="THE VILLAGE IS SURROUNDED BY A TALL WALL.":GOSUB5500
5106 IF GF<>0 THEN GOSUB5190:RETURN:REM GATE
5107 SN$="A GUARD IS AT THE ENTRANCE GATE TO THE VILLAGE":GOSUB5500
5115 PRINT:INPUT"INSTRUCTIONS";IS$
5120 GOSUB2500:IF F=0 THEN 5115:REM INVALID
5125 GOSUB3000:REM NORMAL INSTRUCTIONS
5130 IF VB$="LOOK"THEN GOSUB2000:REM DESCRIBE
5135 IF VB$="GO" AND MF=1 THEN RETURN
5140 IF VF=1 THEN 5115:REM NEXT INSTRUCTION
5145 IF VB$<>"KILL"THENSN$="I DON'T UNDERSTAND":GOSUB5500:GOTO5115
5150 REM ** KILL **
5155 SN$="WHAT WILL YOU USE TO KILL THE GUARD?":GOSUB5500
5160 SN$="ENTER OBJECT OR <I> FOR INSTRUCTION":GOSUB5500
5162 INPUT IS$:IF IS$="I" THEN 5115
5165 GOSUB2500:REM SPLIT
5167 IF F=0 THEN 5160:REM INVALID
5170 GOSUB5300:IF F=0 THEN SN$="THERE IS NO "+W$:GOSUB5500:GOTO5160
5172 OV=F:GOSUB5450:REM IS OBJECT HELD
5174 IF HF=0 THEN SN$="YOU DO NOT HAVE THE "+IV$(F,1):GOSUB5500:GOTO5160
5175 IF F<>1 THEN SN$="THE "+IV$(F,1)+" IS NO USE":GOSUB5500:GOTO5160
5180 SN$="YOU KILL THE GUARD":GOSUB5500:GF=1
5185 :
5190 REM **** LOCKED GATE S/R ****
5195 SN$="YOU MOVE FORWARD AND TRY TO OPEN THE GATE TO THE VILLAGE"
5200 SN$=SN$+" BUT THE GATE IS LOCKED AND WILL NOT MOVE":GOSUB5500
5205 PRINT:INPUT"INSTRUCTIONS";IS$
5210 GOSUB2500:IF F=0 THEN 5205:REM INVALID
5215 GOSUB3000:REM NORMAL INSTRUCTIONS
5220 IF VB$="LOOK"THEN GOSUB2000:REM DESCRIBE
5225 IF VB$="GO" AND MF=1 THEN RETURN
5230 IF VF=1 THEN 5205:REM NEXT INSTRUCTION
5232 IF VB$="USE"THEN 5240
5234 IF VB$="UNLOCK"THENSN$="HOW?":GOSUB5500:GOTO5205
5235 SN$="I DON'T UNDERSTAND":GOSUB5500:GOTO5205
5240 GOSUB5300:REM VALID OBJECT
5242 OV=F:GOSUB5450:REM IS OBJ CARRIED
5244 IF F=0 THEN SN$="THERE IS NO "+W$:GOSUB5500:GOTO5205
5246 IF HF=0 THEN SN$="YOU DO NOT HAVE THE "IV$(F,1):GOSUB5500:GOTO5205
5248 IF F<>3 THEN SN$="THE "+IV$(F,1)+" IS NO USE":GOSUB5500:GOTO5205
5250 REM ** THROUGH GATE AND SAFE **
5255 SN$="YOU UNLOCK THE GATE, AND DISGUISING YOURSELF IN THE DEAD"
5260 SN$=SN$+" GUARD'S CLOTHES, WALK UNNOTICED THROUGH THE VILLAGE"
5265 SN$=SN$+" AND THE SAFETY OF THE OUTSIDE WORLD.":GOSUB5500
5270 END
5295 :
5300 REM **** VALID OBJECT S/R ****
5310 NN$=NN$+" ":LN=LEN(NN$):C=1:F=0
5315 FOR K=1 TO LN
