Einsteigen - Verstehen - Beherrschen

DM 3,80 öS 30 sfr 3,80

computer kurs

Ein wöchentliches Sammelwerk

Heft 59

BASIC: Leuchtspuren

Montage des Robot-Arms

Rechner Tandy Color

CAD-Anwendung

computer kurs

Heft 59

Inhalt

WIE SIE JEDE WOCHE IHR HEFT BEKOMMEN

Computer Kurs ist ein wöchentlich erscheinendes Sammelwerk. Die Gesamtzahl der Hefte ergibt ein vollständiges Computer-Nachschlagewerk. Damit Sie jede Woche Ihr Heft erhalten, bitten Sie Ihren Zeitschriftenhändler, Computer Kurs für Sie zu reservieren.

Zurückliegende Hefte
Ihr Zeitschriftenhändler besorgt Ihnen gerne zurückliegende Hefte. Sie können sie aber auch direkt beim Verlag bestellen.

Deutschland: Das einzelne Heft kostet DM 3,80. Bitte füllen Sie eine Postzahlkarte aus an: Marshall Cavendish Int. Ltd. (MCI), Sammelwerk-Service, Postgiroamt Hamburg 48064-202, Postfach 105703, 2000 Hamburg 1, Kennwort: Computer Kurs

Österreich: Das einzelne Heft kostet öS 30. Bitte füllen Sie eine Zahlkarte aus an: Computer Kurs, Wollzeile 11, 1011 Wien, Postscheckkonto Wien 7857201 oder legen Sie Ihrer Bestellung einen Verrechnungsscheck bei. Kennwort: Computer Kurs.

Schweiz: Das einzelne Heft kostet sfr 3,80. Bitte wenden Sie sich an Ihren Kiosk; dort werden Sie jederzeit die gewünschten Exemplare erhalten.

Abonnement

Sie können Computer Kurs auch alle 2 Wochen (je 2 Ausgaben) per Post zum gleichen Preis im Abonnement beziehen. Der Abopreis für 12 Ausgaben beträgt DM 45,60 inkl. Mwst., den wir Ihnen nach Eingang der Bestellung berechnen. Bitte senden Sie Ihre Bestellung an: Marshall Cavendish Int. Ltd. (MCI), Sammelwerk Service, Postgiroamt Hamburg 86853-201, Postfach 105703, 2000 Hamburg 1, Kennwort: Abo Computer Kurs. Bitte geben Sie an, ab welcher Nummer das Abo beginnen soll und ob Sie regelmäßig für jeweils 12 Folgen einen Sammelordner wünschen.

WICHTIG: Bei Ihren Bestellungen muß der linke Abschnitt der Zahlkarte Ihre vollständige Adresse enthalten, damit Sie die Hefte schnell und sicher erhalten. Überweisen Sie durch Ihre Bank, so muß die Überweisungskopie Ihre vollständige Anschrift gut leserlich enthalten.

SAMMELORDNER
Sie können die Sammelordner entweder direkt bei Ihrem Zeitschriftenhändler kaufen (falls nicht vorrätig, bestellt er sie gerne für Sie) oder aber Sie bestellen die Sammelordner für den gleichen Preis beim Verlag wie folgt:

Deutschland: Der Sammelordner kostet DM 12. Bitte füllen Sie eine Zahlkarte aus an: Marshall Cavendish International Ltd. (MCI), Sammelwerk-Service, Postgiroamt Hamburg 48064-202, Postfach 105703, 2000 Hamburg 1, Kennwort: Sammelordner Computer Kurs.

Österreich: Der Sammelordner kostet öS 98. Bitte füllen Sie eine Zahlkarte aus an: Computer Kurs Wollzeile 11, 1011 Wien, Postscheckkonto Wien 7857201 oder legen Sie Ihrer Bestellung einen Verrechnungsscheck bei. Kennwort: Sammelordner Computer Kurs.

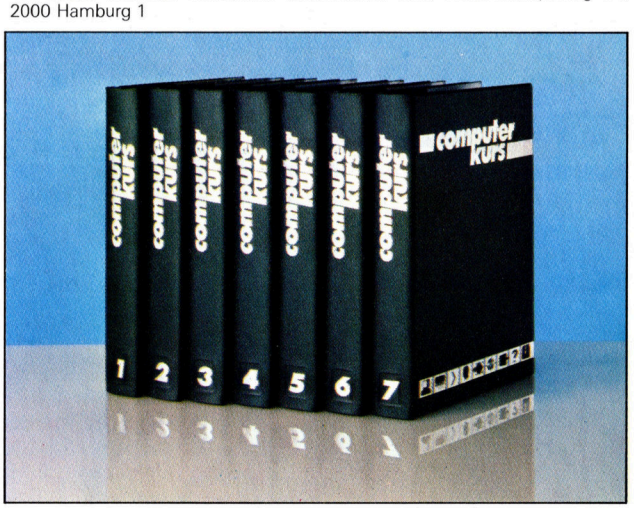
Schweiz: Der Sammelordner kostet sfr 15. Bitte wenden Sie sich an Ihren Kiosk; dort werden Sie jederzeit die gewünschten Exemplare erhalten.

INHALTSVERZEICHNIS

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

Redaktion: Winfried Schmidt (verantw. f. d. Inhalt), Peter Aldick, Holger Neuhaus, Uta Brandl (Layout), Sammelwerk Redaktions-Service GmbH, Paulstraße 3, 2000 Hamburg 1

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



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

Produktionsmittel

Bei modernen Fertigungsverfahren werden Zeichnungen umgesetzt, die mit Hilfe von CAD erstellt wurden. In diesem Artikel stellen wir verschiedene Techniken vor.

Früher lasen Ingenieure Bauzeichnungen und ließen die Werkzeugmaschinen mit der Hand fertigen. Numerisch gesteuerte Maschinen (NC) leiteten einen wesentlich schnelleren und genaueren Herstellungsprozeß ein: Die Designer erstellten technische Zeichnungen, die von den Produktionsingenieuren in NC-Programme übersetzt wurden, und die Werkzeugmaschinen führten die eingegebenen Kommandos aus.

NC-Programmierer haben einen Industrie-Befehls-Standard gesetzt, mit denen die Bewegungen, die eine Maschine ausführen kann, definiert sind. Umfangreiche Aufgaben werden häufig in verschiedenen einzelnen Bauabschnitten gelöst, die von der Zeichnung zum Fertigprodukt führen. Diese Methode wird allmählich durch computergesteuerte, numerisch programmierte Maschinen (CNC) abgelöst. In den fortschrittlichsten System dieser Art findet bereits die Möglichkeit Anwendung, über CAD automatisch generierte NC-Programme einzufügen.

Nicht alle CAD-Systeme sind dazu imstande, doch kann in jeder Entwicklungs- und Produktionsphase die zentrale CAD-Datenbank genutzt werden. Diese enthält genaue geometrische Beschreibungen der Einzelteile. Der Produktionsingenieur hat Zugriff auf eine „Bibliothek" von Werkzeugmaschinen. Er wählt beispielsweise die für das Drehen erforderliche Prozedur und sieht, während er das Programm schreibt, wie sich das Werkzeug längs der Kontur der Zeichnung bewegt. Auf diese Weise lassen sich kleine, aber letztlich äußerst kostspielige Fehler bereits vor der Produktion vermeiden.

Beschleunigte Fertigung

Die Verwendung derselben Datenbank beschleunigt auch den Fertigungsprozeß. Jede Änderung, die der Zeichner vornimmt, wird sofort an den Ingenieur weitergeleitet. Zeichnungsfehler können somit direkt und ohne Verzögerung korrigiert werden.

Moderne Produktions-Steuerungs-Systeme sind Grundlagen für einen reibungslosen Ablauf in fast allen Industriezweigen. Dazu bedient man sich natürlich der Microprozessoren. Ein Prozessoren-Steuerungssystem besteht aus folgenden fünf Komponenten:

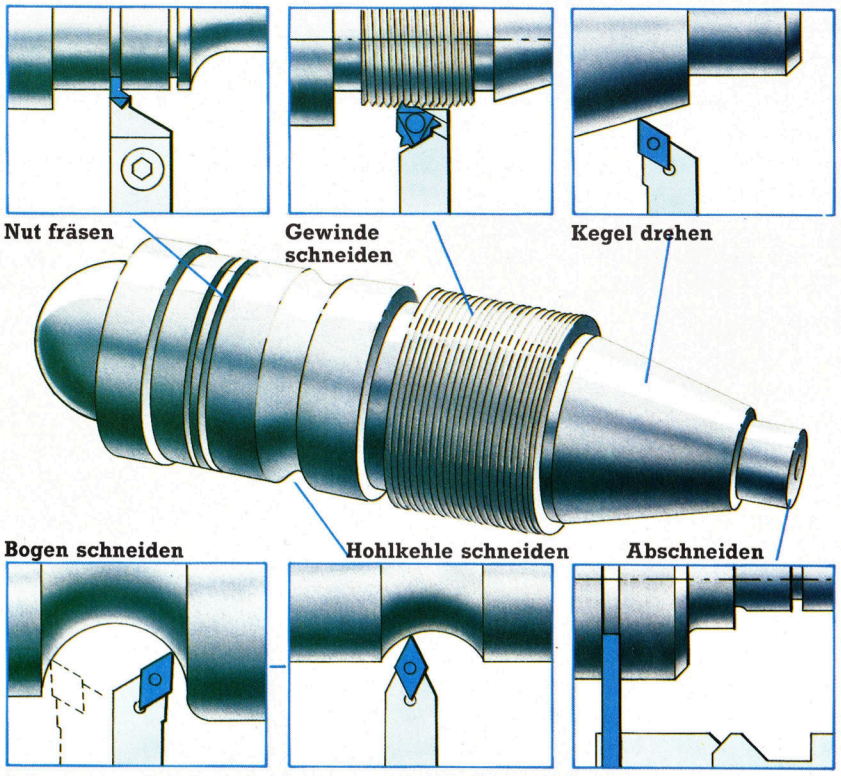
- Einem Sensor mit dem bestimmte Charakteristiken wie Temperatur, Flüssigkeitsmenge oder Druck gemessen werden.
- Einem Aktuator, mit dem eine Tätigkeit als Reaktion auf ein Signal eingeleitet wird.
- Einem Kontrollventil oder vergleichbarem. Dies wird durch den Aktuator gesteuert.
- Einer Steuereinheit, die bestimmt, welcher Vorgang auszuführen ist – basierend auf der Information, die der Sensor ihm zuführt.

Diese Information leitet der Transmitter weiter,

Fortschritte in der Verarbeitung natürlicher Sprachen erlauben den Programmierern Schnittstellen zu konstruieren, die Beschreibungen in deutscher Sprache akzeptieren. Künftig könnte eine Mischung aus CNC und Roboter-Technologie „intelligente" Maschinen erzeugen.

Präzisionswerkzeuge

CNC-Maschinen können viele unterschiedliche Aufgaben mit ein und derselben Komponente durchführen, indem unterschiedliche Werkzeuge entsprechend den Vorgaben des Hauptprogramms eingesetzt werden. Die Eingabe erfolgt entweder über den Ingenieur oder direkt über das zentrale Computersystem. Durch Mikrocomputer-Steuerung beherrscht eine einzige Maschine alle Techniken einer herkömmlichen Drehbank.



Steuerungskontrolle

Die eigentliche Arbeit eines Steuerungs-Systems erledigen die Steuerungs-Sensoren. Dabei handelt es sich um viele verschiedene Geräte, die Daten lesen, messen und auswerten.

Temperaturmessung

- Thermoelement: Das preiswerteste und verbreitetste Temperatur-Meßinstrument. Dabei werden zwei unterschiedliche Metalle verwendet, die sich unter Hitzeeinwirkung unterschiedlich ausdehnen. Dadurch wird eine Spannung erzeugt, die als Basis für ein Analogsignal dient.
- Widerstandsthermometer: –Dieses basiert auf dem Prinzip, daß Hitze den Widerstand leitender Drähte vergrößert. Der Strom durchläuft einen erhitzten und einen unerhitzten Draht. Der Spannungsunterschied wird gemessen und die Widerstandsveränderung berechnet.
- Strahlungspyrometer: Es mißt Strahlungshitze und wird für höhere Temperaturen verwendet. Das Pyrometer fokussiert abgestrahlte Hitze auf Thermoelemente und mißt die dort vorhandene Temperatur.

Flußmessung

- Differentiale Druckmesser: Sie berechnen die Flußrate durch Messen des Drucks an unterschiedlichen Punkten an beiden Seiten eines Hindernisses.

Druckmessung

- Bourdon-Röhre: Hierbei werden Metallrohre verwendet, die C-, Spiral- oder helixförmig gebogen sind. Wird Druck auf ein Ende ausgeübt, versucht das Rohr, sich zu „strecken". Die Druckmessung erfolgt dadurch, daß der Bewegungsgrad des freien Endes gemessen wird.
- Manometer: Ein U-förmiges Rohr mit einem längeren Arm, an dem sich ein Schwimmer befindet. Druck im längeren, dünneren Arm verändert den Oberflächenspiegel im anderen.
- Membran: Ein mechanisches Element mißt Bewegungen, die auf eine Gummi-Membran ausgeübt werden. Im Balgen befindet sich ein empfindlicherer Sensor, zu dem die Daten geleitet und gemessen werden.

Stufenmessung

- Schwimmer: Arbeitet nach dem Prinzip eines Schwimmers wie beim Angeln. Verände-

der das Sensor-Signal empfängt und zuvor in eine Form umwandelt, die die Steuereinheit „verstehen" kann.

Analog-Geräte geben Informationen als Spannung weiter. Aus diesem Grunde ist in das System ein Analog/Digital-Konverter eingebaut, der das Signal in eine für den Computer verständliche Form umwandelt. Produktions-Steuerungs-Systeme sind an zentraler Stelle in den Produktionsprozeß integriert, wobei sie nur eine Aufgabe haben: Dasselbe Programm immer wieder ablaufen zu lassen.

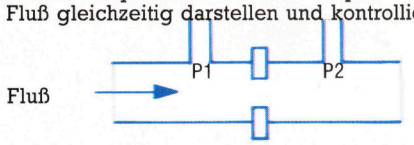
Steuerungs-Systeme lassen sich in zwei Arten klassifizieren: Ständige Kontrolle und sequentielle Kontrolle.

Optimale Reaktionen

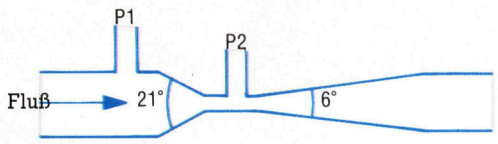
Beide Systeme schließen einander nicht aus und können sogar in dasselbe System integriert sein. Bei der ständigen Kontrolle werden Rohmaterialien in einen Fertigungsprozeß eingebracht. Hierbei ist lediglich eine Steuerung der Flußrate erforderlich. Um optimale Kondi-

Schneller Durchfluß

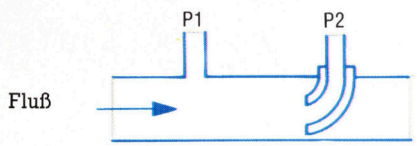
Der Fluß von Flüssigkeiten und Gasen kann gemessen und an einen Computer weitergeleitet werden. In jedem Fall wird der Druck an zwei Punkten gemessen, P1 und P2. Der Druckunterschied entspricht P1–P2. Ein Computer kann den Fluß gleichzeitig darstellen und kontrollieren.



Mündungsblatt



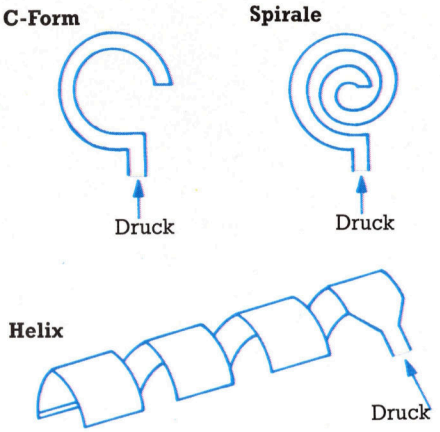
Düse



Pitot-Rohr

Unter Druck

Das Bourdon-Rohr besteht aus einem Rohr, das sich ausstreckt, sobald es einem Druck ausgesetzt wird. Die Bewegung des Rohres wird dann gemessen, und das Ergebnis gibt die Druckstärke an. Diese kann zwecks Analyse an den Computer weitergeleitet werden. Es gibt drei Arten der Bourdon-Röhre, C-Form, Spirale und Helix.



rungen werden gemessen, wenn der Schwimmer sich auf- oder abwärts bewegt.

- Druckmesser: Der Druck am Boden eines Tanks ist Maßeinheit für den Inhalt der Flüssigkeit eines Tanks, somit auch des Flüssigkeitsstandes.
- Strahlung: Eine Zelle auf einer Seite des Gefäßes berechnet die Stärke der Gammastrahlung von der anderen Seite.
- Ultraschallsensor: Hierbei wird die Tiefe unter Verwendung eines Ultraschallsignals gemessen.

Gewichtsmessung

- Elektrischer Widerstand: Hier findet das Prinzip Anwendung, daß elektrische Konduktoren ihren Widerstand unter Spannung erhöhen.
- Halbleiter-Anzeige: Halbleiter erzeugen unter Druck eine Spannung und können deshalb als Maßeinheit verwendet werden.

Chemische Analyse

- Infrarot. Diese Analyseart wird bei der Gasmessung verwendet. Gasproben werden an einer Infrarot-Quelle vorbeigeleitet und zu einem Infrarot-Detektor geführt. Die Menge der blockierten Infrarot-Strahlen ist Maßeinheit für die Gasmenge.
- Thermische Leitfähigkeit: Veränderungen eines Gases beeinflussen seine Wärmeleitfähigkeit. Das Gas wird durch ein Heizelement geleitet. Ändert sich die Leitfähigkeit, kann festgestellt werden, was mit dem Gas geschehen ist.

Geschwindigkeit

- Tachogenerator: Hiermit wird die Drehgeschwindigkeit gemessen. Ein Tachometer wird an die rotierende Achse angeschlossen, die beim Drehen Strom erzeugt. Da der Strom proportional der Geschwindigkeit ist, kann letztere gemessen werden.

Die Werte dieser Sensoren werden in einen Dreiwert-Kontroller eingespeist. Dort wird der gemessene Wert der Sensoren abgenommen und mit einem vom Operator gesetzten Wert verglichen. Das ist die Ebene, durch die eine Reaktion erzeugt wird. Liegt der Wert über oder unter dem Richtwert, wird die Ventil-Funktion aktiviert und der Prozeß in erforderlicher Form korrigiert.

tionen zu erreichen, muß das System in der Lage sein, auf kleine Veränderungen zu reagieren. Bei der sequentiellen Kontrolle hingegen erfolgt dieser Vorgang in Abschnitten. Das Rohmaterial durchläuft eine Reihe physikalischer oder chemischer Bearbeitungsprozesse. Die Kontrollfunktion besteht also darin, eine Reihe von Aktionen zur rechten Zeit und unter den richtigen Bedingungen auszulösen. Zugleich müssen die verschiedenen Sequenzen kontrolliert werden, um sicherzustellen, daß die gleiche Bearbeitung erfolgt.