5320 IF MID$(NN$,K,1)<>" " THEN NEXT K:RETURN
5325 W$=MID$(NN$,C,K-C):C=K+1
5330 LW=LEN(W$)
5335 FOR J=1 TO 3
5340 LI=LEN(IV$(J,1)):REM LENGTH OF OBJECT
5350 FOR I=1 TO LI-LW+1
5360 IF MID$(IV$(J,1),I,LW)=W$ THEN F=J:I=LI:J=3:K=LN
5370 NEXT I,J,K
5380 RETURN
5390 :
5450 REM **** IS OBJECT HELD S/R ****
5460 HF=0
5470 IF IV$(OV,2)="-1" THEN HF=1
5480 RETURN
5490 :
5500 REM **** FORMAT OUTPUT S/R ****
5510 LC=0:    REM CHAR/LINE COUNTER
5520 OC=1:    REM OLD COUNT INITIAL VALUE
5530 OW$="":  REM OLD WORD  INITIAL VALUE
5540 LL=40:   REM LINE LENGTH
5550 SN$=SN$+" DUMMY "
5560 PRINT
5570 FOR C=1 TO LEN(SN$)
5580 LC=LC+1
5590 IF MID$(SN$,C,1)=" " THEN GOSUB5800
5600 NEXT C
5605 PRINT
5610 RETURN
5620 :
5800 REM ** END OF LINE CHECK S/R **
5810 NW$=MID$(SN$,OC,C-OC+1):REM NEW WORD
5820 IF LC<LL THENPRINTOW$;:GOTO5840
5830 PRINTOW$:LC=LEN(NW$)
5840 OC=C+1:OW$=NW$
5850 RETURN
6000 REM **** READ OBJ & MAP DATA ****
6010 DIM IV$(3,2),LN$(10),EX$(10),IC$(2)
6020 FOR C=1 TO 3
6030 READ IV$(C,1),IV$(C,2)
6040 NEXT C
6050 :
6060 FOR C=1 TO 10
6065 READ LN$(C),EX$(C)
6070 CC=CC+VAL(EX$(C)):REM CHECKSUM TOTAL
6080 NEXT C
6090 :
6100 READ CD:IFCD<>CC THENPRINT"CHECKSUM ERROR":STOP
6110 :
6120 REM ** OBJECT DATA **
6130 DATA GUN,10,LAMP,9,KEY,5
6140 :
6150 REM ** MAP DATA **
6160 DATA NEAR A TUNNEL ENTRANCE,00000800
6170 DATA IN A SWAMP,00000000
6180 DATA IN A VILLAGE,07000000
6190 DATA NEAR A TUNNEL ENTRANCE,05060000
6200 DATA ON A PATH,00020400
6210 DATA ON A PATH,02070004
6220 DATA ON A PATH,08000306
6230 DATA ON A PATH,09000702
6240 DATA ON A PATH,01100800
6250 DATA IN A CLEARING,00000009
6260 REM ** CHECKSUM DATA **
6270 DATA 32253121
6280 RETURN
```

Elegantes Design

Der italienische Konzern Olivetti hat als Hersteller „eleganter und leistungsfähiger" Büromaschinen einen so guten Ruf erworben, daß er heute, neben amerikanischen und japanischen Firmen, als einer der wichtigsten Anbieter im Büromaschinenbereich gilt.

Der Olivetti M20 ist ein 16-Bit-Rechner, der 1981 auf den Markt kam. Er arbeitet mit einem Zilog Z8000 und verwendet Olivettis PCOS-Betriebssystem. Vor einiger Zeit brachte Olivetti eine IBM-kompatible Version des Rechners auf den Markt.