Der große Vorteil des Computereinsatzes bei der Fertigungskontrolle besteht darin, daß viele Dinge gleichzeitig getan werden können. Die einzelnen Vorgänge können durch Standard-Komponenten individuell gesteuert werden. Schließlich kann ein einziger Mitarbeiter viele verschiedene Produktionsvorgänge von einem einzigen Terminal aus überwachen. Statt Hunderte von Anzeigen und Instrumenten zu lesen, schaut der Mitarbeiter lediglich auf einen Monitor, der die Fabrik mit ihren Fertigungsstraßen in Diagrammform zeigt.

Entscheidungshilfe

? Welche Datenbankverwaltung (DBV) soll ich kaufen, wenn ich die benötigte Anzahl der Felder noch nicht kenne?

In den meisten DBVs kann ein Datensatz nur eine bestimmte Anzahl Felder enthalten. Es ist oft schwierig, die Felderzahl im voraus genau zu bestimmen. Die meisten DBVs sind für eine große Anzahl von Feldern ausgelegt. Die Länge der einzelnen Felder ist jedoch oftmals auf eine Bildschirmzeile begrenzt. (Damit wird der Aufbau von Texten, die mehrere Zeilen umfassen, recht umständlich.) Weitaus häufiger ist nicht die Feldzahl beschränkt, sondern die Anzahl der Zeichen pro Datensatz. Bei einer typischen Grenze von 1020 Zeichen pro Satz kann der Anwender selbst bestimmen, wie diese Kapazität zwischen Anzahl der Felder und Zeichen pro Feld aufgeteilt werden soll. Stellen Sie aber in jedem Fall sicher, daß die maximal mögliche Felderzahl pro Datensatz und die Zeichen pro Feld für Ihre Anwendung ausreichen.

? Welche Faktoren muß ich berücksichtigen, wenn die Daten eines Datensatzes sich auch auf den Inhalt anderer Dateien beziehen sollen?

Die einfacheren DBVs können immer nur eine Datei zur Zeit bearbeiten. Für viele Anwendungen reicht das völlig aus. Praktischer sind jedoch Datenbanken, mit denen sich zwei oder mehr Dateien gleichzeitig ansprechen lassen. Ein klassisches Beispiel dafür ist eine Lagerhaltung, bei der man pro Artikel mehrere Lieferanten definieren kann und deren Dateien „ARTIKEL" und „LIEFERANTEN" sich für Suchaufgaben miteinander kombinieren lassen. Es erleichtert die Arbeit sehr, wenn über eine Datei auf eine zweite zugegriffen werden kann.

Nehmen wir als Beispiel einen Antiquitätenladen, der seine Artikel in der Datei LAGER gespeichert hat. Weiterhin existiert eine Lieferantendatei, deren Artikel abrufbar sind, aber nicht im Lager stehen. So kann es vorkommen, daß wir unter BESCHREIBUNG „MÖBEL SCHWEDISCH" keinen Eintrag finden, aber beim Durchsuchen der Datei „LIEFERANTEN" auf die Information KURT JAKOBSEN ANTIK, GAMLA STAN 56, STOCKHOLM stoßen, die in der Datei LAGER folgende Einträge enthält:

```
BESCHREIBUNG: SOFA
FABRIKANT:    ANDERSEN
DATUM:        1824
PREIS:        22 000
BESCHREIBUNG: SOFA
FABRIKANT:    GRIMM
DATUM:        1874
PREIS:        18 000
```

Hier liefert eine zweite Datei Hinweise, die beim Arbeiten mit nur einer Datei nicht so einfach zu erhalten gewesen wären.

Obwohl sich in einen Datensatz theoretisch immer neue Felder für zusätzliche Informationen einfügen lassen, ist dieser Weg in der Praxis kaum gangbar. Nehmen Sie an, Monsieur Dupont in Paris teilt mit, er hätte zwei Tische aus dem 18. Jahrhundert erhalten, an denen wir interessiert sein könnten. Hier ist es viel einfacher, diese Informationen über die Lieferantendatei unter dem Datensatz DUPONT einzutragen, als in der Datei mit der Bezeichnung LAGER die neuen Angebote einzufügen.

Kurz gesagt, in jeder Anwendung, in der Querverweise auf Dateien mit anderem Inhalt notwendig sind, lohnt es sich, eine DBV anzuschaffen, die mit mehreren Dateien gleichzeitig arbeiten kann.

? Wie viele Datensätze sollte eine DBV verwalten können?

Unabhängig davon, ob Sie wenig oder viele Datensätze anlegen werden, gibt es eine Faustregel zur Feststellung der benötigten Kapazität: Berechnen Sie die größtmögliche Anzahl von Datensätzen, die Sie je benötigen werden, verdoppeln Sie diese Zahl und suchen Sie dann nach einer geeigneten DBV. Wenn Sie damit Ihr Warenlager verwalten wollen, aber nicht viele Artikel führen, brauchen Sie natürlich keine DBV für 32 000 Datensätze. Wenn sie jedoch den Katalog einer Bücherei führen wollen, könnten auch 32 000 Sätze nicht ausreichen. DBVs für mehr als 64 000 Datensätze laufen normalerweise nur auf größeren Computern mit Festplattenspeichern.

? Kann ich eine DBV selbst modifizieren, oder bin ich an die vom Computer vorgegebene Sprache gebunden?

Einige DBVs verfügen über eine eingebaute Programmiersprache, mit der sich umfangreiche Abläufe automatisch ausführen lassen. Im kommerziellen Bereich lassen sich mit diesen Sprachen Befehlsabfolgen programmieren.

Zwei Beispiele für DBVs dieser Art sind dBASE II und Archive. Wenn Sie mit den Daten Ihrer Datenbank ständig wiederkehrende Abläufe ausführen (etwa: „eine Liste aller während des Tages verkaufter Artikel drukken, dann eine Liste aller Artikel, deren Bestand unter dem festgesetzten Minimum liegt"), benötigen Sie eine DBV mit eingebauter Programmiersprache. Wenn Sie dagegen Ihre Datenbank jedesmal auf andere Weise abfragen, genügt eine DBV mit standardmäßigen Suchbefehlen.

? Was sollte außerdem beim Kauf einer Datenbankverwaltung beachtet werden?

Textsysteme benötigen oft sehr viel Zeit, um Dateien zu finden oder Textblöcke zu verschieben. Im Vergleich mit den Abläufen des gesamten Textsystems fällt die Wartezeit für das „Suchen und Sortieren" nicht ins Gewicht, da die Dateneingabe (Eintippen der Texte) und der Druck die meiste Zeit benötigen.

Bei DBVs besteht die umgekehrte Situation: Hier wirken Datensätze wie kurze Texte, die zwar schnell eingegeben werden, für deren Sortierung und Suche eine DBV jedoch die meiste Zeit aufwendet. Bei größeren Dateien, die nicht im Ganzen in den Arbeitsspeicher geladen werden kön-

nen (eine Situation, die oft eintritt), muß das System zum Lesen oder Schreiben auf das externe Speichermedium zugreifen. An dieser Stelle treten die Nachteile der Cassettenspeicherung deutlich zutage: Nur bei sehr kleinen Dateien lohnt es sich nicht, großartig in ein Diskettensystem zu investieren.

Bei kommerziellen Datenbanken gilt die Devise: „Zeit ist Geld". Wenn Sie eine DBV kommerziell einsetzen wollen, sollten Sie sich in jedem Fall ein Diskettensystem zulegen.

? Welcher Unterschied besteht zwischen menügesteuerten und befehlsgesteuerten DBVs, und für welche Anwendungsbereiche eignen sich diese beiden Typen?

Menügesteuerte Datenbanken zeigen in jedem Arbeitsstadium die zur Verfügung stehenden Bearbeitungsmöglichkeiten. Bei befehlsgesteuerten Programmen müssen Sie die einzelnen Befehle (oft eine Kombination von CTRL-Taste und Buchstabenfolgen) erst lernen. Obwohl menügesteuerte Systeme für den Anfänger leicht zu bedienen sind, erweisen sich befehlsgesteuerte DBVs bei der Verarbeitung der Daten als schneller und flexibler.

? Wie wichtig sind Schlüsselfelder, und wie viele werde ich brauchen?

Schlüsselfelder können von der DBV durchsucht oder als Sortierkriterium eingesetzt werden. Ein Feld, das nicht als Schlüsselfeld angelegt wurde, kann nicht für die Identifizierung eines Datensatzes herangezogen werden. Hier einige Beispiele für den Einsatz von Schlüsselfeldern:

```
SEARCH LIEFERANT FOR
"Simons Bau GmbH"
```

Dabei wird das in jedem Datensatz enthaltene Schlüsselfeld LIEFERANT auf eine Übereinstimmung mit dem angegebenen String durchsucht, während

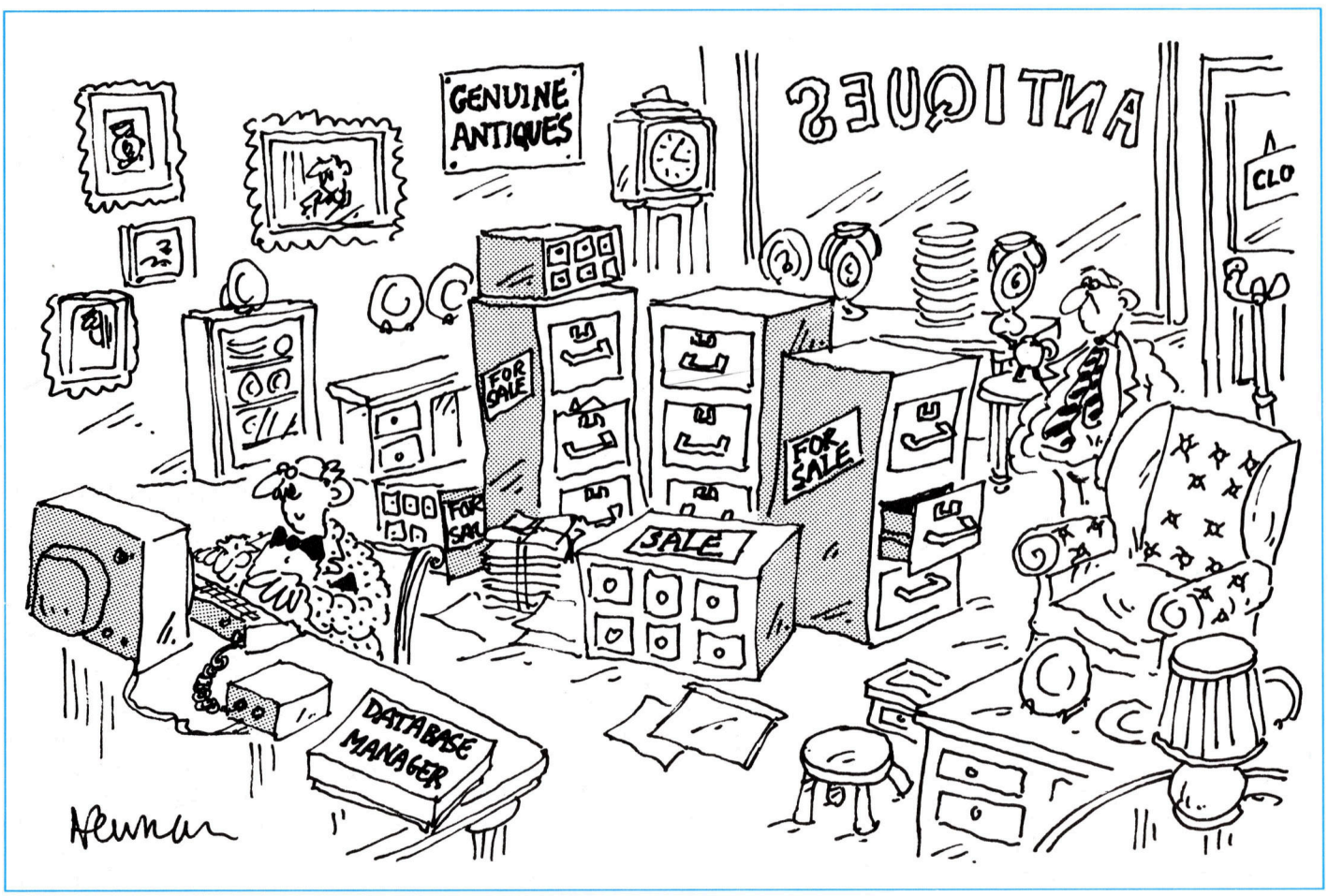
```
SEARCH PREIS <= 45,50
```

überprüft, ob die im Schlüsselfeld gespeicherten Zahlen kleiner oder gleich 45,50 sind. In einigen DBVs lassen sich alle Felder als Schlüsselfelder einsetzen.

? Bedingungsabhängige oder berechnete Felder scheinen sehr praktisch zu sein. Für welche Aufgaben kann ich sie einsetzen?

Viele DBVs bieten sogenannte bedingungsabhängige Felder. Diese Felder erscheinen (bei der Dateneingabe und auch später) nur, wenn sie angesprochen werden. Dabei unterdrückt beispielsweise eine Datenbank, deren Feld „ANZAHL KINDER" den Eintrag 0 erhält, alle Zusatzfelder wie „NAME DES ERSTEN KINDES", „ALTER DES ERSTEN KINDES" etc., die bei „ANZAHL KINDER" = 1 (oder höher) erscheinen.

Bei einigen DBVs lassen sich die Daten bestimmter Felder als Argumente für automatische Berechnungen einsetzen. In derartigen Systemen wird beispielsweise die unter „LAGERMENGE" eingesetzte Zahl automatisch mit der Zahl des Feldes „PREIS" multipliziert und das Ergebnis ebenfalls summiert, so daß der Gesamtwert des Lagers ohne weitere Operationen sofort zu erkennen ist. Diese praktische Funktion gehört jedoch zum Standard jeder DBV.



Leinen los!

Nachdem wir alle Reisevorbereitungen getroffen haben, kann nun der Anker gelichtet werden. Während der Fahrt müssen die Lebensmittelrationen verteilt und die Gesundheit der Mannschaft überprüft werden.

Die wöchentlichen Ereignisse während der Fahrt können mit Hilfe einer FOR. . .NEXT-Schleife, unter Verwendung der Variablen WK zum Zählen der Wochen, kontrolliert werden. Die obere Grenze der Schleife wird durch den Wert der Variablen JL gesetzt. Innerhalb der Schleife können Unterroutinen aufgerufen werden, um die verschiedenen wöchentlichen Aufgaben zu handhaben. Im Logbuch werden die Stärke der Mannschaft, die Vorräte usw. aufgeführt. Diese Schleife liegt zwischen den Zeilen 800 und 889.

Ausreichende Ernährung ist der wichtigste Faktor, um die „Stärke" der Mannschaft zu erhalten. Um vollkommen gesund zu bleiben, muß jedes Besatzungsmitglied den gesamten wöchentlichen Bedarf an Obst, Gemüse, Fleisch und Wasser erhalten. Bei Beginn der Reise ist die Stärke jedes Mitglieds im Array TS(,) auf 100 Einheiten gesetzt. Das Programm führt Berechnungen aus, um zu prüfen, ob die Vorräte bis zum Ende der Reise ausreichen. Wird der Vorrat eines Nahrungsmittels knapp, wird der Spieler gefragt, ob die Mannschaft für diesen Vorrat auf halbe Ration gesetzt werden soll. Wenn ja, so reduziert sich die Stärke der Mannschaft um jeweils fünf Einheiten.

Ist ein Vorratstyp vollständig aufgebraucht, so wird die Stärke wöchentlich um zehn Einheiten reduziert. Erhält die Mannschaft beispielsweise kein Obst mehr und nur noch die halbe Ration Wasser, so reduziert sich ihre Stärke wöchentlich um fünfzehn Einheiten. Ist dann überhaupt kein Wasser mehr vorhanden, so sind es zwanzig Einheiten.

Der Gesundheitszustand der Mannschaft wird am Anfang jeder Woche durch die Unterroutine ab Zeile 4000 ausgegeben. Es gibt vier Stufen. Die höchste ist „sehr gesund" (Stärke 75–100), „gesund" (50–75), „krank" (25–50) und „sehr krank" (1–25). Erreicht die Stärke eines Mannes den Wert 0, so stirbt er und wird im Meer beigesetzt. Sein wöchentlicher Lohn wird jedoch weiterberechnet und nach der Rückkehr an die Angehörigen ausgezahlt. Diese Unterroutine stellt auch die Lohnrechnung für die folgende Woche, die bisherigen Gesamtlöhne und das verbliebene Gold dar. Der errechnete Gesamtbetrag wird am Anfang dieses Programmteils bei Zeile 800 auf 0 gesetzt und durch den Betrag des Wochenlohnes während der Schleifendurchläufe erhöht. Übersteigt der Gesamtbetrag das vorhandene Kapital (Überprüfung in der Unterroutine ab Zeile 5000), wird der Spieler informiert, daß ein Gewinn erzielt werden muß, um die Löhne der Mannschaft bezahlen zu können.

Logbuch führen

Wenn alles gut geht, sollte die Reise acht Wochen dauern, doch unvorhersehbare Umstände können diese Zeit verlängern. Die String-Variable H$ in Zeile 801 zeigt an, wenn die Mannschaft auf halbe Ration gesetzt ist. Die Routine bestimmt die Stärke und den Mannschaftstyp jedes Mannes, indem sie das Array TS(,) überprüft. Der Platz im Array reicht für 16 Männer, so daß der Zähler auf 16 gesetzt wird. Ist an einem Platz für den Mannschaftstyp eine 0 eingetragen, so ist der Abschnitt leer, und Zeile 4070 verzweigt das Programm zu Zeile 4110.

Stirbt ein Mann, muß dies im Logbuch verzeichnet werden. Da jedoch eine 0 (für Stärke) den Tod eines Mannes kennzeichnet, würde jeder leere Abschnitt im Array TS(,) einen „Toten" repräsentieren. Um das Problem zu umgehen, wird der Wert, wenn er 0 erreicht, für eine

Woche auf −999 gesetzt. Erst am Ende der Woche wird er dann auf 0 gesetzt, und das Programm informiert den Spieler am Anfang der nächsten Woche, daß ein Besatzungsmitglied gestorben ist. Die Stärke der übrigen Männer wird mit den Zeilen 4080 bis 4098 überprüft. Der Wert wird mit den vier Gesundheitsstufen verglichen, und in Zeile 4099 wird der Status jedes Mannes ausgegeben. Die Schleife endet in Zeile 4110, und das Programm untersucht den nächsten Abschnitt des Arrays.

Die wöchentliche Lohnabrechnung wird in einer anderen Schleife zwischen den Zeilen 4120 bis 4135 berechnet. Die Formel in Zeile 4130 multipliziert die Anzahl eines Mannschaftstypes mit der entsprechenden Lohnhöhe und addiert den Wert zur Variablen WW. WW wird am Anfang jeder Woche durch Zeile 4119 auf 0 gesetzt und repräsentiert den Gesamtwochenlohn, der in Zeile 4145 ausgegeben wird. Danach wird er zum bisherigen Gesamtlohn (WT) hinzugefügt und in den Zeilen 4160 und 4165 ausgegeben. Der Spieler kann diesen Wert mit dem übrigen Kapital vergleichen, das in Zeile 4175 dargestellt wird.

Die Unterroutine bei Zeile 5100 handhabt die Verteilung der Rationen an die Mannschaft. Unter Verwendung einer FOR. . .NEXT-Schleife für jeden Vorratstyp, werden diverse Vergleiche in Bezug auf die Vorratsmengen ausgeführt. Zuerst wird getestet, ob noch Vorrat vorhanden ist. Dazu wird der Wert des jeweiligen Vorrats, gespeichert in PA(), auf die Bedingung größer 0 getestet. Ist das Ergebnis negativ, wird die Stärke jedes Mannes um zehn reduziert. Dies geschieht in der Unterroutine bei Zeile 9300, die die Stärke um den Wert von WF reduziert. Mit dieser Routine können also unterschiedliche Mengen abgezogen werden, indem WF zuvor ein Wert zugewiesen wird.