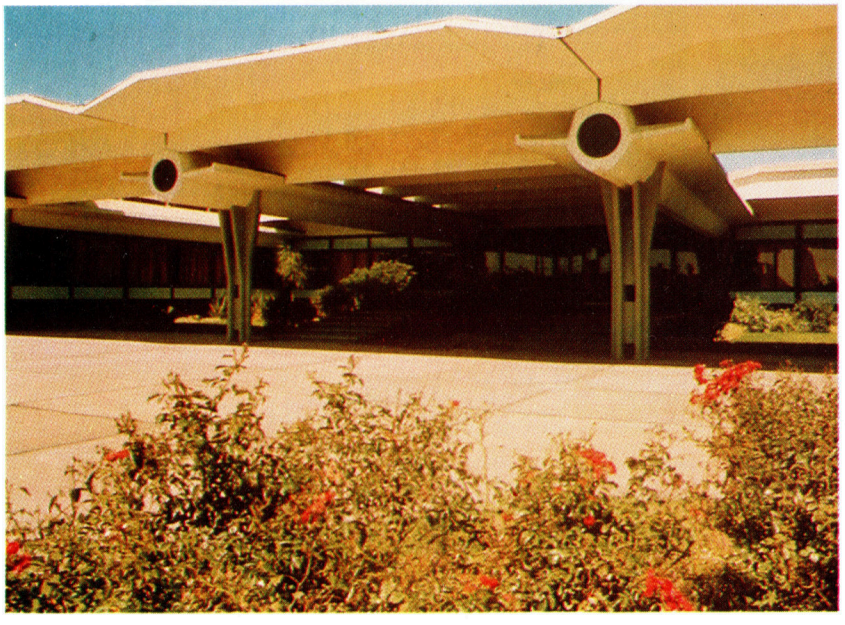
Olivetti hat sich einen Ruf wegen des ungewöhnlichen Produktdesigns erworben. Dieses Erscheinungsbild schlägt sich auch in der Architektur der Betriebsgebäude des Unternehmens nieder. Dieses Gebäude beispielsweise wurde 1959 errichtet und weist Merkmale auf, die von anderen Architekten später übernommen wurden.

Camillo Olivetti gründete sein Unternehmen 1908. Mit 20 Mitarbeitern eröffnete er ein Maschinengeschäft in Ivrea, damals eine kleine Stadt im ländlichen Norditalien, und begann mit der Produktion der ersten Schreibmaschine seines Hauses, der M1. Seinerzeit war die italienische Wirtschaft überwiegend landwirtschaftlich ausgerichtet. – Kaum eine Spur von jenen Schwerindustrien, die für Deutschland, England und die USA den wirtschaftlichen Aufschwung bedeuteten. Und Olivettis Produktion wuchs ständig. Wurden 1914 nur vier Maschinen pro Tag hergestellt, so waren es 1929 bereits 50.

In den 30er Jahren organisierte Camillos Sohn Adriano das Unternehmen um und stellte unter anderem Studenten der Olivetti-eigenen Abendschule ein, die 1924 gegründet worden war. Auch das Angebot an Sozialleistungen wurde erweitert – Sicherheit „von der Wiege bis zum Grabe" –, wie es später auch von den Japanern praktiziert wurde. Während weltweit die Industrie unter der Wirtschaftskrise litt, setzte sich Olivettis Wachstum fort. 1933 hatte das Unternehmen bereits 15 Millionen Büroprodukte verkauft. Im Jahr 1937 kam Olivettis erster Fernschreiber auf den Markt, 1940 der erste Rechner.

Der Zweite Weltkrieg bremste Olivettis Aufschwung. In den Nachkriegsjahren konzentrierte sich das Unternehmen auf die Entwicklung neuer Märkte. Der Erfolg der Firma lag im eleganten Styling und in der Qualität seiner Produkte begründet. Selbst IBM-Manager ließen verlauten, daß Olivetti-Produkte „wie ein wunderschönes Mosaik" zusammenpaßten.

In den 50er und 60er Jahren begann Olivetti, sich auf die Entwicklung von Bürocomputern zu konzentrieren. Eingeleitet wurde diese Entwicklung mit dem ersten, 1955 vorgestellten, Olivetti-Rechner. Nur wenige Jahre später kam der erste Großrechner namens Elea auf den Markt. Und Olivetti bewegte sich vom rein mechanischen Büromaschinengeschäft in den zukunftsträchtigen Elektronikmarkt.



Erfolgsunternehmen

Heute stellt die Firma eine breite Palette elektronischer Büromaschinen her. Das Unternehmen investiert auch große Summen in die Entwicklung neuer Software. 1982 war Olivetti zweitgrößter Computerhersteller in Europa (nur von IBM übertroffen). Sowohl der M10-Portable-Computer als auch die M20-Rechner verkaufen sich gut.

Der M10 Hand-held wiegt nur 1,7 Kilo und wird mit einem 8 x 40-Zeichen-Schirm geliefert. Der Rechner ist batteriebetrieben und verfügt über acht KByte RAM intern, die auf 64 KByte erweitert werden können. Der M20 (ein 16-Bit-Rechner) verfügt über einen Z8001-Prozessor, der sich bei anderen 16-Bit-Rechner-Herstellern keiner großen Beliebtheit erfreut. Er enthält überdies einen 8086-Prozessor, mit dem im gewissen Umfang Kompatibilität zu CP/M-86 und MS-DOS gewährleistet ist.

Darüber hinaus plant Olivetti einen neuen IBM-kompatiblen Rechner, der weniger kosten soll als das Original. M24 ist mit einem 8086-2-Prozessor ausgestattet sowie mit einer Z8001-Karte als Option, die ihn mit dem Olivetti M20 kompatibel macht.

Mittlerweile unterzeichnete Olivetti einen Vertrag mit AT&T (dem größten Telekommunikationsunternehmen der Welt) zwecks gemeinsamer Entwicklung eines Betriebssystems. Unzweifelhaft ist, daß Olivetti mit seinem weltweiten Händlernetz und seinen Forschungs- und Entwicklungsabteilungen seinen Ruf als Hersteller gut gestylter und hervorragend gefertigter Produkte ausbauen wird.

Förderkurse

Lern-Software für Schüler, denen Prüfungen bevorstehen, wird immer häufiger angeboten. Fast alle Programme sind auf der Frage-Antwort-Methode aufgebaut und beinhalten nur Fragestellungen, bei denen es eindeutige Antworten gibt.

Es gibt eine Fülle von Lern-Programmen auf dem Markt, denn sie sind nicht gerade schwierig oder aufwendig zu programmieren. Die Aufgabe des Programmierers besteht lediglich darin, Text zu präsentieren und eine Antwort vom Benutzer zu erfragen. Ist dieses Raster erst einmal gefunden, kann es beliebig modifiziert werden, da sich dieses Schema themenunabhängig überall verwenden läßt. Wenngleich solche Programme effektiv sein mögen, muß man zugeben, daß sie reichlich geistlos wirken. Künftige Generationen von Prüfungsvorbereitungs-Software werden sicherlich die Farb- und Grafikmöglichkeiten der Rechner besser nutzen. Bleibt abzuwarten, ob sie auch die anderen Möglichkeiten der Heimcomputer intensiver einsetzen.

Fremdsprachen

In England sind fast alle Fremdsprachen als Software erhältlich, die in die Abschlußprüfungen der Schulen einbezogen werden, sowohl für den weitverbreiteten Acorn B und den ZX Spectrum als auch für Rechner, die üblicherweise mehr zum Spielen verwendet werden wie etwa Atari-Computer.

Kosmos-Software liefert die Programme für den Acorn B und den ZX Spectrum auf Cassette. Da gibt es „French Mistress", den „German Master" und den „Spanish Tutor", die alle ähnlich aufgebaut sind und in zwei Schwierigkeitsgraden zur Verfügung stehen. Im ersten Schwierigkeitsgrad sind Wörterlisten und kurze Sätze bzw. Redewendungen (bis zu 59 Zeichen) in Englisch und der betreffenden Sprache enthalten, geordnet nach Kategorien wie „Familie", „Essen" oder „Lebewesen". Auf den Cassetten mit dem Schwierigkeitsgrad 2 sind Adverben und Adjektive enthalten. Zudem werden die Konjugation und die Deklination erklärt.