Ist die Menge des übrigen Vorrats größer als 0, wird überprüft, ob der Vorrat für den Rest der Reise reicht. Dies wird in Zeile 5180 durch die Anzahl an Männern, CN, die benötigte wöchentliche Ration, PN(), und die Anzahl der noch verbleibenden Wochen berechnet. Ist die Menge unzureichend, so wird der Spieler gefragt, ob er die Mannschaft für diesen Proviantyp auf halbe Ration setzen möchte. Der Status (entweder halbe oder volle Ration) wird durch das in Zeile 802 dimensionierte Array HR() angezeigt. Bei vollen Rationen ist das entsprechende Element auf 1 gesetzt, ansonsten auf 0,5. Ist die Mannschaft während der Reise auf halbe Ration gesetzt, so wird durch die Unterroutine bei Zeile 9300 die Stärke um den Wert 5 reduziert.

Der abschließende Vergleich in dieser Routine kontrolliert, ob die Menge des vorhandenen Vorrats kleiner als die auszugebende Menge ist. Ist dies der Fall, so wird das entsprechende Element des Proviant-Arrays auf −999 gesetzt.

BASIC-Dialekte

Spectrum:
Ändern Sie die folgenden Zeilen:

```
 847 LET I$ = INKEY$:IF I$ = "" THEN GO TO 847
4010 CLS
4190 LET I$ = INKEY$:IF I$ = "" THEN GO TO 4190
4205 CLS
4295 LET I$ = INKEY$:IF I$ = "" THEN GO TO 4295
4305 CLS
4398 LET I$ = INKEY$:IF I$ = "" THEN GO TO 4398
5010 CLS
5080 LET I$ = INKEY$:IF I$ = "" THEN GO TO 5080
5103 CLS
5220 INPUT I$:LET I$ = I$ (1 TO 1)
5298 LET I$ = INKEY$:IF I$ = "" THEN GO TO 5298
```

Acorn B:
Führen Sie bitte folgende Änderungen aus:

```
4010 CLS
4190 I$ =GET$
4205 CLS
4295 I$ = GET$
4305 CLS
4398 I$ = GET$
5010 CLS
5298 I$=GET$
```

Modul vier: Die Fahrt

„Länge der Fahrt"-Variable

```
40 JL=8:REM LENGTH OF VOYAGE
```

Hauptschleife der Fahrt

```
800 WT=0
801 H$="N" :REM HALF RATION INDICATOR
802 DIMHR(4):HR(1)=1:HR(2)=1:HR(3)=1:HR(4)=1
820 FORWK=1TOJL : REM MAIN VOYAGE LOOP
825 GOSUB4000: REM CREW STATUS REPORT
830 GOSUB4200:REM PROVISIONS REPORT
835 GOSUB4300:REM OTHER GOODS REPORT
840 GOSUB9200:PRINTCHR$(147)
842 PRINT:PRINT:PRINT
843 S$="IT IS ESTIMATED THAT THE VOYAGE*":GOSUB910
0
844 PRINT"WILL TAKE A FURTHER";JL-WK+1;"WEEKS"
845 GOSUB9200
846 PRINT:S$=K$:GOSUB9100
847 GET1$:IF1$=""THEN847
850 GOSUB5000:REM CHECK WAGE BILL
855 GOSUB5100: REM ISSUE RATIONS
889 NEXT WK
```

Status-Report der Mannschaft

```
4000 REM CREW STATUS REPORT
4010 PRINTCHR$(147)
4020 S$="          CAPTAINS LOG*":GOSUB9100
4025 S$="          ------------*":GOSUB9100
4030 GOSUB9200
4035 PRINT"AT THE START OF WEEK";WK
4040 S$="THE STATE OF THE CREW IS:*":GOSUB9100
4045 GOSUB9200:PRINT
4055 PRINT
4060 FORT=1TO16
4070 IFTS(T,1)=0THEN4110
4075 PRINTC$(TS(T,1));"    (";
4078 IFTS(T,2)=-999THENS$="DEAD !!!!!!!)*":GOTO409
9
4080 IFTS(T,2)>75THENS$="VERY HEALTHY)*":GOTO4099
4085 IFTS(T,2)>50THENS$="HEALTHY)*":GOTO4099
4095 IFTS(T,2)>25THENS$="SICK ! )*":GOTO4099
4098 S$="VERY SICK !! )*"
4099 GOSUB9100:GOSUB9200
4110 NEXT T
4115 GOSUB9200:PRINT
```

Wöchentliche Lohnabrechnung

```
4119 WW=0
4120 FORT=1TO5
4130 WW=WW+(CC(T)*WG(T))
4135 NEXT
4140 S$="WAGE BILL FOR THE WEEK*":GOSUB9100
4145 PRINTWW;"GOLD PIECES"
4150 GOSUB9200
4155 WT=WT+WW
4160 S$="TOTAL WAGES FOR VOYAGE SO FAR:*":GOSUB910
0
4165 PRINTWT;"GOLD PIECES"
4170 GOSUB9200
4175 PRINT"MONEY LEFT = ";MO;"GOLD PIECES"
4180 PRINT:S$=K$:GOSUB9100
4190 GET1$:IF1$=""THEN4190
4199 RETURN
```

Bericht über Vorräte

```
4200 REM PROVISIONS REPORT
4205 PRINTCHR$(147)
4206 PRINT"AT THE START OF WEEK";WK:GOSUB9200
4210 S$="YOU HAVE THE FOLLOWING*":GOSUB9100
4215 S$="PROVISIONS LEFT:*":GOSUB9100
4220 PRINT:GOSUB9200
4225 FORT=1TO4
4226 IFPA(T)=0ORPA(T)=-999THEN4240
4230 PRINTPA(T);U$(T);"S OF ";P$(T)
4232 X=INT(PA(T)/(CN*PN(T)))
4235 PRINT"(ENOUGH FOR";X;" WEEKS)"
4239 GOSUB9200
4240 NEXT
4290 PRINT:S$=K$:GOSUB9100
4295 GET1$:IF1$=""THEN4295
4299 RETURN
```

Bericht über Handelsgüter

```
4300 REM OTHER GOODS REPORT
4305 PRINTCHR$(147)
4306 PRINT"AT THE START OF WEEK";WK:GOSUB9200
4310 S$="YOU ALSO HAVE:*":GOSUB9100
4320 PRINT:GOSUB9200
4322 IFOA(1)=0THEN4332
4325 PRINTOA(1);:S$="BOTTLES OF MEDICINE*":GOSUB91
00
4330 GOSUB9200
4332 IFOA(2)=0THEN4342
4335 PRINTOA(2);:S$="GUNS*":GOSUB9100
4340 GOSUB9200
4342 IFOA(3)=0THEN4352
4345 PRINTOA(3);:S$="BAGS OF SALT*":GOSUB9100
4350 GOSUB9200
4352 IFOA(4)=0THEN4362
4355 PRINTOA(4);:S$="BALES OF CLOTH*":GOSUB9100
4360 GOSUB9200
4362 IFOA(5)=0THEN4372
4365 PRINTOA(5);:S$="KNIVES*":GOSUB9100
4370 GOSUB9200
4372 IFOA(6)=0THEN4380
4375 PRINTOA(6);:S$="JEWELS*":GOSUB9100
4380 GOSUB9200:PRINT
4382 PRINT"YOU HAVE";MO;:S$=" GOLD PIECES LEFT*":G
OSUB9100
4384 GOSUB9200
4397 PRINT:S$=K$:GOSUB9100
4398 GET1$:IF1$=""THEN4398
4399 RETURN
```

Unterroutine für Gesamtlohn-Abrechnung

```
5000 REM CHECK WAGE BILL
5005 IFWT>MOTHEN5010
5008 GOTO5099
5010 PRINTCHR$(147)
5020 PRINT:PRINT:PRINT
5025 S$="THE CREW HAVE HEARD A RUMOUR*":GOSUB9100
5030 S$="THAT YOU DON'T HAVE ENOUGH*":GOSUB9100
5035 S$="GOLD TO PAY THEM AT THE END*":GOSUB9100
5040 S$="OF THE VOYAGE.*":GOSUB9100
5045 GOSUB9200:PRINT
5050 S$="THEY ARE GETTING ANGRY !!*":GOSUB9100
5055 GOSUB9200:PRINT
5060 S$="LET'S HOPE YOU MANAGE TO MAKE*":GOSUB9100

5065 S$="A TRADING PROFIT!*":GOSUB9100
5066 GOSUB9200
5070 PRINT:S$=K$:GOSUB9100
5080 GET1$:IF1$=""THEN5080
5099 RETURN
```

Unterroutine zur Rationsverteilung

```
5100 REM ISSUE RATIONS
5103 PRINTCHR$(147)
5105 S$="       ISSUING RATIONS*":GOSUB9100
5106 S$="       ---------------*":GOSUB9100
5107 GOSUB9200:PRINT"WEEK";WK:PRINT
5108 H$="N"
5110 FORT=1TO4
5112 HR(T)=1
5115 IFPA(T)>0THEN5180
5120 PRINT"NO ";P$(T);" LEFT!!!":GOSUB9200
5130 S$="THE CREW IS GETTING WEAKER !!*":GOSUB9100

5135 WF=10:GOSUB9300
5139 GOTO5290
5180 X=(PN(T)*CN)*(JL-WK+1)
5185 IFPA(T)<XTHEN5200
5190 GOTO5270
5200 PRINT"RUNNING SHORT OF ";P$(T)
5205 GOSUB9200
5210 S$="DO YOU WANT TO PUT THE CREW ON*":GOSUB910
0
5215 PRINT"HALF RATIONS OF ";P$(T)
5220 INPUTI$:I$=LEFT$(I$,1)
5221 IF I$<>"Y"AND I$<>"N" THEN 5220:REM INPUT ERR
OR
5225 IF I$="N"THEN 5270
5230 HR(T)=.5:H$="Y"
5240 WF=5:GOSUB9300
5250 S$="THE CREW IS GETTING WEAKER!*":GOSUB9100
5270 X=PN(T)*HR(T)*CN
5272 IFX>PA(T)THENX=PA(T)
5275 PA(T)=PA(T)-X
5280 IFPA(T)=0THENPA(T)=-999
5285 PRINTX;U$(T);"S OF ";P$(T);" ISSUED"
5290 PRINT:GOSUB9200:NEXT
5295 PRINT:S$=K$:GOSUB9100
5298 GET1$:IF1$=""THEN5298
5299 RETURN
```

Reduzierung der Mannschaftsstärke

```
9300 REM REDUCE CREW STRENGTHS BY WF
9310 FORS1=1TO16
9320 TS(S1,2)=TS(S1,2)-WF
9330 IFTS(S1,2)<1THENTS(S1,2)=-999
9340 NEXT
9399 RETURN
```

Aktionsbereit

Diese Folge beschließt unsere Einführung in das Betriebssystem des Acorn B. Wir sehen uns den Einsatz der Vektoren genauer an und untersuchen, wie wir über das Betriebssystem mit dem Computer kommunizieren können.

Die meisten OS-Routinen (Betriebssystem-Routinen) des Acorn B arbeiten mit Vektoren. Das bedeutet, daß das OS für den Aufruf der Routine OSCLI zunächst eine Routine bei &FFF7 anspricht, die OSCLI (indirekte Weise) ansteuert. Die Routine findet dabei zunächst heraus, wo OSCLI überhaupt liegt, indem sie den Inhalt zweier Bytes auf Seite 2 des RAM untersucht. Diese zwei Bytes werden Vektor genannt: Das niederwertige Adreßbyte befindet sich im Vektorbyte mit der niedrigeren Bytenummer und das höherwertige Adreßbyte im höheren Vektorbyte. Bei der Routine OSCLI liegt das höherwertige Vektorbyte also in &209 und das niederwertige in &208. Diese Methode wird Hi-Lo-Adressierung genannt und für alle Adreßformate der 6502-Prozessoren eingesetzt. Bei jedem Reset legt das OS die Vektoren neu an. Diese etwas komplizierte Art, Betriebssystemroutinen aufzurufen, bietet große Vorteile.

Sie haben sicher bemerkt, daß alle bisher erwähnten OS-Routinen des Acorn B eine Adresse im Bereich von &FF00 und &FFFF aufrufen. Das hat gute Gründe: Für den Aufruf einer Adresse in diesem Bereich wird zunächst eine Routine angesprochen, die einen Sprung auf den Vektor dieser speziellen OS-Routine veranlaßt (wie beim Aufruf von CLI und OSCLI). Die aufgerufene Adresse zwischen &FF00 und &FFFF ist in allen OS-Versionen des Acorn B gleich. Falls die Routinen des OS-ROMs geändert werden müssen, haben die Konstrukteure der Maschine nur die entsprechenden Vektoradressen umzustellen. Wenn der Anwender die OS-Routinen über die korrekten Vektoren angesprochen hat, muß er bei Änderungen des OS seine Programme nicht verändern. In unterschiedlichen OS-Versionen kann daher der Inhalt der Vektoren durchaus variieren. Solange Sie die Einsprungsadressen zwischen &FF00 und &FFFF verwenden, werden Sie die Veränderungen jedoch nicht bemerken.

Der Einsatz von Vektoren hat noch einen weiteren Vorteil: Die Abläufe der OS-Routine lassen sich damit beeinflussen. Wenn Sie den Inhalt des Vektors verändern, können Sie statt der Maschinencodes des OS eigene Routinen verwenden. In einem späteren Teil des Kurses beschäftigen wir uns genauer mit den Vektoren der wichtigsten OS-Routinen.

Sehen wir uns einmal den Vektor USERV an, der die Speicherstellen &200 und &201 einnimmt. Das besondere an diesem Vektor ist, daß er im allgemeinen keine Funktion hat. Er wird von den *-Befehlen (*CODE und *LINE) aufgerufen. Bei ihrer Verwendung erscheint normalerweise die Fehlermeldung „BAD COMMAND".

Mit USERV können Sie jedoch eine Funktion definieren, die *CODE und *LINE ausführen sollen – sogenannte „anwenderdefinierte Befehle". *CODE eignet sich besonders für die Übergabe von Parametern an Maschinencodeprogramme:

Register	*CODE x,y	*LINE Textstring
A	0	1
X	Enthält den Wert des ersten Parameters nach *CODE. Er muß zwischen 0 und 255 liegen.	Enthält das niederwertige Adreßbyte, in dem sich das erste Stringzeichen nach *LINE befindet.
Y	Enthält den Wert des zweiten Parameters nach *CODE. Auch dieser Parameter muß zwischen 0 und 255 liegen.	Enthält das höherwertige Adreßbyte, in dem sich das erste Stringzeichen nach *LINE befindet.

Die Tabelle zeigt den Zustand der drei CPU-Register beim Aufruf der Routine, auf die der Inhalt von USERV zeigt. A enthält entweder Null oder Eins und zeigt damit an, welcher der beiden Befehle von USERV angesprochen wurde. Die Werte von X und Y hängen davon ab, ob Sie *CODE oder *LINE eingesetzt haben. *CODE 3,2 ruft daher die Routine auf, auf die USERV mit 0 in A, 3 in X und 2 in Y zeigt, wobei die beiden aufgeführten Parameter an diese Routine übergeben werden.

Einfacher *CODE-Befehl

Unser Beispielprogramm zeigt den Ablauf eines einfachen *CODE-Befehls. Die Maschinencoderoutine ist von Adresse &C00 an assembliert (das Ergebnis der Zuordnung in Zeile 40). Die Ganzzahlvariable P% wird dabei auf die gleiche Weise in den Befehlszähler (PC) geladen, wie A%, X% und Y% in die Regi-

Neu definierte Befehle (USERV)

```
10 USERV=&200
20 ?USERV=&00
25 ?(USERV+1)=&0C
30 FOR I%=0TO2STEP2
40 P%=&C00
50 [ OPT I%
60    CMP #0
70    BNE notcode
80    TXA
90   .loop
95    JSR &FFE3
100   DEY
110   CPY #0
120   BNE loop
130   RTS
140  .notcode
145   RTS
150 ]:NEXT I%
160
200 FOR rep=1 TO 10
210 FOR asc=33 TO 48
220 *CODE asc,rep
250 NEXT:NEXT
```

ster A, X und Y. Durch Setzen des niederwertigen Adreßbytes (&00) auf das Vektorbyte &200 und des höherwertigen Adreßbytes (&0C) auf das Vektorbyte &201, zeigt USERV auf diese Routine. Die Routine stellt die angegebene Zeichenfolge auf dem Bildschirm dar. Der erste Parameter von *CODE enthält dabei den ASCII-Code des Zeichens und der zweite die Angabe, wie oft das Zeichen wiederholt werden soll.

*LINE hat kein so breites Einsatzspektrum wie *CODE, wird aber auf die gleiche Weise eingesetzt. Ihre Hauptfunktion ist es, Textstrings an den Maschinencode zu übergeben. Diese beiden Befehle sind eine elegante Lösung, wenn nicht viele Parameter übergeben werden sollen.

Zeile 20 des Programms setzt USERV auf die Adresse der Maschinencoderoutine. Die Schleife zwischen Zeile 90 und 120 gibt das Zeichen, dessen ASCII-Code im A-Register liegt, Y-mal auf dem Bildschirm aus. Wird die Routine über *LINE angesprochen, fangen Zeile 60 und 70 diesen Fall ab und beenden die Routine. Zeile 200 bis 250 geben *CODE mit unterschiedlichen Parametern aus.

OSRDCH-Routine

Die Kommunikation mit einem Microcomputer wird hauptsächlich per Tastatur und Bildschirm abgewickelt. Wie steuert nun das OS diese beiden wichtigen Peripheriegeräte?

Wir untersuchen zunächst einen OS-Aufruf, mit dem wir Zeichen aus dem aktuellen Eingabekanal (Current Input Stream) lesen können. Die entsprechende Routine hat den Namen OSRDCH, liegt bei Adresse &FFE0 und wird über die Vektoren in &210 und &211 angesprochen. Sie empfängt über den Eingabekanal einzelne Zeichen. Wir werden uns daher zunächst ansehen, wie der Eingabekanal überhaupt gewählt wird. Es gibt zwei Haupteingabekanäle: die Tastatur und die RS423-Schnittstelle. Sie werden über OSBYTE oder *FX angewählt. Die folgende Tabelle zeigt diesen Befehl im Maschinencode und in BASIC:

Die Auswahl des Eingabekanals

n	Tastatur	RS423	Assembler	BASIC
0	✔	✘	LDA #2	*FX2,n
1	✘	✔	LDX #n	
2	✔	✔	JSR &FFF4	

*FX2,1 schaltet die Tastatur ab und aktiviert die RS423 als Eingabekanal. Die per RS423 empfangenen Daten werden nun wie Tastatureingaben behandelt. In der Assemblerversion führt der Wert 1 für die Variable n die gleiche Aufgabe aus: Der Rechner betrachtet die Daten via Schnittstelle als Tastatureingaben.