Der Benutzer kann wahlweise von der Fremdsprache zum Englischen oder umgekehrt vorgehen und überdies eigene Vokabularien erstellen, die sich auf Cassette speichern lassen. Die Cassetten sind in jeweils 16 „Lektionen" untergliedert, wobei der Maximalumfang 250 Wörter umfaßt. Diese Einschränkung gilt auch für die selbst gefertigten Wörterverzeichnisse.

Außerdem stehen weitere Wahlmöglichkeiten zur Verfügung: Entweder geht der Benutzer auf den Lern-Modus, in dem zunächst das fremdsprachige Wort und dann das englische Äquivalent dargestellt werden. Oder er entscheidet sich für den Selbsttest, in dem Wörter nur einsprachig zu sehen sind. Und schließlich gibt es einen zeitlimitierten Genauigkeitstest.

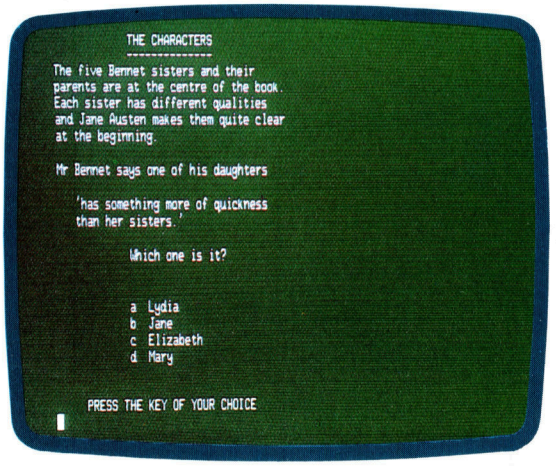
Muttersprache Englisch

Das Lernen der eigenen Sprache unterscheidet sich erheblich vom Lernen einer Fremdsprache. Ziel hierbei ist es, die Sprache optimal zu nutzen. Der Computer ist insofern hilfreich, als Grammatikregeln vermittelt werden können. Rechtschreibung, Nacherzählungen und Aufsätze sind aber eine Sache persönlicher Beurteilung und folglich nicht nach festgeschriebenen Regeln zu erlernen.

Commodores „Englisch-Lehrgang" ist auf Cassette erhältlich und basiert auf Lehrbüchern, die in internationalen Sprachschulen Anwendung finden. Dieser Kurs konzentriert sich auf Grammatik und die Anwendung von Wörtern. Das Hauptmenü bietet Wahlmöglichkeiten zwischen Aussprache, Grammatik, Wortsinn und Satzbau (im Grunde aber eine Mischung aus Übungen von Wortdefinitionen und Verwendung der Wörter nach dem Prinzip der Mehrfachwahl).

Englische Literatur

Zum Leidwesen der Software-Hersteller besteht die Pflichtlektüre in der englischen Literatur – für Grundkurse wie für Fortgeschrittenen-Kurse – aus bestimmten Werken. Und diese Literaturliste ist lang.



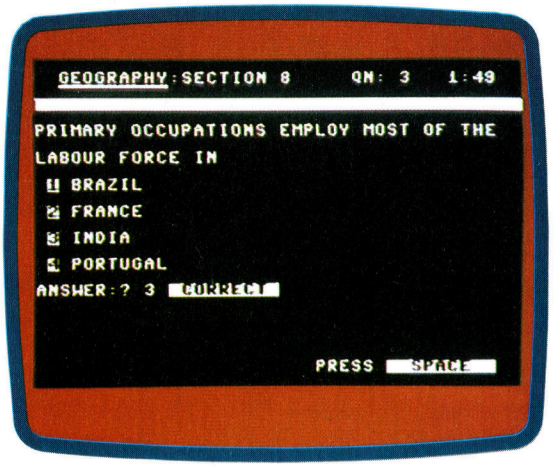
Sussex Softwares „Pride and Prejudice" ist ein exzellentes Beispiel dafür, wieviel Arbeit für die Erstellung eines solchen Programmpakets erforderlich ist. Geschrieben für den 380Z von Research Machines (und künftig mit gewissen Abwandlungen auch für andere CP/M-fähige Computer lieferbar), enthält das Programmpaket drei Disketten und konzentriert sich auf Charaktere, Themen und Inhalt sowie das soziale Umfeld der Personen.