Der gewählte Eingabekanal läßt sich nun mit OSRDCH ansprechen. OSRDCH ist eigentlich nur für Assemblerprogramme nötig, da BASIC mit GET und INPUT schon über ausreichende Eingabemöglichkeiten verfügt. Nach dem Aufruf der Routine bei Adresse &FFE0 und dem Rücksprung befindet sich das vom Eingabekanal gelesene Zeichen im Register A. Wenn während des Lesens ein Fehler erkannt wurde, steht das Übertragsflag auf Eins, anderenfalls auf Null. Wenn also nach dem Rücksprung von OSRDCH das Übertragsflag C auf 1 steht, ist der in A gespeicherte Code nicht korrekt. Beim Einlesen per Tastatur löst das Drücken der Escape-Taste einen Fehler aus. Das A-Register enthält dann den Wert 27 (der ASCII-Wert für Escape). Dieser Fehler muß abgefangen werden, da das OS des Acorn B vom Programm eine Bestätigung erwartet. Doch zum Glück gibt es einen Weg.

Die Bestätigung geschieht durch den Aufruf von OSBYTE mit A=126. Damit werden verschiedene Teile des OS-Arbeitsbereiches gelöscht. Der BASIC-Interpreter erledigt diese Bestätigung nach jedem Escape natürlich automatisch. Die folgende Routine liest den Eingabekanal und reagiert erfreulicherweise auch auf ein Escape:

```
1000  .input  JSR &FFE0
1010          BCS error
1020          RTS
1030  .error  CMP#27
1040          BNE out
1050          LDA#126
1060          JSR&FFF4
1070  .out    RTS
```

Zeile 1000 ruft die Routine OSRDCH auf, und Zeile 1010 prüft das Übertragsflag. Ist es gelöscht, wird ein RTS zum aufrufenden Programm weitergeleitet – mit dem korrekt gelesenen Zeichen des A-Registers. Ist das Flag gesetzt, prüft Zeile 1030, ob der Fehler von einem Escape ausgelöst wurde. Falls ja, rufen Zeile 1050 und 1060 OSBYTE auf und bestätigen den Fehler. Auch für die String-Eingabe brauchen Sie keine eigene Assemblerroutinen zu entwickeln. Mit einer der OSWORD-Routinen lassen sich Strings vom Eingabekanal lesen. Wir werden OSWORD jetzt einsetzen, gehen aber später noch genauer darauf ein, um alle Unsicherheiten zu beseitigen.

Die OSWORD-Routinen liegen bei der Adresse &FFF1. Da es mehrere gibt, müssen wir vor dem Aufruf einen Wert in das A-Register stellen. Bei jedem Aufruf von OSWORD zeigen die Register X und Y auf einen Speicherblock („Control-Block"), der die Parameter für die Routine enthält. Das X-Register speichert dabei das niederwertige Byte der Control-Block-Adresse und das Y-Register das höherwertige. Die folgende Tabelle soll Ihnen den Aufbau verdeutlichen.

Der Control-Block von OSWORD	
Control-Block Eintrag	**Funktion**
0	Niederwertiges Adreßbyte, auf das die Zeichen geschrieben werden sollen
1	Höherwertiges Adreßbyte, auf das die Zeichen geschrieben werden sollen
2	Maximale Zeilenlänge
3	niedrigster akzeptabler ASCII-Wert
4	höchster akzeptabler ASCII-Wert

Bei der Zeicheneingabe über diese Routine behält die Delete-Taste ihre normale Funktion. Die Routine kann über Escape oder Return verlassen werden. Der Control-Block könnte nach einem Aufruf von OSWORD folgende Werte enthalten:

Beispiel für einen Control-Block von OSWORD	
Control-Block Eintrag	**Wert**
0	&00
1	&0C
2	&07
3	32
4	96

- Das erste Eingabezeichen ist in &C00 gespeichert, das zweite in &C01 etc.
- Es werden nur sieben Zeichen angenommen. Wenn Sie versuchen, mehr Zeichen einzulesen, wird ein akustisches Signal ausgegeben, und alle weiteren Zeichen werden ignoriert.
- Nur Zeichen mit ASCII-Codes zwischen 32 (Leerzeichen) und 96 (das $-Zeichen) werden angenommen, alle anderen zurückgewiesen.

Sie sehen, wie sich mit dieser Routine unerwünschte Zeichen ausfiltern lassen. Bei Beendigung der Routine gibt das C-Flag die Information aus, warum die Routine abgebrochen wurde. Bei C=1 wurde Escape gedrückt, bei C=0 die Return-Taste. Das Y-Register enthält die Länge des eingegebenen Strings zuzüglich des ASCII-Wertes für Return – falls die Zeichenkette damit beendet wurde. Beachten Sie, daß Sie diese Routine auf beiden Eingabekanälen verwenden können.

Die Eingabe von Daten in den Acorn B ist also recht einfach. Sehen wir uns nun an, wie Zeichen in den aktuellen Ausgabekanal (Current Output Stream) gesandt werden. Auch hier muß der Kanal über eine OS-Routine (der Befehl *FX3,n) ausgewählt werden. Jedes Bit von n steuert einen anderen Ausgabekanal an. So schaltet beispielsweise *FX3,1 die serielle Bildschirm- und Druckerausgabe an und ermöglicht die Ausgabe über SPOOL.

Parametertabelle zur Steuerung der Ausgabekanäle						
Bit	**0**	**1**	**2**	**3**	**4**	**6**
1	RS423 **AN**	Bildschirm **AUS**	Drucker **AUS**	Drucker **AN***	gespoolt **AUS**	Drucker **AUS****
0	**AUS**	**AN**	**AN**	**AUS***	**AN**	**AN****

*Drucker ist an oder aus, unabhängig davon, ob der Drucker auf andere Weise ausgeschaltet wurde.

**Drucker ist aus, wenn vor dem Zeichen kein CHR$(1) steht

Die Hauptroutine zum Senden von Zeichen an den Ausgabekanal heißt OSWRCH. Sie liegt bei &FFEE und wird über die Vektoren bei &20E und &20F aufgerufen. Ihr Einsatz ist einfach: Das A-Register muß nur mit dem ASCII-Code des Zeichens geladen und die Routine aufgerufen werden. Die folgenden drei Routinen stellen das Zeichen „A" auf dem Bildschirm dar:

```
1000 VDU 65
1000 PRINT (CHR$(65)
1000 LDA #65
1010 JSR &FFEE
```

Der BASIC-Befehl VDU hat eine ähnliche Wirkung wie OSWRCH. Mit Ausnahme des ASCII-Codes 127 (Delete) lassen sich alle Zeichen für die Codes zwischen 32 und 255 auf dem Bildschirm ausgeben. Die Zeichen im Bereich zwischen 0 und 31 haben jedoch Spezialfunktionen. Mit diesen Codes können wir per OSWRCH Grafik auf den Bildschirm bringen, die Befehle COLOUR und GCOL ausführen, Zeichen definieren oder den 6845-Chip steuern, der die Bildschirmanzeige des Acorn B kontrolliert.

Das Senden von Zeichen per OSWRCH wird oft auch als „Schreiben an die VDU-Treiber" bezeichnet. Die Tabelle der ASCII-Steuerzeichen zeigt, welche Wirkungen die Codes zwischen 0 und 31 haben, wenn sie an die VDU-Treiber gesandt werden. Es stehen damit alle Grafikfunktionen, die von BASIC aus möglich sind, auch im Maschinencode zur Verfügung.

Die OSWRCH-Routinen können alle normalen Grafikbefehle ansprechen. Unser zweites Programmbeispiel bringt eine rote Linie auf den Bildschirm. Die Zeilen 50 bis 75 setzen mit dem Befehl GCOL 0,1 die Farbe der Zeile auf Rot. Die Zeilen 90 bis 150 führen PLOT 5,100,100 aus (entspricht DRAW 100,100). Zeile 100 sendet den PLOT-Modus (in diesem Fall 5) an die VDU-Treiber, gefolgt von einer X-Koordinate (zwei Bytes im Lo-Hi-Format) und einer Y-Koordinate (zwei Bytes im Lo-Hi-Format). Wenn die 5 durch eine 4 ersetzt wird, läßt sich ein MOVE-Befehl ausführen (MOVE entspricht dem Befehl PLOT 4,x,y). Auch andere Grafikbefehle – wie PLOT 85 für das Zeichnen von Dreiecken – lassen sich auf diese Weise ausführen. Dabei ist zu beachten, daß die VDU-

Grafik mit OSWRCH

```
10 MODE 1
20 FOR I%=0TO2STEP2
30 P%=&C00
40 [OPT I%
50 LDA #18
55 JSR &FFEE
60 LDA #0
65 JSR &FFEE
70 LDA #1
75 JSR &FFEE
80
90 LDA #25
95 JSR &FFEE
100 LDA #5
105 JSR &FFEE
110 LDA #0
115 JSR &FFEE
120 LDA #100
125 JSR &FFEE
130 LDA #0
135 JSR &FFEE
140 LDA #100
145 JSR &FFEE
150 RTS
160 ]:NEXT I%
170 CALL &C00
```

Treiber fünf Bytes erwarten, wenn PLOT den Wert 25 als erstes Byte sendet. Werden diese Bytes nicht empfangen, können unerwünschte Effekte entstehen. Das trifft auf alle Vorgänge der VDU-Treiber zu, die mehr als ein Byte benötigen.

VDU23 ist ein weiterer Befehl für die VDU-Treiber. Mit ihm lassen sich neue Zeichen definieren.

```
VDU23,224,255,255,255,255,255,255,255,255
```

legt beispielsweise das (normalerweise nicht definierte) Zeichen 224 als Block an. Der Befehl gibt dem Anwender die Möglichkeit, die Zeichen 224 bis 255 in den Modi 0 bis 6 neu festzulegen. In Verbindung mit einer der OSBYTE-Routinen kann VDU23 auch Zeichen ändern. Alle VDU23-Aufrufe, die vom OS nicht erkannt werden (beispielsweise VDU23,0...) werden über einen Spezialvektor bei &226 und &227 geleitet. Wenn Sie die Adresse dieses Vektors verändern, können Sie eigene VDU23-Routinen einsetzen.

Über VDU23 läßt sich auch der 6845-Video Steuerchip ansprechen. Der Befehl hat folgendes Format:

```
VDU23,0,Register,Wert,0,0,0,0,0,0
```

wobei „Register" das angesprochene Register des 6845 bezeichnet und „Wert" den Wert, der dorthin geschrieben werden soll. Das folgende Programm verändert mit VDU23 zwei der Register des 6845, die dem Chip mitteilten, welcher RAM-Bereich als Bildschirmspeicher eingesetzt werden soll. Das Programm legt den Anfang des Video-RAMs auf die Adresse &0000 und stellt den Arbeitsbereich des Acorn B auf den Bildschirm dar. Setzen Sie dort probeweise einmal einige Codezeilen, Array-Dimensionierungen, etc. ein. Die Routine ist zwar in BASIC geschrieben, läßt sich aber leicht in Assembler übersetzen.

```
10 MODE 0
20 VDU23,0,12,0,0,0,0,0,0
30 VDU23,0,13,0,0,0,0,0,0
40 VDU28,0,10,30,0:REM Textfenster anlegen
50 CLS
```

Zwei weitere OS-Aufrufe beziehen sich auf OSWRCH: OSNEWL und OSASCII. OSNEWL (bei &FFE7) bringt einen Zeilenvorschub und ein Return auf den Bildschirm, während jedoch OSASCII (bei &FFE3) eine Variante von OSWRCH ist und sich für die Textverarbeitung einsetzen läßt. Wenn über diese Routine das Zeichen 13 (Return) geschrieben wird, wird automatisch auch ein Zeilenvorschub (Zeichen 10) ausgegeben. Für an die VDU-Treiber gesandte Grafiktabelle dürfen Sie diese Routine nicht einsetzen, da ein zusätzliches CHR$(10) den Bildschirmaufbau beeinflussen kann.

Mit *SPOOL und *EXEC schließlich lassen sich Ein- und Ausgaben in das aktuelle Dateisystem stellen. *EXEC dateiname bewirkt, daß eine Datei mit dem angegebenen Dateinamen eröffnet und deren Inhalt eingelesen wird. *SPOOL schreibt Zeichen in die im Befehl angegebene Datei.

Damit endet unsere Einführung in das Betriebssystem des Acorn B. In den nächsten Folgen werden wir einige Maschinencoderoutinen untersuchen, mit denen sich die Bildschirmausgabe des Commodore 64 wesentlich verbessern läßt.

Tabelle der ASCII-Steuerzeichen

CODE	Zusatz-Byte	Beschreibung
0	0	Null
1	1	Nächstes Zeichen zum Drucker senden
2	0	Drucker anschalten
3	0	Drucker ausschalten
4	0	Text/Grafikcursor trennen
5	0	Text/Grafikcursor verbinden
6	0	VDU-Treiber einschalten
7	0	kurzes Klingelzeichen
8	0	Rückwärtsschritt des Cursors
9	0	Vorwärtsschritt des Cursors
10	0	Cursor eine Zeile nach unten
11	0	Cursor eine Zeile nach oben
12	0	Textbereich löschen
13	0	Return
14	0	Seitenschaltung ein
15	0	Seitenschaltung aus
16	0	Grafikbereich löschen
17	1	Textfarbe definieren
18	2	Grafikfarbe mit nächstem Byte definieren
19	5	logische Farbe definieren
20	0	logische Farbe auf Systemeinstellung stellen
21	0	VDU-Treiber abschalten
22	1	Darstellungmodus mit nächsten Byte auswählen. 22 und 7 schalten Modus 7 ein. HIMEM wird nicht verändert
23	9	dargestelltes Zeichen umdefinieren
24	8	Grafikfenster definieren
25	5	PLOT-Befehl ausführen
26	0	Text- und Grafikfenster zurückstellen
27	0	Null
28	4	Textfenster definieren
29	4	Grafikursprung definieren
30	0	Textcursor nach links oben stellen
31	2	Textcursor auf die Koordination x,y der nächsten zwei Bytes stellen. 31,10,10 setzt den Cursor beispielsweise auf die Position 10,10

Abgeräumt

In diesem Artikel stellen wir Ihnen „Snooker" von Visions vor. Die Simulation eines Billardspiels wird für den Acorn B, Commodore 64 und Spectrum angeboten.

Für professionelle Spieler bedeutet die Computer-Version des Billard-Spiels ein Sakrileg. Wie sollen beispielsweise Kugellauf und die Varianten des Tischaufbaus quantifiziert und programmiert werden? Und doch kann Billard als eine recht einfache Darstellung von Kraftwirkungen betrachtet werden. Diese Brechnungen können vom Computer ausgeführt werden.

Als Auftakt erscheint bei dem Programm „Snooker" (von Visions) eine Aufsicht des Tisches, auf dem die Kugeln durch farbige Kreise dargestellt sind. Man zielt auf die weiße Kugel durch Bewegen eines Kreuzes auf dem Bildschirm und definiert so den Weg, den die Kugel nehmen soll. Die weiße Kugel wird dann auf das Ziel gelenkt, wobei die Stoßstärke durch die Dauer des Tastendruckes bestimmt wird. Fingerspitzengefühl ist nötig.

Bereits nach wenigen Spielzügen werden einem erfahrenen Spieler zwei wesentliche Probleme deutlich. Zunächst kann der Computer die Winkel und den Effekt nicht genau genug berechnen, um überzeugende Ergebnisse auf den Bildschirm zu bringen. Daraus folgert, daß manche Stöße zu unvorhersehbaren Ergebnissen führen (obwohl das in einem wirklichen Spiel zuweilen auch geschehen kann).

Das andere Problem, das in der Praxis eine weitaus größere Bedeutung hat, ist der Blick auf den Tisch. Will man eine Kugel stoßen, kann man nicht über das Queue anvisieren. Da der Bildschirm zudem leicht gebogen ist, ist das Einschätzen von Winkeln schwer. Anfangs sind deshalb die Spielversuche recht frustrierend, doch Übung macht Meister.

Mit scharfem Schnitt

Hat man sich aber einmal damit abgefunden, daß das Spiel recht wenig mit echtem Billard zu tun hat und die Besonderheiten verstanden, wird das Programm doch zu einem recht interessanten strategischen Spiel. Dann erst nimmt man subtile Programmdetails wahr, wie etwa die Möglichkeit, der Kugel einen besonderen Effet zu geben, wobei die Ergebnisse wiederum unberechenbar sind.

Die drei Versionen des Programms unterscheiden sich darin, daß sie die Stärken und Schwächen der Rechner, auf denen sie gespielt werden, widerspiegeln. Das Programm benötigt überraschend wenig Speicherkapazität und ist deshalb schnell zu laden – selbst in den Commodore, der ja für seine Ladeträgheit bekannt ist. Folglich muß die Qualität der einzelnen Versionen an den Grafikmöglichkeiten der Rechner gemessen werden.

In dieser Hinsicht schneidet der Spectrum am schlechtesten ab. Die Darstellung des Tisches ist klein, und die Kugeln sind nicht in ihren tatsächlichen Farben abgebildet. Die Commodore-Version bietet realistische Farben, und die Kugeln sind maßstabsgerechter. Die Punktzahl ist in dieser Version deutlicher illustriert, und vor Spielbeginn wird eine Demo-Option geboten. Die beste Spielversion ist die für den Acorn B. Die Darstellung des Tisches erstreckt sich über die gesamte Bildschirmbreite, das Queue läßt sich erstaunlich exakt steuern.

Das Spiel kann von einer oder zwei Personen gespielt werden, wobei die Ein-Personen-Version zum Üben gedacht ist. Gesamteindruck des Spiels: Es ist sein Geld wert, doch die Speicherkapazität der Rechner hätte besser genutzt werden können. Trotz des Mangels an Realität: Wer hat schon genug Platz für einen echten Billardtisch . . .

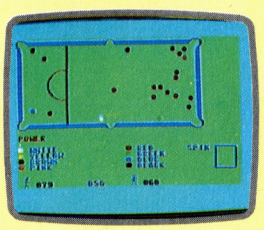
Snooker:
Für Commodore 64, 16 K Spectrum und Acorn B
Hersteller: Visions (Software Factory) Ltd., 1 Felgate Mews, Studland Street, London W6 9JT
Autor: Tim Bell (Acorn-Version von Andy Williams)
Joystick: Option
Programm: Cassette

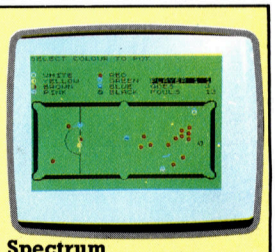
Wichtigstes Element beim Billard sind genaues Zielen und exakte Stoßeinschätzung. In Visions' „Snooker" muß ein Zielkreuz auf dem Tisch positioniert werden, wogegen die Stoßstärke durch Festhalten und Loslassen des Actionknopfes bestimmt wird.

Von Ecke zu Ecke

Acorn B

Commodore 64





Spectrum

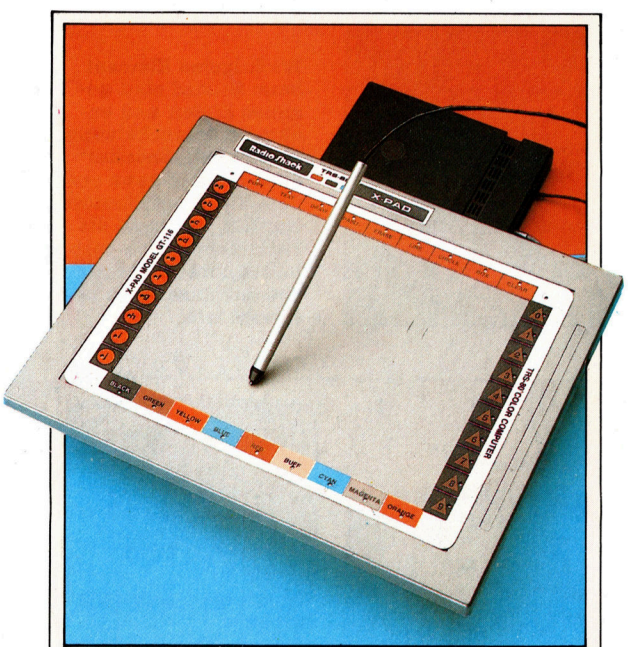
Tandy Color Computer

Der Tandy Color Computer ähnelt vom Konzept her dem Dragon 32. Beide Rechner arbeiten mit 6809E Prozessor von Motorola.