Dieses Programmpaket soll angeblich die von der englischen Prüfungskommission gestellten Mindestanforderungen bei weitem überschreiten. In Wirklichkeit aber ist es allenfalls für Schüler der Mittelstufe geeignet.

Geschichte und Erdkunde

In Geschichte, Erdkunde und Wirtschaftslehre müssen Schüler überwiegend bestehende Fakten auswendiglernen. Der Computer ist deshalb als Hilfe bestens geeignet.

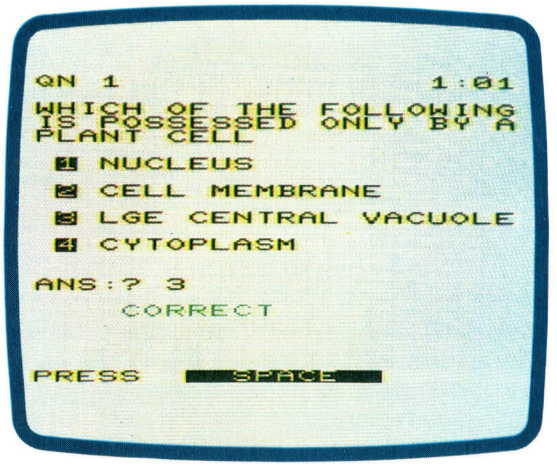
Sussex Software liefert über ein Dutzend solcher Programmpakete. Auch hier ist das Gros der Programme für 380Z- und Commodore-4000-Benutzer gedacht. Eine parallele Serie wird für den Sinclair Spectrum geschrieben, vertrieben unter dem Label „Akadimias".



Auch von Commodore gibt es Geschichts-Software, so „Die Geschichte des 20. Jahrhunderts" für den C 64 und den VC 20. Diese mit Ivan-Berg-Software produzierte Programmsammlung enthält eine Zusammenfassung der politisch, soziologisch und wirtschaftlich wichtigen Daten dieses Zeitraums, gegliedert in acht Rubriken. Das Programm ist auf Cassette erhältlich.

Mathematik u. Naturwissenschaft

Nach englischen Prüfungsreglements gibt es im Fach Mathematik und den Naturwissenschaften keine festgelegten Schwierigkeitsgrade. Diese Stoffe sind auch am leichtesten im Programm darstellbar. Es gibt eine Fülle unterschiedlicher Produkte, in denen die Naturwissenschaften und Mathematik allgemein oder speziell (beispielsweise Geometrie oder Trigonometrie) dargestellt werden.



Commodores Mathematik-Programme sind gute Beispiele dafür, wie man Schulstoff per Computer aufbereiten kann. Vergleichbare Software gibt es für fast alle handelsüblichen Heimcomputer. Da mathematisches wie naturwissenschaftliches Material leicht durch einfache Diagramme illustriert werden kann, tauchen bei diesen Computerprogrammen häufig ansprechende Grafiken auf.

Fachwörter von A bis Z

Index = Index
Das Wort Index wird in der Datenverarbeitung in mehrfachem Sinn gebraucht. Häufig ist darunter die Positionsnummer einer Informationseinheit innerhalb einer längeren Liste zu verstehen. Die Indizierung ermöglicht den Einzelzugriff, indem lediglich nach dieser Nummer gesucht wird. Die Feldelemente eines BASIC-Arrays werden zum Beispiel über ihren individuellen Feldindex angesprochen. Bei der Maschinencode-Programmierung spielt das Indexregister eine wichtige Rolle. Vielfach wird auch die Schleifenzähler-Variable bei zyklischen Programmabschnitten als Index bezeichnet.

Indexed File = Index-organisierte Datei
Wenn eine Datei mit einem Index organisiert wird, so wird dieser oft wie der separate Index eines Buches völlig von der Datei getrennt. Ein vertrautes Beispiel ist die Verwaltung von Disketten, bei der der Index als Directory auf einer Extraspur

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.