Trotz aller äußeren Unterschiede steht der Tandy Color Computer funktionsmäßig den Dragon-Rechnern so nahe, daß viele Programme ohne weiteres zwischen den beiden Rechnern austauschbar sind.

Es gibt aber auch viele Hardware-Gemeinsamkeiten: Beide Rechner sind mit dem gleichen Modulsteckplatztyp ausgestattet und arbeiten mit dem gleichen Prozessor. Der 6809E von Motorola ist eine außerordentlich leistungsfähige CPU, die sonst bei Heimcomputern nur selten zu finden ist und in gewisser Hinsicht auch beim Tandy nicht ganz in das übrige Konzept paßt – der 6809 ist hier mit einer Taktfrequenz von 895 kHz eigentlich nicht ausgelastet.

Der Rechner ist mit einer speziellen BASIC-Version (Tandy Color BASIC) ausgerüstet, die trotz der niedrigen Taktfrequenz schnell arbeitet. Das liegt an der ausgereiften CPU, die über



Tandy-Grafiktablett
Das Grafiktablett GT-116 ist als Digitalisierer für das Abtasten von Grafiken und Bildern wie auch als Auswahlgerät für die Menüsteuerung geeignet.

Tastatur
Sie besteht aus 53 Tasten. Die Cursorsteuerungstasten befinden sich links und rechts der Tastatur.

Joystick-Ports
Zum Anschluß von zwei Steuerknüppeln. Über diese Schnittstellen können auch Analogsignale von anderen Peripherien verarbeitet werden.

Netzschalter

Netzteil-Komponenten
Für die Gleichrichtung der Transformator-Ausgangsspannung. Die freigesetzte Verlustwärme wird über Kühlkörper an die Umgebung abgeführt.

Transformator

RS232-Schnittstelle
Zum Anschluß von Druckern, Modems und anderen seriellen Geräten.

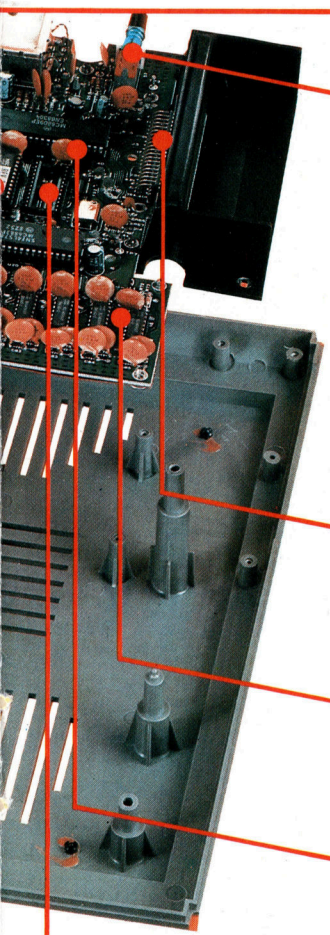
Cassettenrecorder-Buchse

Kanalumschaltung
Der Schalter hat zwei Einstellmöglichkeiten.

Tastatur-Anschlußstecker
Hier wird die Platine mit der Tastatur verbunden.

Video-Controller 6847
Die Programmierung von Ausgabeformat und Farben ist in BASIC und im Maschinencode möglich.

8-KByte-ROM
Enthält das Tandy-Color-BASIC.



Video-Taktgeber
Dieser Schwingquarz steuert die Bildschirmausgabe.

Reset-Taste
Für den Warmstart des Rechners.

Modulsteckplatz
Für ROM-Cartridges und zum Anschluß von Diskettenstationen.

RAM
Die 16 K Bit verteilen sich auf acht Chips.

6809E – CPU
Die CPU arbeitet mit interner 16-Bit-Arithmetik und externem 8-Bit-Datenbus.

Leersockel für ROM-Chips

eine Reihe spezieller Befehle für die Hardware-Aussteuerung verfügt. Als Video-Chip besitzt der Rechner einen 6847-Steuerbaustein, der frei programmierbar ist, aber generell nur 16 Zeilen à 32 Zeichen erzeugt. Normalerweise erscheint auf dem Bildschirm schwarze Schrift auf grünem Grund. Sie können jedoch für Vorder- oder Hintergrund auch eine andere der neun verfügbaren Farben wählen. Im Grafikmodus beträgt die Auflösung 32x64 Pixel (acht Farben bzw. 192x256 Pixel).

Für die Tonerzeugung steht nur ein Kanal zur Verfügung, und somit lassen sich lediglich sehr einfache Soundeffekte programmieren.

Der Tandy Color hat den gleichen Modulschacht wie der Dragon. Es steht ein umfangreiches Angebot an ROM-Cartridges zur Verfügung – nicht nur die üblichen Spiele wie Schach, Dame und Flipper, sondern auch Lernprogramme für Musik, Rechnen und Maschineschreiben. Außerdem wird Software für Finanzplanung, Archivierung und Textverarbeitung wie auch Programmierhilfen für BASIC und Maschinencode angeboten, ferner ein Dignose-ROM, mit dem sich das Funktionieren der Maschine überprüfen läßt.

Über eine RS232-Schnittstelle kann eine Anzahl von Peripheriegeräten betrieben werden; dieser Anschluß eignet sich in Verbindung mit einem Modem auch für die Kommunikation mit andern Rechnern. Außerdem ist der Rechner mit zwei Joystick-Ports ausgestattet.

Erfreulicherweise sind die Eingänge für Proportional-Joysticks ausgelegt, also nicht nur für die einfachen Schaltermodelle ohne differenzierte Bewegungssteuerung. Daher lassen sich diese Buchsen auch für andere analoge (d. h. stetig veränderliche) Signale verwenden.

Über den Modulschacht ist der Rechner erweiterungsfähig, unter anderem mit bis zu vier 5¼-Zoll-Diskettenlaufwerken, was eine Speicherkapazität von mehr als 600 KByte ergibt.

Tandy Color Computer

ABMESSUNGEN
369 x 344 x 94 mm

TAKTFREQUENZ
0,894 MHz

SPEICHER
16 KByte RAM, erweiterbar auf 32 KByte; 8 KByte ROM

BILDSCHIRMDARSTELLUNG
16 Zeilen zu 32 Zeichen, 9 Farben mit unabhängiger Wahl von Vorder- und Hintergrund. 127 feste und 127 frei definierbare Zeichen.

SCHNITTSTELLEN
RS232-Interface, Cassettenrecorder-Interface, zwei Proportional-Joysticks, umschaltbarer Fernsehantennenausgang für zwei Kanäle

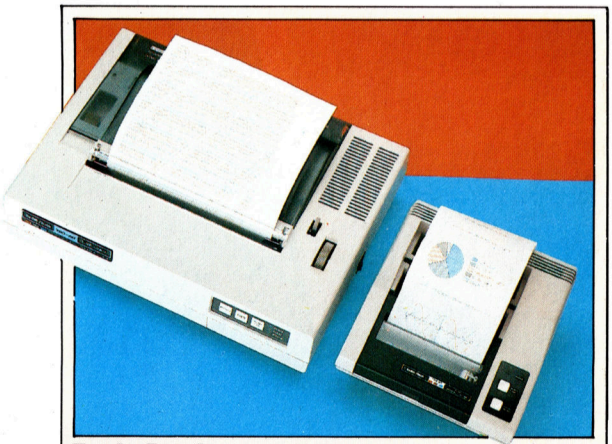
MITGELIEFERTE SPRACHEN
BASIC

WEITERE SPRACHEN
6809-Maschinencode mit Assemblerpaket

STANDARD-ZUBEHÖR
Betriebsanleitung und BASIC-Handbuch, Antennenkabel

TASTATUR
53 Tasten

DOKUMENTATION
Das Benutzerhandbuch enthält alle wichtigen Informationen, so z. B. eine völlig ausreichende Bedienungsanleitung.



Tandy-Drucker
Für den Color Computer gibt es eine ganze Reihe Drucker von Tandy. Der CPG-220 ist ein Tintenstrahldrucker, der sieben Farben wiedergeben kann. Rechts im Bild sieht man den CPG-115 mit vier Kugelschreiberpatronen, die ebenfalls mehrfarbige Ausdrucke und das Zeichnen von Grafiken erlauben.

Leuchtspuren

Im letzten Artikel entwickelten wir das BASIC-Programm „Lichtreiter" für den ZX Spectrum. Im folgenden werden die Versionen für zwei andere Computer, den Commodore 64 und den Acorn B, ausführlich vorgestellt.

Im Gegensatz zu den BASIC-Versionen des Spectrum und Acorn B verfügt das Commodore-BASIC über keine Befehle, mit denen man einzelne Pixel PLOTten kann. In der hier gezeigten Version des Spiels verwenden wir die niedrige Auflösung, um die Lichtspuren zu zeichnen. Dazu wird mittels POKE ein inverses Leerzeichen benutzt, dessen Wert in den Bildschirmspeicher gePOKEt und die Farbe in der entsprechenden Farbadresse angegeben wird.

Ähnlich wie bei der Spectrum-Version wurde auch beim Commodore-Programm nach Möglichkeit auf Unterprogramme verzichtet. An den Punkten des Spieles, an denen Tempo unwichtig ist, wurden jedoch Unterroutinen für die Punktzahl und die Bildschirmdarstellung eingefügt.

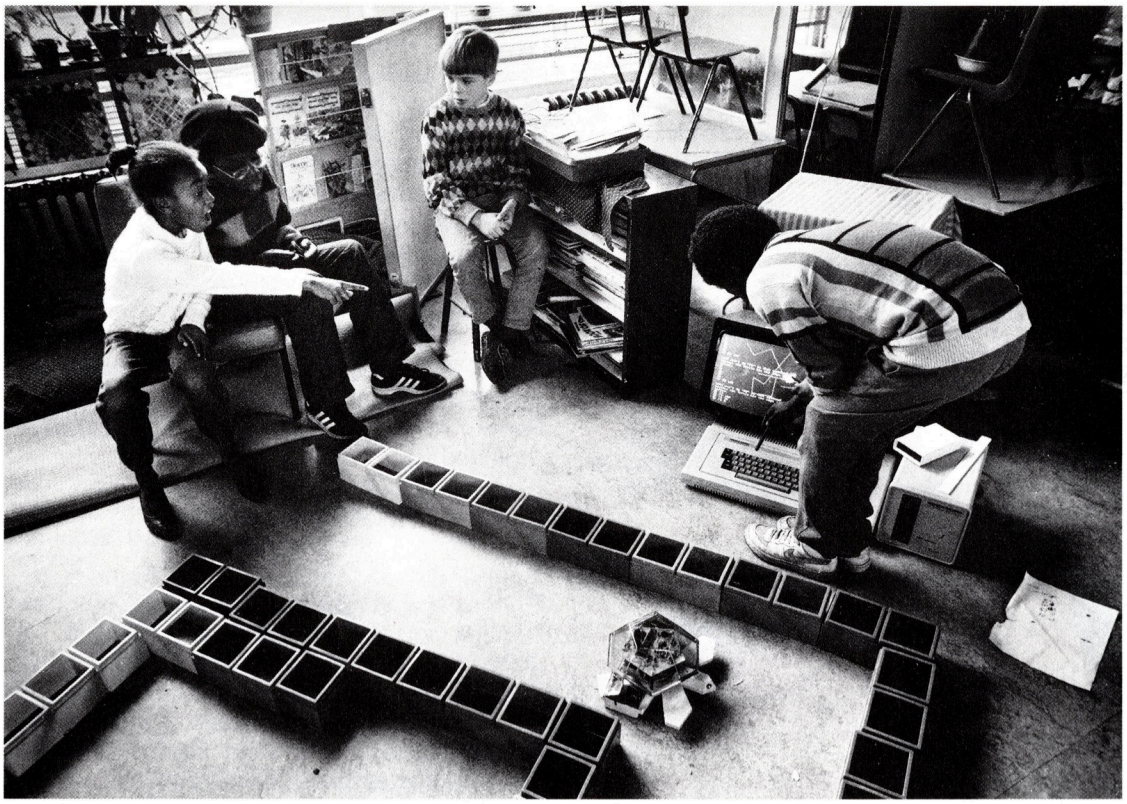
Da das BASIC des Acorn B erheblich schneller ist als das des Spectrum oder Commodore und es den Aufruf von strukturierten Prozeduren gestattet, ist die Acorn-B-Version sehr übersichtlich. Bei den meisten BASIC-Versionen reduziert die Strukturierung durch Unterroutinen die Programmgeschwindigkeit recht erheblich. Das liegt daran, daß bei jedem Aufruf das gesamte Programm nach der ersten Zeile der entsprechenden Routine durchsucht werden muß. Im Gegensatz dazu vermerkt das Acorn-B-BASIC beim ersten Aufruf einer Prozedur deren Position in einer Tabelle.

Commodore 64

```
10 REM CBM 64
15 POKE53281,0:POKE53280,4:REM SCR/BORD COLOUR
20 SC=1024:CO=55296
25 PRINTCHR$(147):REM CLEAR SCREEN
30 X1=2:Y1=12:X2=37:Y2=12
35 DX=1:DM=-1:DN=0:DY=0
40 PRINTCHR$(19)CHR$(158)"PLAYER 1:"S1
50 PRINTCHR$(19)TAB(27)"PLAYER 2:"S2
52 FORI=1TO8:PRINTCHR$(17);:NEXT
54 PRINTTAB(14)"PRESS A KEY"
55 GETJ$:IFJ$<>""THEN55
56 GETA$:IFA$=""THEN56
57 PRINTTAB(14)CHR$(145)"
60 REM MAIN LOOP
70 GETA$
80 IFA$="W"THENDY=-1:DX=0
90 IFA$="X"THENDY=1:DX=0
100 IFA$="A"THENDX=-1:DY=0
110 IFA$="D"THENDX=1:DY=0
120 IFA$="Q"THENDN=-1:DM=0
130 IFA$="M"THENDN=1:DM=0
140 IFA$="J"THENDM=-1:DN=0
150 IFA$="L"THENDM=1:DN=0
155 Y1=Y1+DY
156 IFY1<1ORY1>24THENF=0:GOSUB1000:GOTO25
157 X1=X1+DX
158 IFX1<0ORX1>39THENF=0:GOSUB1000:GOTO25
160 Y2=Y2+DN
162 IFY2<1ORY2>24THENF=1:GOSUB1000:GOTO25
164 X2=X2+DM
166 IFX2<0ORX2>39THENF=1:GOSUB1000:GOTO25
167 P1=X1+40*Y1
168 P2=X2+40*Y2
170 IFPEEK(SC+P1)=160THENF=0:GOSUB1000:GOTO25
180 IFPEEK(SC+P2)=160THENF=1:GOSUB1000:GOTO25
190 POKE SC+P1,160
200 POKE CO+P1,14
210 POKE SC+P2,160
220 POKE CO+P2,5
230 GOTO70:REM RESTART LOOP
240
1000 REM SCORE S/R
1020 IF F=1THENS1=S1+1:GOSUB2000:RETURN
1030 S2=S2+1:GOSUB2000:RETURN
1999
2000 REM FLASH SCREEN S/R
2010 FORJ=1TO10
2020 FORI=0TO15
2030 POKE53281,I
2040 NEXTI,J
2045 POKE53281,0
2050 RETURN
```

Acorn B

```
10  REM BBC
20 MODE1:cycle1=0:cycle2=0
30 PROCinitialise
40 PROCkey
50 PROCborder
60 PROCscore
70 PROCplot
80 PROCkeyboard
90 PROCtest_point
100 GOTO70
110 END
120 DEF PROCborder
130 GCOL0,border
140 MOVE0,0
150 DRAW1279,0
160 DRAW1279,980
170 DRAW0,980
180 DRAW0,0
190 ENDPROC
200 DEF PROCkeyboard
210 REM PLAYER ONE
220 IF INKEY(-51)=-1THEN dx=4:dy=0
230 IF INKEY(-87)=-1THEN dm=4:dn=0
240 IF INKEY(-66)=-1THEN dx=-4:dy=0
250 IF INKEY(-70)=-1THEN dm=-4:dn=0
260 IF INKEY(-67)=-1THEN dx=0:dy=-4
270 IF INKEY(-34)=-1THEN dx=0:dy=4
280 IF INKEY(-55)=-1THEN dm=0:dn=4
290 IF INKEY(-102)=-1THEN dm=0:dn=-4
300 x=x+dx:y=y+dy:m=m+dm:n=n+dn
310 ENDPROC
320 DEF PROCinitialise
330 x=100:y=490:m=1179:n=490
340 dx=4:dy=0:dm=-4:dn=0
350 col1=1:col2=2:border=3
360 ENDPROC
370 DEF PROCtest_point
380 pixel1=POINT(x,y)
390 pixel2=POINT(m,n)
400 IF pixel1 THEN PROCexplode(x,y,1)
410 IF pixel2 THEN PROCexplode(m,n,2)
420 ENDPROC
430 DEF PROCplot
440 GCOL0,col1:PLOT69,x,y
450 GCOL0,col2:PLOT69,m,n
460 ENDPROC
470 DEF PROCexplode(ex,ey,which)
480 FORI=1TO100
490 MOVEex,ey
500 GCOL0,RND(3)
510 PLOT1,RND(50)-25,RND(50)-25
520 NEXT
530 IF which=2 THEN cycle1=cycle1+1
540 IF which=1 THEN cycle2=cycle2+1
550 PROCinitialise
560 CLS
570 PROCkey
580 PROCborder
590 PROCscore
600 ENDPROC
610 DEF PROCscore
620 PRINTTAB(1,0)"Cycle One=";cycle1
630 PRINTTAB(28,0)"Cycle Two=";cycle2
640 ENDPROC
650 DEF PROCkey
660 PRINTTAB(8,12)"PRESS ANY KEY TO START"
670 *FX21
680 A$=GET$
690 PRINTTAB(8,12)"
700 ENDPROC
```



Kinder an einer Grundschule in Southmead (England) werden mit den Grundlagen der Geometrie durch diese Übung mit der „Turtle" vertraut gemacht. Sie spielen eine aktive Rolle, indem sie ein Labyrinth aufbauen und die Turtle so programmieren, daß sie den Hindernissen ausweicht. Die Übung lehrt die Grundlagen der Winkelveränderung und andere abstrakte Konzepte.

Bessere Leistung

Durch den Einsatz von Bodenrobotern im Schulunterricht können auch junge Schüler abstrakte mathematische Probleme durch ein dreidimensionales, sich bewegendes Objekt verstehen. Neben Lernspaß bringen Roboter neue und effektive Lernmethoden.

Die Verwendung von Robotern in der Schule stellt den Versuch dar, abstrakte Konzepte auch ohne ein konkretes Modell zu vermitteln. So können zum Beispiel komplexe trigonometrische Aufgaben anschaulich dargestellt werden. Denkprozesse lassen sich durch dreidimensionale Bewegungen einer Turtle verdeutlichen, die auch dazu benutzt werden kann, Grundlagen der Computerprogrammierung zu vermitteln.

In der Vergangenheit wurde Geometrie mit Bleistift, Lineal und Dreieck gelernt. Ein achtjähriges Kind kann jedoch mit Winkelwerten wenig anfangen, von grundlegenden geometrischen Konzepten einmal ganz abgesehen. Andererseits gewinnen Winkel an Wichtigkeit, wenn sie bestimmen, ob eine vom Schüler programmierte Turtle den Weg durch ein „Plastikflaschen"-Labyrinth auf dem Boden des Klassenzimmers schafft oder nicht. Geometrie mag vielleicht auf diese Weise nicht verstanden werden, doch zumindest erfolgt ein Motivationsanstoß, um die praktischen Anwendungsmöglichkeiten zu verstehen.

Einer der Grundaspekte der „Turtle-Technologie" ist, daß man durch Probieren und Fehler lernt. Ein Kind kann ein Programm solange ändern, bis die gewünschte Turtlebewegung erfolgt, und es beschäftigt sich dabei ständig mit Winkeln, Linien und Maßeinheiten. Ein Fehler bedeutet lediglich, daß etwas anderes versucht werden muß, damit die gewünschte Bewegung stattfindet. Das Programmieren eines Roboterarms, der etwas heben und einen Gegenstand bewegen soll, setzt das Denken in drei Bewegungsebenen voraus. Sich so etwas auf Papier vorzustellen, ist nicht gerade leicht. Mit einem Roboterarm hingegen werden die Bewegungen deutlich sichtbar. Die Programmierergebnisse sind sofort zu überprüfen, Fehler können einfach beseitigt und entsprechende Korrekturen vorgenommen werden, bis der Lauf wunschgemäß erfolgt. Der Roboter bietet zudem sofortige Rückkopplung für den Programmierer, demonstriert dabei jeden Winkel und jede Bewegung in einer Abfolge, die weitaus effizienter ist als eine vergleichbare Übung mit Bleistift und Papier.

Ein Roboter-Spielzeug, das besonders in der Kinderausbildung sehr populär ist, wird als

Konkretes Beispiel

Kleine Kinder können abstrakte Zusammenhänge oft nicht erfassen, selbst wenn eine physikalische Demonstration der Prinzipien erfolgt. Eines der bekanntesten Beispiele dafür ist die Aufbewahrung von Mengen und die transitiven Beziehungen. Dabei wird das Kind mit Behältern unterschiedlicher Form konfrontiert, die aber den gleichen Inhalt haben. Im unteren Beispiel enthalten A, B und C gleiche Flüssigkeitsmengen. Die Behälter B und B¹ sind identisch.

Der Lehrer schüttet A in B und fordert das Kind auf, den Flüssigkeitsspiegel in B und B¹ zu überprüfen. Daraufhin wird die Flüssigkeit von B zurück in A gegossen.

Nun führt der Lehrer eine ähnliche Operation durch, diesmal mit den Gefäßen C und B¹

Die Gefäße A und C werden nebeneinander gestellt, und das Kind wird gefragt, ob beide Gefäße denselben Inhalt haben.

Die Gefäße A, B und C werden auf den Tisch gestellt. Die Frage lautet, ob alle drei Behälter dieselbe Flüssigkeitsmenge enthalten. In den beiden letzten Phasen des Experiments wird das Kind aufgefordert, seine Antwort zu begründen – falls sie richtig war. Kinder unter sieben Jahren sind damit meist überfordert. Dieses Experiment führte der Kinderpsychologe Piaget mit Kindern durch. Die Anwendung von Robotern führt die Kinder an abstrakte Konzepte heran, bietet ihnen aber ein konkretes Modell und weist ihnen eine aktive Rolle beim Lernen zu.

„Big Trak" angeboten. Hierbei handelt es sich um ein futuristisches Fahrzeug, das über einen kleinen internen Speicher verfügt und auf dem Dach eine Taschenrechner-Tastatur hat. Pfeiltasten bestimmen die Richtung, in die sich das Fahrzeug bewegt. Diesem Befehl folgt eine Eingabezahl von 1 bis 99. FORWARD 1 bringt „Big Trak" um eine Einheit voran. RIGHT 15 veranlaßt eine Rechtsdrehung um 45 Grad. Bis zu acht Befehle können gleichzeitig gespeichert werden. Das Gerät wird als Turtle-Ersatz benutzt und regt kleine Kinder dazu an, über eigene Programmier-Routinen nachzudenken. „Big Trak" demonstriert unter anderem, daß Befehle in der Reihenfolge ihrer Eingabe befolgt werden.

LOGO für Kinder

Ursprünglich wurden Turtles entwickelt, um Kindern die Programmierung von Computern deutlich zu machen. Seymour Papert, der „Schöpfer" der Turtle, sagt, Kinder sollten Computer programmieren können, aber nicht von Computern programmiert werden. Diese Philosophie wird in der von Papert entwickelten Computersprache LOGO deutlich.

Paul Cheung von der Universität Edinburgh entwickelte die robotische Seite von LOGO. Er sah die Notwendigkeit vielseitigerer Software bereits zu einem Zeitpunkt, als sich die meisten Leute mit der Entwicklung von Hardware beschäftigten. LOGO ist für die Programmierung von Turtles ideal, kann aber nur in begrenztem Umfang für die Programmierung anderer Geräte benutzt werden. Mit LOGO als Grundlage strukturierte Cheung die Sprache neu und machte so ihre Anwendung für andere Bereiche möglich. Zugleich erhöhte er ihre Kapazität, um Eingabedaten von Geräten wie Berührungssensoren, Balkencode-Lesern und Lichtdetektoren empfangen zu können. Ergebnis war das „Concurrent-LOGO" oder C-LOGO, eine Sprache, die zu „paralleler Verarbeitung" imstande ist.

Mit C-LOGO kann man bis zu acht Vorgänge gleichzeitig steuern. Die zwei besonders interessanten Funktionen dabei sind FOREVER und WHENEVER. Die erste, FOREVER, leitet einen Vorgang ein und läßt ihn solange laufen, bis C-LOGO ihn beendet. WHENEVER wartet, bis ein bestimmter Zustand erreicht ist. Daraufhin erfolgt die Aktivierung der angegebenen Funktion. Dies ist Grundlage der Parallel-Verarbeitung. Mit C-LOGO können Schalter betätigt werden. Damit läßt sich aber auch feststellen, ob ein Schalter betätigt wurde. Drückt ein Berührungssensor gegen ein Hindernis, wird ein Schalter aktiviert und der Computer informiert, daß ein Kontakt hergestellt wurde. Kinder können ein Fahrzeug so programmieren, daß es Hindernissen ausweicht, indem mit WHENEVER ein entsprechender Befehl an den Roboter gegeben wird.

Kinder zwischen 13 und 15 Jahren haben Programme für Windmühlen, Lifts, Mechanikarme und Fahrzeuge geschrieben, aber auch Schnittstellen programmiert, die mit einem Schalter verbunden sind, über den sich Tür- und Fensterbewegungen in einem „Roboterhaus" steuern lassen. Das Haus ist mit einer Alarmanlage ausgestattet, die durch das Öffnen von Fenstern aktiviert wird. Türen lassen sich erst nach Eingabe eines Kennwortes in den Computer öffnen. Beide Programme laufen aufgrund der Parallel-Verarbeitung gleichzeitig problemlos ab.

Die drei Etagen des Hauses sind über einen Fahrstuhl zugänglich. Der aus einem Metallbaukasten erstellte Fahrstuhl wird von einem Gleichstrom-Motor mit drei Stufenschaltern angetrieben. Die Kinder programmierten ihn so, daß er durch Drücken eines Knopfes ins Erdgeschoß fuhr, durch Drücken eines zweiten in die erste Etage und eines dritten ins Dachgeschoß. Sie stellten sich selbst die Frage: „Was geschieht, wenn man einen Knopf drückt, während der Fahrstuhl sich bewegt?" und gelangten zu der Antwort, daß alle anderen Signale zu ignorieren seien, bis der Fahrstuhl anhielt. Nach einigem Nachdenken gelangten die Kinder aber zu der Erkenntnis, daß es besser wäre, die Anweisungen im Computer zu speichern und sie in richtiger Reihenfolge ausführen zu lassen.

„Fahrplan" mit FOREVER

Das klingt nach einem komplizierten Programm, ist aber durch den Einsatz der Sprache C-LOGO weit weniger schwierig und für Kinder durchaus möglich. Wichtige Komponenten bei diesem Projekt waren ein Signaldetektor, ein „Fahrplan" und eine Fahrstuhlsteuerung, wobei der Signaldetektor und die Fahrstuhlsteuerung parallel arbeiteten. Durch Verwendung von FOREVER wurden Signale erkannt und an den „Fahrplan" weitergeleitet. Daraufhin „fragte" die Fahrstuhlsteuerung, wohin sie fahren sollte, steuerte die Zieletage an und wartete dann auf die nächste Anweisung. Nachdem die Befehle hintereinander gespeichert worden waren, schickte sie der „Fahrplan" in der entsprechenden Reihenfolge an die Steuerung. Durch die direkt sichtbare Umsetzung der Befehlsfolgen wurden die Kinder dazu angeregt, solange zu programmieren, bis sie die richtige Lösung gefunden hatten.

Viele Pädagogen glauben, daß einer der Vorteile der Roboter-Programmierung darin besteht, daß Kinder ihre Fehler besser erkennen und beseitigen können. Es wird also nicht mehr nach „richtig" oder „falsch" gefragt, sondern das Kind wird zum Nachdenken angeregt und dazu, sich verändernden Situationen richtig anzupassen. Ob Kinder so wesentlich besser lernen, ist jedoch fraglich; eine Antwort hierauf steht noch aus.

Spielen mit der Turtle

Die Rolle der Turtle im Klassenzimmer ist nicht allein auf Geometrie, Trigonometrie und Programmieren beschränkt. Es wurden beispielsweise Labyrinthe konstruiert, um ein Mythologie-Projekt (Theseus und der Minotaurus) erarbeiten zu können. Ein sehr anspruchsvolles Projekt beinhaltete die Konstruktion einer Modellstadt (gebaut im Werkunterricht), in der sich die Turtle zurechtfinden mußte und dabei verschiedene Geschäfte zu besuchen und „Einkäufe" zu tätigen hatte.

Ein anderes Turtle-Spiel ist „Racetrack", wobei jedes Kind mit allen anderen im Wettbewerb steht, um eine einfache Rennstrecke als erstes zu durchmessen.

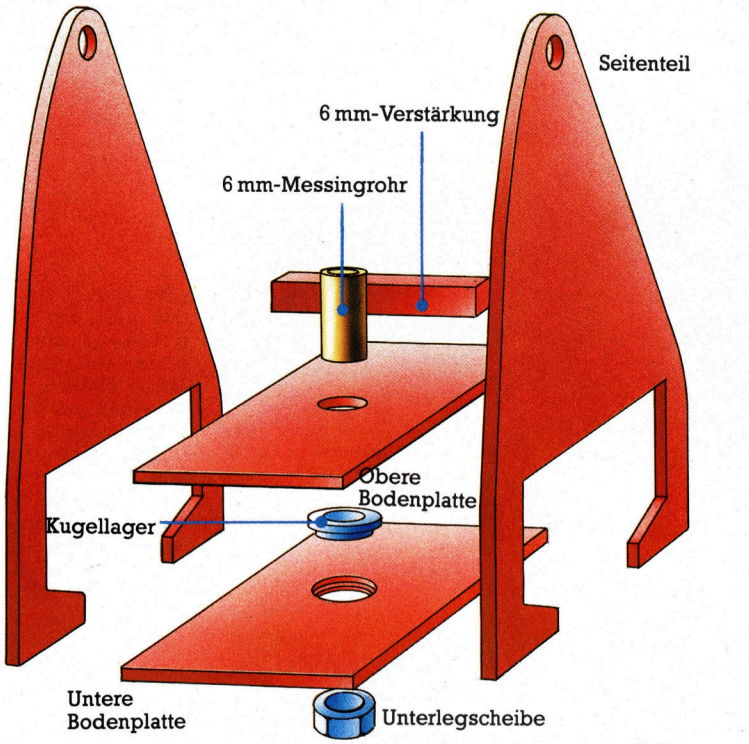
Zwei speziell für den Unterricht über Winkelverschiebung und Entfernung entwickelte Programme sind „Turn-Turtle" und „Shove-Turtle". Bei „Turn-Turtle" wird die Turtle im Mittelpunkt eines in Abschnitte gegliederten Kreises gesetzt. Jeder Abschnitt ist mit einer anderen Zahl versehen (Punktwert). Das Kind muß die Turtle so programmieren, daß sie sich dreht und die anderen Kreisabschnitte erreicht. Das Kind, das die höchste Punktzahl mit den wenigsten Programmschritten erreicht, ist Sieger. Bei „Shove-Turtle" findet eine ähnliche Methode Anwendung.

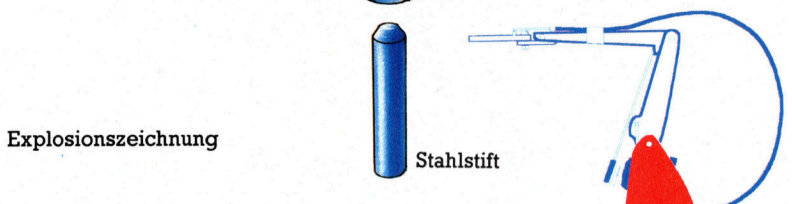
„Big Track" ist ein System, das auf LOGO-Art über Tastatur programmiert wird. Die eingegebenen Befehle bestehen aus einer Richtungsanweisung (links, rechts, etc.), gefolgt von einer Schrittanzahl (bis zu 99). „Big Track" kann bis zu acht verschiedene Befehle speichern und in richtiger Reihenfolge ausführen.

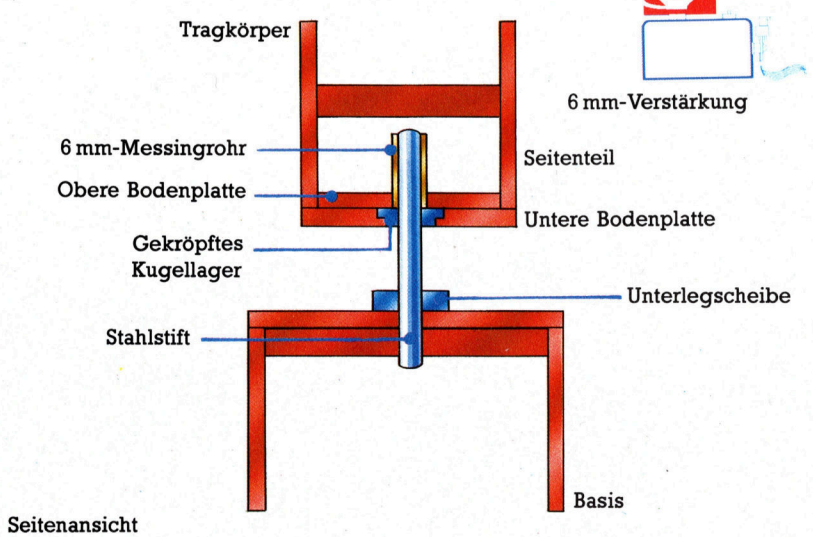
Montage des Robot-Arms

Bisher haben wir alle nötigen Bauelemente beschrieben – jetzt werden Tragkörper, Ober- und Unterarm zusammengesetzt und mit Gelenken versehen.

Montage des Tragkörpers







Seitenansicht

1. Montage des Tragkörpers

Für das Kugellager bohren Sie ein Loch in die untere (breitere) Bodenplatte des Tragkörpers. Mit einem dicken Holzbohrer wird das Bohrloch an der Oberseite etwas erweitert, damit der Lagerflansch Platz findet. Die obere Bodenplatte wird ebenfalls mit einer Bohrung versehen, in die ein 25 mm langes Messingrohr (Innendurchmesser 6 mm) passen muß. Kugellager einsetzen und die beiden Platten verkleben. Das Messingrohr wird von oben eingesteckt und ebenfalls festgeklebt.

Nach dem Trocknen Seitenteile ansetzen. Oben in die Seitenteile werden Löcher gebohrt, durch die ein Röhrchen mit 4 mm Außendurchmesser stramm (Preßpassung) hindurchpaßt. Als Verstärkung wird zwischen den Seitenteilen – neben dem Motorausschnitt – noch eine 37 mm lange Holzleiste (6 mm × 6 mm) eingeklebt. Wenn alles trocken ist, eine Unterlegscheibe (große Mutter geht auch) auf den aus der Basis vorstehenden Stahlstift stecken.

2. Oberes Armteil montieren

Von einem Messingrohr mit 5 mm Außendurchmesser wird ein 37 mm langes Stück abgeschnitten. Bohren Sie nun dafür passende Löcher in der Mitte der beiden unteren Armteile. An den schmaleren Enden werden Löcher für ein Messingrohr von 4 mm Außendurchmesser hergestellt – auch hier sollten die Löcher eher knapp bemessen sein (Preßpassung).

Das 5 mm-Rohr durch die mittleren Löcher stecken. Dabei dient der Servomotor als Abstandhalter. Zwei 20 mm lange Holzleisten (6 mm × 6 mm) werden als Verstärkung zwischen die Seitenteile geklebt – eine sitzt direkt unter dem Messingrohr, die zweite ganz unten (siehe Zeichnung). Nach dem Abbinden des Klebstoffs wird der Motor mit vier Holzschrauben an die Verstärkungsleisten geschraubt (Gummiunterlegscheiben verwenden).