gespeichert wird. Vor jedem Zugriff wird zunächst die Directory befragt, wo die gesuchte Datei auf der Diskette abgelegt ist.

Ein besonders wichtiger Fall ist die indizierte sequentielle Datei. Hier wird zunächst eine sequentielle Datei angelegt. Die zugehörige Indexdatei enthält nur die Schlüsseldaten der Datensätze, und zwar in derselben Reihenfolge. Zum Aufsuchen eines Datensatzes wird erst die Indexdatei nach den gewünschten Schlüsseldaten durchsucht, und anhand der zugehörigen Positionsziffer kann dann in der sequentiellen Datei direkt auf den richtigen Datensatz zugegriffen werden. Dieses Verfahren wurde ursprünglich für die Bandspeicher von Großrechenanlagen entwikkelt, da die sequentielle Datei selbst wegen ihres Umfangs oft nicht mehr in den Arbeitsspeicher paßt. Die Indexdatei läßt sich jedoch auf kleinstem Raum unterbringen.

Zugriff per Index

Finde „David"

Schlüssel	Nr.
Andreas	1
Bach	−1
Braun	5
Christian	−1
David	7
Daxel	23
Fisch	15
Georg	28
Harms	37
Johnes	25
Klaus	11
Marks	10

Indexdatei

Hauptdatei

	Name	Tel. Büro	Tel. priv.	Beruf
1	Andreas	242 0791	727 0942	Designer
2	Philips	636 2418	221 3940	Buchhalter
3	Schmidt	631 0836	286 8170	Redakteur
4		Datensatz löschen		
5	Braun	729 8213	236 2190	Zahnarzt
6	Peter	836 6622	298 4310	Dekorateur
7	David	743 7216	450 6926	Gärtnerin
8		Datensatz löschen		
9		Datensatz löschen		
10	Marks	730 6321	429 7592	Mechaniker
11	Klaus	493 9899	455 8431	Rechtsanwalt
12	Walter	736 7700	693 0452	Friseurin

Die Indexdatei enthält die Schlüsselfelder der Datensätze in der gleichen Reihenfolge, wie sie in der Hauptdatei stehen. Zum Aufsuchen eines bestimmten Datensatzes wird in der Indexdatei die Position des Schlüssels festgestellt. Zum kurzfristigen Löschen einzelner Sätze genügt ein Vermerk in der Indexdatei.

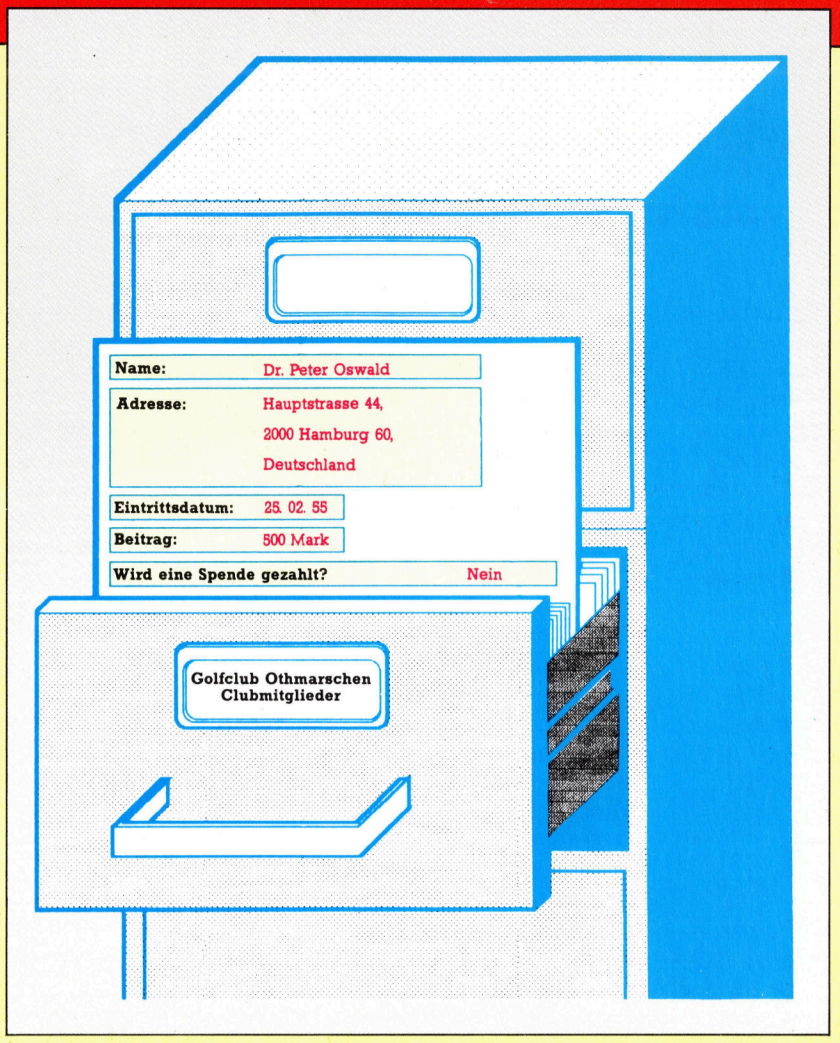
Index Register = Indexregister
In der Zentraleinheit eines Microprozessors sind ein oder mehrere Ein-Byte-Register für das Speichern der vom Programm benötigten Indizes vorgesehen, insbesondere zur Unterstützung der indirekten Adressierung. Dabei wird der Inhalt eines Indexregisters zur Operandenadresse eines Befehls addiert, und der Zugriff erfolgt dann auf die berechnete neue Adresse. Zum Beispiel bewirkt LDA BASE,X das Laden des Akkumulators mit dem Byte, dessen Adresse sich als Summe der Basisadresse BASE und des Index im Register X ergibt.