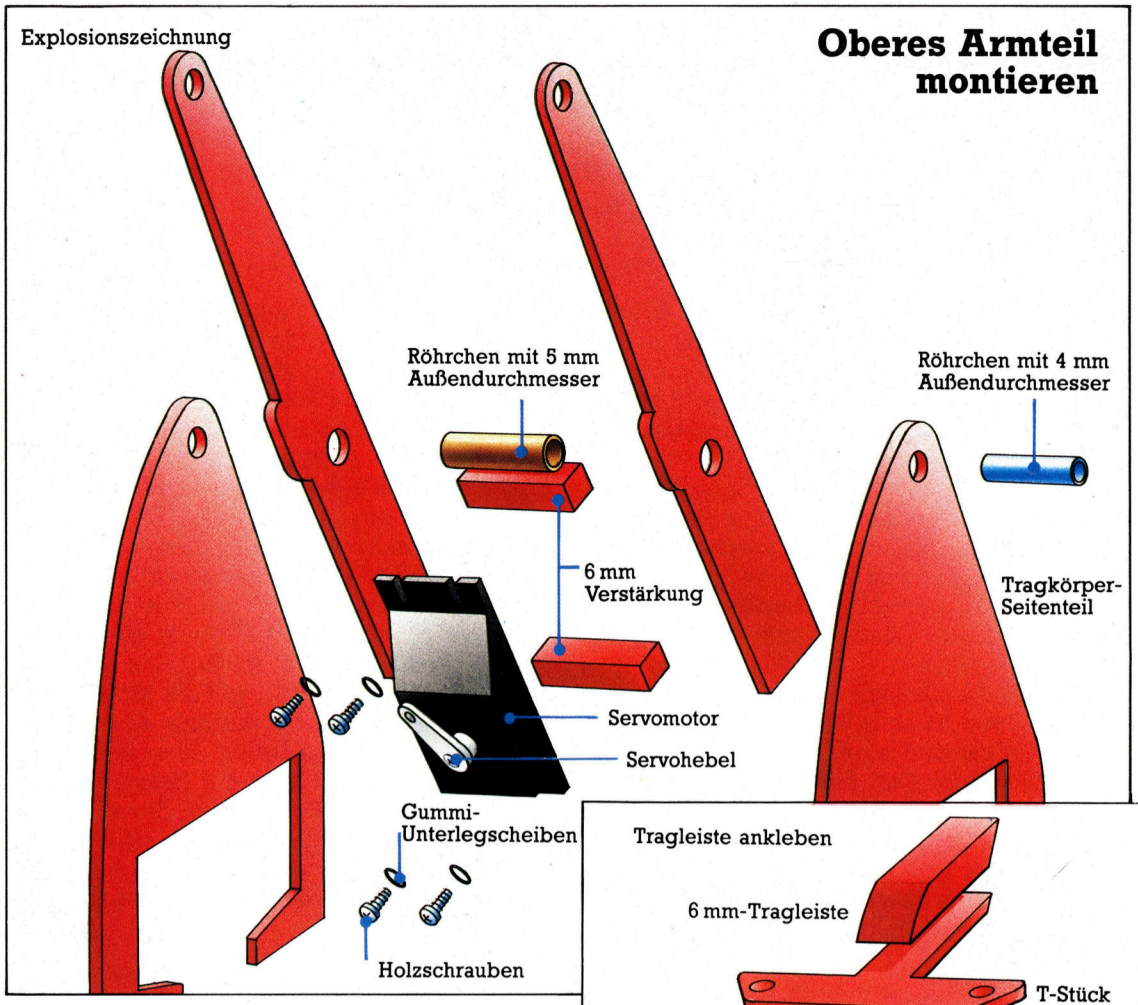
Als nächstes das Armteil durch ein „Schultergelenk" mit dem Tragkörper verbinden. Dazu stecken Sie ein 45 mm langes Stück Messingrohr (4 mm) durch eines der oberen Löcher des Tragkörpers, dann durch die Messingröhrchen des Oberarmes und durch das Loch auf der anderen Seite des Tragkörpers.

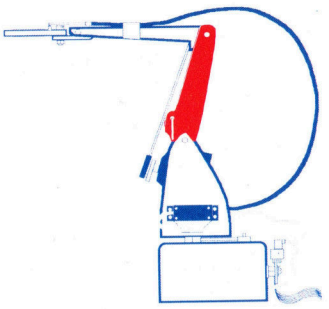
3. Montage des Unterarms

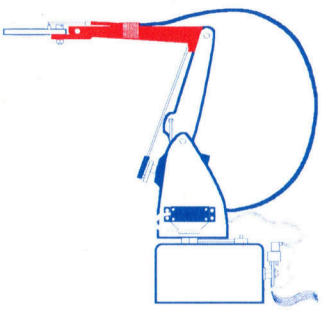
Die Seitenteile des Unterarmes bekommen am breiteren Ende je ein Loch für das 5 mm-Messingrohr. Am anderen Ende werden Löcher für die Gewindeschrauben vorgesehen. Auf das T-Stück eine Tragleiste (6 × 6 × 25 mm) kleben, die an den Enden abgeschrägt wird. Sie dient der Aufhängung des T-Stückes. Das T-Stück zwischen die Seiten des Oberarms halten und durch die aufgesetzte Tragleiste bohren. Eine Schraube hält T-Stück und Seitenteile zusammen.

Durch die vorgebohrten Löcher am anderen Ende des Arms wird ein 17 mm langes Stück Messingrohr (5 mm) gesteckt. Das Ellbogengelenk wird wie vorher durch Einführen eines 25 mm langen Messingröhrchens (4 mm) hergestellt. Vor dem Festkleben zentrieren.

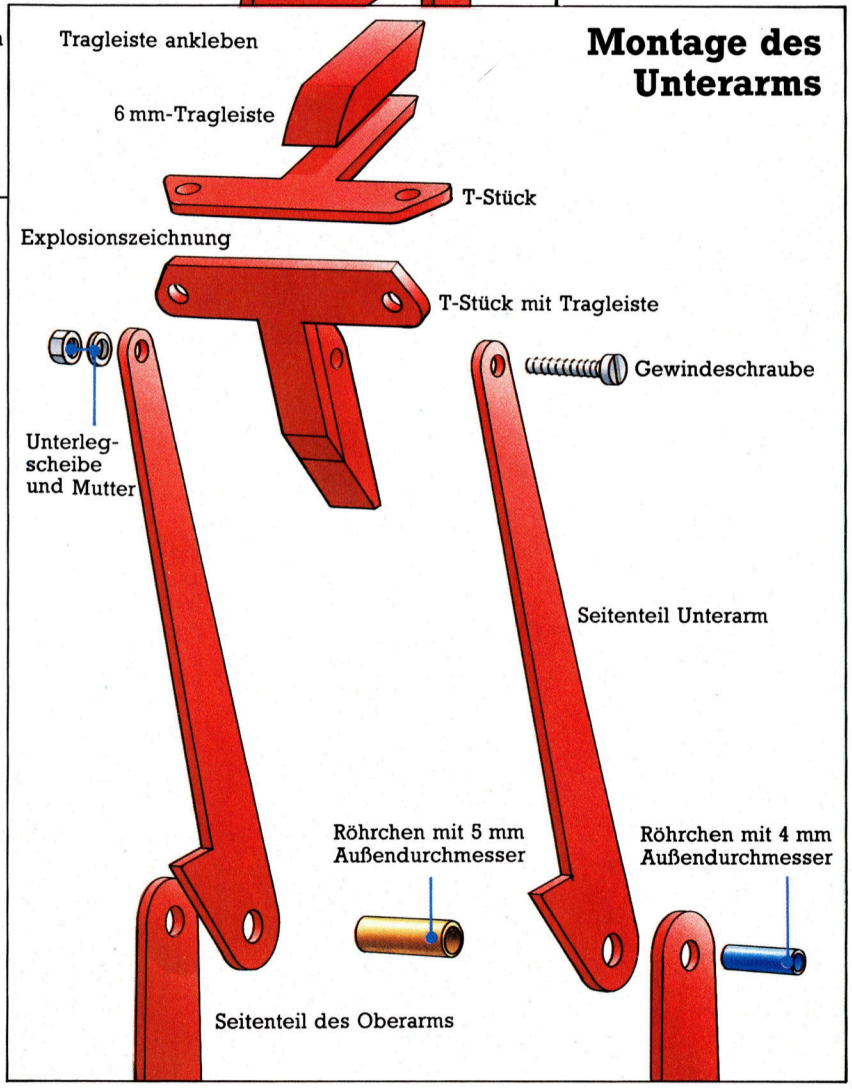
Oberes Armteil montieren

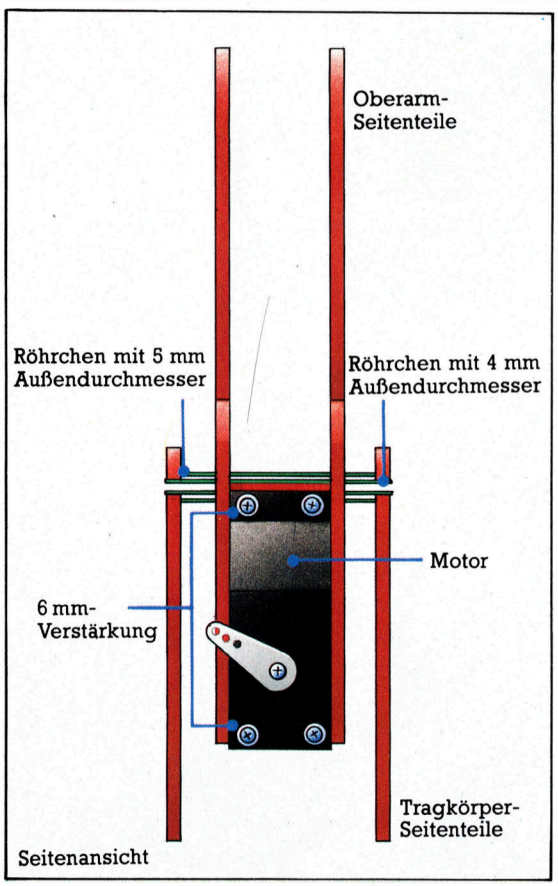






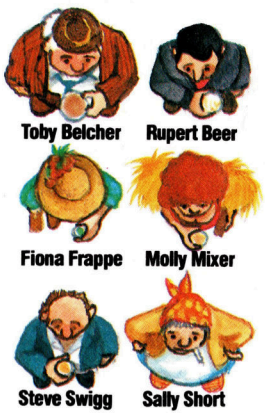
Montage des Unterarms





Wer, wo und wie?

Nach dem Konzept eines Adventure-Spiels behandeln wir nun die Bildschirmsteuerung und legen die Eigenschaften der unterschiedlichen Figuren fest.



Unser Bild zeigt die Spielfiguren in der Startposition, die sie durch die in Zeile 6030 und 6040 vorgegebenen Werte einnehmen. Auch die Gegenstände, deren Daten in Zeile 6140 und 6150 gespeichert sind, erscheinen auf dem Bild. Das Spiel beginnt im Foyer.

Das erste Programm-Modul beginnt in Zeile 10 mit der Bildschirmdarstellung und DIMensioniert die im letzten Abschnitt behandelten Arrays. Danach werden die nötigen Daten aus den entsprechenden DATA-Zeilen abgerufen (Zeile 6000 aufwärts).

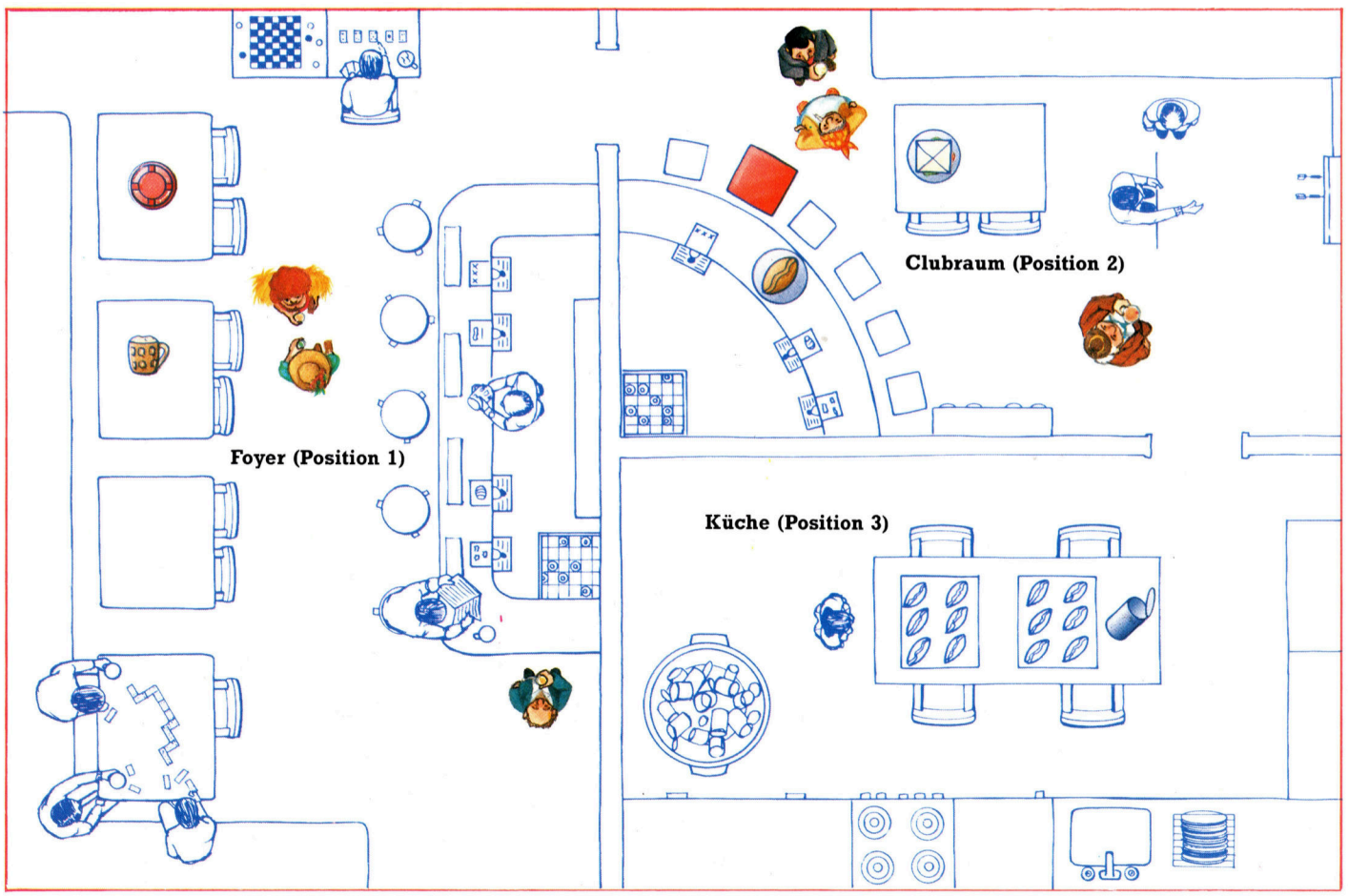
Diese beiden Module werden natürlich später ausgebaut, wenn zusätzliche Programmteile und das Figurenmodul selbst integriert werden. Das modulare Verfahren hat den Vorteil, daß bereits einzelne Abschnitte des Programms mit RUN gestartet und separat getestet werden können.

Die Abfrage „Default Values (y/n)" (vorgegebene Werte) in Zeile 70 erlaubt uns, anstelle der vorgegebenen eigene Werte für die Eigenschaften einzelner Personen in die DATA-Zeilen einzubauen. Durch diese Möglichkeit läßt sich das Verhalten von Personen mit mehr als zwei oder drei Eigenschaften auch während des Spieles verändern. Falls etwa Sally Short zu kurzangebunden ist oder zu schnell von einer Position an die nächste gelangt, kann das leicht modifiziert werden.

Zeile 530 ist eine Testschleife zum Aufruf aller bisher eingegebenen Hauptroutinen. Bei der weiteren Vervollständigung des Programms wird diese Zeile noch verändert.

Die Eingaberoutine in Zeile 2030 wartet auf einen Tastendruck. Alle Tasten außer 0,1,2 oder 3 (ASCII 48–51) werden ignoriert und führen das Programm wieder zu Zeile 530. Durch Betätigen einer „erlaubten" Taste gelangt der Spieler an die gewählte Position und kann das Geschehen verfolgen. Es gibt die Möglichkeit, eine Eingaberoutine für direktes Ansprechen jeder Spielfigur einzubauen. In unserem Beispielprogramm wird aber die Interaktion zwischen Spieler und Spielfiguren durch das Figurenmodul ausgeführt, weil wir den Spieler selbst als Spielfigur vorgesehen haben.

Falls Sie unsere Methode für eines Ihrer ei-



genen Adventure-Spiele nutzen wollen, ist es wichtig, daß das Figurenmodul auch unabhängig von der Eingabe-Routine funktioniert. Wie ein Modul integriert werden kann, haben wir bereits im BASIC-Kurs mit dem Spiel Haunted Forest erklärt.

Durch die Taste 0 kann jederzeit zur Programmzeile 2300 mit der „Default Values"-Routine gesprungen werden, um die Eigenschaften der Spielfiguren zu verändern.

Die Zeilen 2070–2390 enthalten die drei wichtigsten Routinen für die Bildschirmdarstellung. Zeile 2100 gibt die Beschreibung der aktuellen Position. Wie im Initialisierungsmodul vorgesehen, ist das beim ersten Spielstart die Beschreibung des Foyers (Position 1).

Die Routine „Print visible objekts" (Gegenstände zeigen) in Zeile 2150 durchläuft das Array b$ und gleicht die Einträge der Position mit dem aktuellen Standort des Spielers (r) ab. Die entsprechenden Beschreibungen werden auf dem Bildschirm angezeigt. Falls nicht, erhält man die Meldung „You can see: nothing" (Nichts zu sehen). Ähnlich arbeitet auch der Programmteil „Print visible characters" (Spielfiguren zeigen), bei dem das Array c$ durchsucht wird, um die anwesenden Spielfiguren anzugeben.

Das Unterprogramm „Initialise characters" (Spielfiguren erzeugen) kann während des Spiels durch Druck auf die 0-Taste und bei der Initialisierung durch Eingabe von Y auf die Meldung „Default values" aufgerufen werden.

„Sherlock" von Melbourne House ist ein gutes Beispiel für ein Programm, bei dem der Spieler ins Geschehen eingreifen kann, ohne zusätzliche Daten von Cassette oder Diskette nachladen zu müssen. Zur Aufklärung der Verbrechen gelangt der Spieler im Dialog, etwa durch Fragen wie „Erzählen Sie doch einmal von ..." etc. Die ersten Programmversionen wiesen noch einige lustige Mängel im Figurenmodul auf.

Startmodul
Bei zusätzlichen Variablen und Arrays muß das Modul weiter ausgebaut werden.

```
10 REM ¥welcome to the Dog and Bucket¥
20 REM
30 REM .....initialise.....
40 REM
45 GOSUB 4030: REM clear the screen
50 DIM l$(3,5),b$(12,4),c$(7,11),d$(11)
60 r=1
70  PRINT "Default values (y/n)?": GOSUB
 4110: IF i$="y" OR i$="Y" GOTO 90
80  GOSUB 2300: GOTO 100
90  FOR n=1 TO 7: READ c$(n,1): FOR d=2
TO 11: READ c$(n,d): NEXT d: NEXT n
100 FOR n=1 TO 3: READ l$(n,1): FOR e=2
TO 5: READ l$(n,e): NEXT e: NEXT n
110 FOR n=1 TO 12: READ b$(n,1): FOR d=2
 TO 4: READ b$(n,d): NEXT d: NEXT n
120 FOR n=2 TO 11: READ d$(n): NEXT n
```

Programmschleife
Die Zeile wird beim weiteren Wachsen des Programms verändert.

```
500 REM
510 REM test program loop
520 REM
530 GOSUB 2100: GOSUB 2150: GOSUB 2240:
GOSUB 2030: PRINT: PRINT: GOTO 530
```

Modul mit untergeordneten Subroutinen
Die abgedruckte Version ist für den Schneider CPC 464/664. Die BASIC-Dialekt stehen auf der nächsten Seite.

```
2000 REM
2010 REM input routine
2020 REM
2030 GOSUB 4110
2040 IF (ASC(i$)<48) OR (ASC(i$)>51) THE
N RETURN
2050 IF i$="0" THEN GOSUB 2320: RETURN
2060 r=VAL(i$): c$(7,2)=i$: RETURN
2070 REM
2080 REM location print
2090 REM
2100 PRINT l$(r,1)
2110 RETURN
2120 REM
2130 REM print visible objects
2140 REM
2150 PRINT "You can see: ";
2160 p=0: FOR b=1 TO 12: IF VAL(b$(b,2))
<>r GOTO 2190
2170 p=p+1: IF p>1 THEN PRINT ", ";
2180 PRINT b$(b,1);
2190 NEXT b
2200 IF p=0 THEN PRINT "nothing";
2210 PRINT: RETURN
2220 REM
2230 REM print visible characters
2240 REM
2250 p=0: FOR c=1 TO 6: IF VAL(c$(c,2))<
>r GOTO 2290
2260 p=p+1: IF p=1 THEN PRINT "You are i
n the company of: ";: GOTO 2280
2270 PRINT ", ";
2280 PRINT c$(c,1);
2290 NEXT c
2300 IF p=0 THEN PRINT "There's no-one h
ere."
2310 RETURN
2320 REM
2330 REM initialise chars subroutine
2340 REM
2350 PRINT: RESTORE: FOR c=1 TO 6: READ
c$(c,1): GOSUB 4070: PRINT c$(c,1);" - "
;: PRINT "Set this char?": GOSUB 4110
2360 IF (i$<>"y") AND (i$<>"Y") THEN FOR
 n=2 TO 11: READ n$: NEXT n: GOTO 2390
2370 FOR d=2 TO 11: READ s$: PRINT d$(d)
;: INPUT i$: IF i$<>"" THEN c$(c,d)=i$
2380 NEXT d
2390 NEXT c: RETURN
```

Dazu wird das Array d$(11) und der Name der Person zu einer Frage verbunden, ob die betreffende Spielfigur festgelegt werden soll. Wird das bejaht, werden die in Array d$ gespeicherten Bezeichnungen der Eigenschaften dargestellt, und für jede wird ein Wert eingegeben. Bei Eingabe eines Null-Strings für eine Eigenschaft (nur ENTER drücken) bleibt der bisherige Wert unverändert. Falls dieser Programmteil während der Initialisierung aufgerufen wird, muß für jede Eigenschaft und alle Personen ein Wert eingegeben werden – sonst arbeitet das Figurenmodul nicht richtig.