Die indirekte Adressierung erleichtert vor allem den Umgang mit Arrays – unter Verwendung spezieller Indexbefehle für die Adreßberechnung. Ein Indexregister läßt sich aber auch zum Ablegen kurzfristig benötigter Variablen benutzen. Damit sparen Sie Speicherplatz und Rechenzeit. Schreiben und Lesen dauert länger als ein Rasterzugriff.

Information Management System = Informationssystem
Ein Informationssystem hat die Aufgabe, Informationen verschiedenster Art anzubieten und die Hilfsmittel zu deren Verknüpfung und Auswertung bereitzuhalten. Sehr häufig handelt es sich dabei um Datenbanksysteme. Es genügt nicht, die Information nur vorrätig zu haben, sondern der Benutzer muß darauf auch leicht und schnell zugreifen können, zum Beispiel über einen Suchindex. Ein vollwertiges Informationssystem ist in Verbindung mit der dezentralen Datentechnik (Distributed Processing) ein außerordentlich wirkungsvolles Werkzeug.

Der Begriff Information Management System wird oft auch allgemein für die Organisationsprogramme eines Systems verwendet, zum Beispiel für das Betriebssystem oder die Speicherverwaltung und alle anderen Routinen, die sich mit der Veränderung und Aktualisierung von Datenbeständen befassen.

Bildnachweise
1457: Robyn Beeche
1458, 1459, 1464, 1480, 1481, U3: Kevin Jones
1460: Mike Clowes
1465: Liz Heaney
1470, 1471: Chris Stevens
1472: Geoff Rowley
1476: Liz Dixon
1483, 1484: Pilot Software, Liz Heaney



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

computer kurs

Datenbanken

Datenbanken ermöglichen einen schnellen Zugriff auf Einzeldaten und Datenkombinationen. Wir sehen uns an, wie sie aufgebaut sind.

Der große Bruder

Der mit Interesse erwartete IBM PC wird unter die Lupe genommen. Das Gerät arbeitet mit dem Intel 80286.

Lückenschließer

Das Interface wird nun zusammengebaut. Unser Roboter läßt sich hierüber mit dem Spectrum ganz einfach steuern.

Unter Tatverdacht

Sie stehen unter Mordverdacht, beweisen aber Ihre Unschuld und überführen den Mörder.