Das Modul „Low Level Subroutines" (Einfache Unterprogramme) ab Zeile 4000 dient der Anpassung an unterschiedliche Rechner.

Daten werden erst ganz zum Schluß gespeichert. Dieses Modul wird beim Programmieren zusätzlicher Routinen sehr viel größer, speziell durch Speicherung der Meldungen.

Nach dem Eintippen und Starten des Programms können Sie durch Druck auf 1,2 oder 3 die einzelnen Standorte durchwandern. Auch der Spielfiguren-Editor läßt sich bereits (mit 0) aufrufen und darauf testen, ob die Figuren mit neuen Werten noch richtig dargestellt werden.

Modul „Wichtige Unterprogramme"
Zu Anfang haben wir nur vier wichtige Unterprogramme – Eingabe, Orte, Figuren und Gegenstände sowie den Editor.

```
4000 REM
4010 REM low level system subroutines
4020 REM
4030 REM clear the screen
4040 REM
4050 CLS: RETURN
4060 REM
4070 REM beep
4080 REM
4090 PRINT CHR$(7);: RETURN
4100 REM
4110 REM get a character from the keyboa
rd
4120 REM
4130 i$=INKEY$: IF i$="" GOTO 4130
4140 RETURN
```

Datenmodul
Beachten Sie, daß die Ortsbeschreibungen im 40-Zeichen-Modus definiert sind. Die Werte in Zeile 6030–6040 dürfen abgewandelt werden. So können Sie den Spielfiguren etwa Namen aus Ihrem Bekanntenkreis geben, oder aus Fernsehserien – und Sie können ihnen einen neuen Charakter verleihen.

```
6000 REM
6010 REM character data
6020 REM
6030 DATA "Toby Belcher","2","7","10","1
0","7","m","0","0","7","4","Fiona Frappe
","1","8","30","10","8","f","0","0","3",
"5","Steve Swig","1","9","8","10","9","m
","0","0","4","6","Sally Short","2","0",
"20","10","10","f","0","0","5","5"
6040 DATA "Rupert Beer","2","11","10","6
","11","m","0","0","4","6","Molly Mixer"
,"1","12","15","6","12","f","0","0","5",
"5","you","1","0","255","255","0","m","0
","0","0","0"
6050 REM
6060 REM location data
6070 REM
6080 DATA "You are in the lounge of the
Dog and     Bucket. Several shady charact
ers are     gathered together in a corner
 playing    dominoes.  Behind the counter
, Fred the Barman looks his usual cheery
 self.      Exits lead east.","0","0","2"
,"0"
6090 DATA "Behold the saloon of the Dog
and Bucket,which looks as if it could do
 with       extensive redecoration. The f
loor        appears to have been regularl
y hosed     down with beer slops.  Doors
lead west   and south.","0","3","0","1"
6100 DATA "Ugh! This is the kitchen, whe
re the      famous Dog and Bucket Cornish
 Pasties    are prepared for an ever-hung
ry          clientele. You notice a numbe
r of empty cat-food tins, which is stran
ge because there are no cats.","2","0","
0","0"
6110 REM
6120 REM object data (for b$(12,4))
6130 REM
6140 DATA " a glass of beer","2","n","y"
," an empty tin of catfood","3","n","n",
" a Dog and Bucket cornish pasty","1","y
","n"," a bar-stool","2","n","n"," an as
htray","1","n","n"
6150 DATA " a stale ham sandwich","2","y
","n"," a pint of bitter","0","n","y","
a creme de menthe","0","n","y"," a whisk
y and water","0","n","y"," a neat vodka"
,"2","n","y"," a pint of lager","0","n",
"y"," a gin and ginger ale","0","n","y"
6160 REM
6170 REM character attribute data (for d
$(11))
6180 REM
6190 DATA "Location","Strength","Invento
ry","Mood","Own object","Sex","Last char
 (lch)","Last command code (lcd)","Handl
e frequency","Move frequency"
```

BASIC-Dialekte

Spectrum:

```
50   DIM L$(3,5,255),b$(12,3,25),c$(6,11,15),
     d$(11,25)
2040 IF (CODE(i$)<48) OR (CODE(i$)>51)
     THEN RETURN
4090 BEEP .5,3: RETURN
```

Commodore:

```
4050 PRINT "<shift/CLR>": RETURN
4090 POKE 54296,15: POKE 54276,17: POKE
     54277,64
4095 FOR A=1TO 50: POKE 54273, 36: POKE
     54272,85: NEXT: POKE 54273, 0: POKE
     54272, 0: POKE 54276,0: RETURN
4130 GET i$: IF i$="" THEN 4130
```

Acorn B:

```
4130 i$=GET$
```

Neu definiert

Aus Gründen der Geschwindigkeit und Optimierung werden die Colon-Definitionen in FORTH „halb-compiliert". In diesem Artikel beschreiben wir die Compiliermethoden von FORTH und untersuchen, welche Wirkung die Neudefinition von bereits bestehenden Wörtern auf die Programmausführung hat.

Die halb-compilierten Colon-Definitionen werden zwar nicht in den Maschinencode übersetzt (wie bei einer vollständigen Compilierung), belegen aber wenig Platz und laufen schnell ab. FORTH Programme haben daher eine hohe Ausführgeschwindigkeit.

Es gibt zwar mehrere Methoden der Halb-Compilierung, doch wird die figFORTH-Methode an häufigsten eingesetzt. Bei der Definition des Wortes:

```
:SQUARE (n-n*n) DUP * ;
```

wird zunächst ein „Header" in das Vokabular eingetragen. Er enthält den Namen SQUARE und die Speicheradresse des Headers des zuletzt definierten Wortes. Da alle Header miteinander verbunden sind, kann der Computer das Vokabular vom zuletzt definierten Wort an rückwärts durchsuchen, bis er das gewünschte Wort findet. Die beiden Teile des Headers heißen „Namensfeld" und „Linkfeld".

Auf den Header folgt die Definition. Dort steht zunächst ein Code, der anzeigt, welche Art der Definition vorliegt: Es gibt unterschiedliche Codes für Colon-Definitionen, Variablen, Konstanten, mit CREATE angelegte Strukturen, Wörter, die in Maschinencode für das System definiert wurden und jede andere Art von Wörtern. Dieser Teil heißt „Codefeld".

Als nächstes kommt der Hauptteil der Colon-Definition. Er besteht aus den Adressen der Wörter, die von der Definition eingesetzt werden („Compilierungsadressen").

Hier die Definition von SQUARE:

Namensfeld: "SQUARE"

Linkfeld: Headeradresse des vorigen Wortes (2 Bytes)

Codefeld: Code für eine Colon-Definition (2 Bytes)

Parameterfeld: Compilierungsadresse von DUP (2 Bytes)
Compilierungsadresse von * (2 Bytes)
Compilierungsadresse von ; (2 Bytes)

Die Compilierung von Colon-Definitionen läuft folgendermaßen ab: Zunächst werden das Namensfeld und das Linkfeld angelegt. Bis hier ist der Vorgang für jede Definition gleich. Als nächstes setzt der Doppelpunkt (:) den speziellen Code für Colon-Definition ein. Von nun an wird jedes eingegebene Wort nicht auf normale Weise ausgeführt, sondern dessen Compilierungsadresse in das Vokabular eingetragen. Damit Sie erkennen können, daß dieser Arbeitsgang aktiviert ist, wird die Meldung OK unterdrückt.

Der Vorgang setzt sich so lange fort, bis ein (;) eingegeben wird. Das Semikolon hat außer seiner im Vokabular eingetragenen Funktion noch die Wirkung, FORTH sofort aus der Compilierung in den normalen Zustand zurückzubringen. Wörter, die als Teil des Compilierungsvorgangs ausgeführt werden, werden „unmittelbare" (immediate) Wörter genannt. Sie befinden sich normalerweise nicht im Vokabular, können aber dort speziell definiert werden. In diesem Fall wird im Vokabular jedoch nicht deren Compilierungsadresse eingetragen, sondern eine Art anonymer „Schatten", der die Abläufe vom „;" für die Programmausführung enthält.

Bei der Ausführung von SQUARE (beispielsweise bei der Eingabe von 4 SQUARE) sucht FORTH im Vokabular und findet dort die entsprechende Definition. Diese teilt FORTH mit, daß zuerst DUP, dann * und dann das Ende (der Definition) ausgeführt werden soll.

Da Zahlen keine Compilierungsadressen besitzen, werden sie in einem zusammengesetzten Format compiliert. Am Anfang steht dabei die Compilierungsadresse einer direkten Behandlungsroutine (Literal-Handler) – und dann die eigentliche Zahl. Ein Literal-Handler, der im Inneren einer Colon-Definition auftritt,

Codefelder

In Codefeldern ist die Codenummer die Speicheradresse des Maschinencodes, der bei Aufruf des Wortes ausgeführt wird. Jede Colon-Definition speichert die Adresse, zu der der FORTH-Interpreter nach Ausführung des Wortes zurückkehrt. Dafür wird die Rücksprungadresse auf einen speziellen Stapel – den „Rücksprungstapel" – geschoben und der Interpreter auf die Anfangsadresse der aktuellen Definition gesetzt.

Die Maschinencoderoutinen für Konstanten und Variablen, deren Parameterfelder nur eine einzige Zahl enthalten, sind anders aufgebaut: Bei sogenannten primitiven Wörtern wie + und * liegt der entsprechende Maschinencode direkt im Benutzerfeld, während das Codefeld die Adresse enthält.

Auch die Schneider-Computer CPC 464 und 664 gehören zu der wachsenden Zahl von Computern, die mit FORTH arbeiten können. Die Kuma-Version von FORTH entspricht dem figFORTH-Standard und bietet Fließkommaarithmetik, Möglichkeiten der Stringverarbeitung und Farbgrafik.

teilt dem System mit, daß die beiden folgenden Bytes keine Compilierungsadresse darstellen, sondern eine Zahl, die auf den Stapel geschoben werden soll.

Das nächste Problem tritt bei der Programmierung von Strukturen wie IF...THEN.... ELSE auf. Im Maschinencode oder in einfacheren BASIC-Versionen läßt sich diese Aufgabe mit Sprüngen lösen, wie:

```
1000 IF oberster Stapelwert = 0 THEN
       GOTO 1000
1010 REM folgende Zeilen ausführen, wenn
       die Bedingung wahr ist
       :
       :
1090 GOTO 1200
1100 REM folgende Zeilen ausführen, wenn
       die Bedingung falsch ist
       :
       :
1200 REM in beiden Fällen hier weiter-
       machen
```

Compiliertes FORTH funktioniert ähnlich. Doch obwohl FORTH die Wörter BRANCH (unbedingter Sprung) und ?Branch (obersten Stapelwert herunterziehen und springen, falls der Wert Null ist) besitzt, können sie nicht ohne weiteres verwandt werden, da sie sich nur in zusammengesetzter Form (wie eine compilierte Zahl) einsetzen lassen: Auf die Compilierungsadresse muß die Adresse folgen, auf die der Sprung stattfinden soll. Es ist Aufgabe der unmittelbaren (immediate) Wörter IF, ELSE, THEN etc., diese Sprungadressen herauszufinden und in die zusammengesetzten Formate einzufügen. So enthält IF beispielsweise die Compilierungsadresse von ?BRANCH und den Platz für die Sprungadresse, kann aber die Sprungadresse nicht einsetzen, da nicht bekannt ist, wo sich das ELSE befindet. IF legt statt dessen die Adresse auf den Stapel, und ELSE setzt später die Sprungadresse ein. Damit lassen sich Strukturen wie

```
IF...BEGIN...ELSE...UNTIL...THEN
```

abfangen. Sehen wir uns einige Beispiele an:

1) Nehmen Sie einmal an, Ihre FORTH-Version könnte zwar zwischen Groß- und Kleinschreibung unterscheiden, doch wären alle Standardwörter nur mit Großbuchstaben im Vokabular eingetragen, – das heißt, LOOP würde erkannt werden, loop dagegen nicht. Wenn Sie die Standardbefehle nun auch in Kleinbuchstaben eingeben wollten, könnten Sie definieren:

```
: drop DROP;
: roll ROLL;
: variable VARIABLE (auch das funktioniert);
```

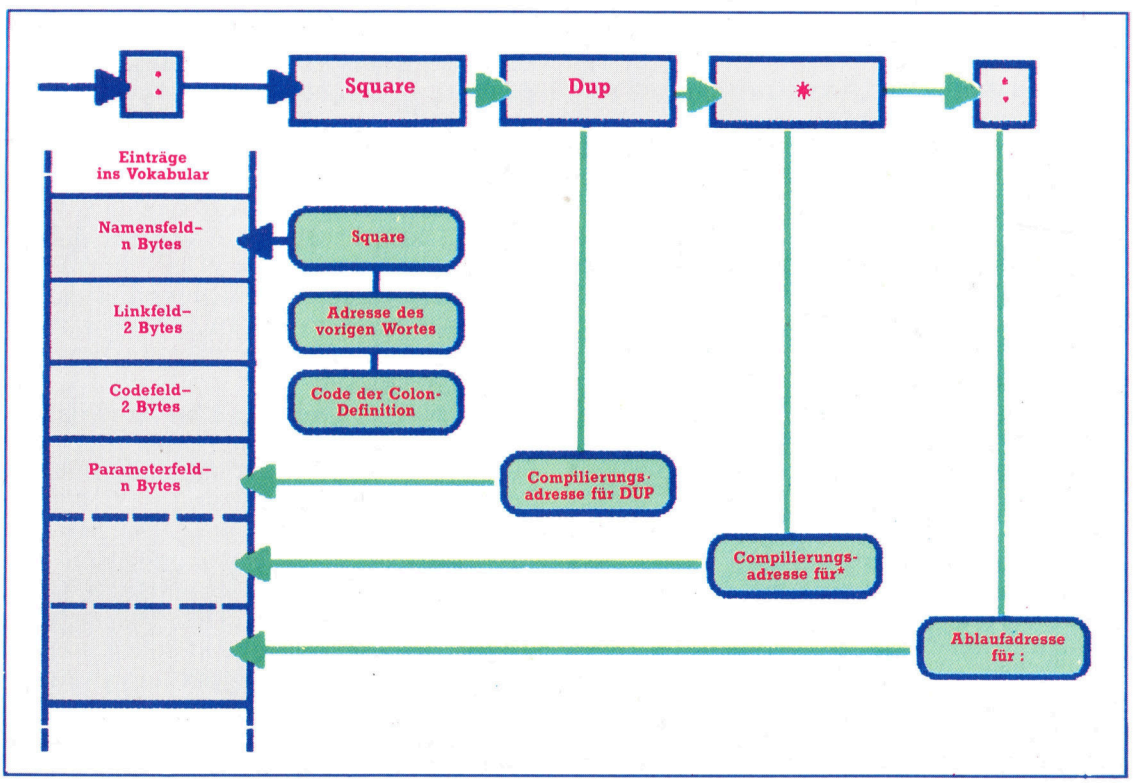
Schwierigkeiten gibt es mit

```
:loop LOOP (funktioniert nicht)
```

da loop ein unmittelbares Wort ist und daher schon bei der Definition ausgeführt wird. Die einzige Möglichkeit, LOOP mit Kleinbuchstaben zu definieren, ist:

```
:loop [COMPILE] LOOP ; IMMEDIATE
```

Hierbei müssen drei Situationen berücksichtigt werden. Zunächst die Definition: Während die Compilierungsadresse unter loop eingetragen wird, schaltet [COMPILE] die sofortige Ausführung ab, und IMMEDIATE kennzeichnet loop als unmittelbares Wort. Ferner ist zu berücksichtigen, daß loop während der Compilierung eines anderes Wortes eingesetzt und als unmittelbares Wort sofort ausgeführt wird. Da dies der Haupteinsatzbereich von loop ist, tritt dieser Fall häufig ein. Die dritte Situation



FORTH arbeitet mit der Methode der „Halb-Compilierung", um die Programmausführung zu beschleunigen. Bei Eingabe einer Colon-Definition werden im Vokabular unterschiedliche Felder angelegt, die mit dem Namensfeld, Linkfeld und dem Codefeld anfangen. FORTH schaltet dann den Compilierungsmodus (Rot) an, in dem das Wort nicht unmittelbar in das Vokabular eingetragen wird, sondern nur eine „Compilierungsadresse" im Parameterfeld der Definition. Nach dem ; schaltet FORTH wieder in den Ausführmodus (Schwarz) zurück.

Bedeutende Wörter

Mit [und] können Sie von einer Colon-Definition in den Ausführungsmodus schalten.
LITERAL, ein unmittelbares (immediate) Wort, nimmt den obersten Wert des Stapels und compiliert ihn in einer Colon-Definition auf übliche Weise in eine Zahl. Normalerweise wird LITERAL zusammen mit [und] eingesetzt, um während der Compilierung einen Ausdruck zu berechnen, dessen Ergebnis compiliert werden soll. So hat

```
[ 9 16 * ] LITERAL
```

die gleiche Auswirkung wie die Zahl 144.
[COMPILE] ist ein unmittelbares Wort, das veranlaßt, daß das nächste Wort nicht ausgeführt, sondern direkt in die Definition eingeschlossen wird.
COMPILE ist kein unmittelbares Wort. Bei seiner Ausführung wird das darauffolgende Wort in das Vokabular eingeschlossen.
BRANCH und ?BRANCH werden wie in unserem Beispiel eingesetzt. >MARK und >RESOLVE geben bei einem Vorwärtssprung die Sprungadressen an: >MARK steht dabei an der Position des Sprunges und >RESOLVE an der Bestimmungsposition. <MARK und <RESOLVE werden auf die gleiche Weise für einen Rückwärtssprung verwandt, wobei <MARK vor der Bestimmungsposition steht und <RESOLVE an der Sprungposition.
IMMEDIATE wandelt das vorangehende Wort in ein unmittelbares Wort um.

entsteht beim eigentlichen Ablauf, wenn bei der Ausführung eines anderes Wortes die von LOOP compilierten Definitionen ablaufen.
2) Das Wortpaar ?DO und ?LOOP ähnelt DO und LOOP. Wenn dabei Anfangswert gleich Endwert ist, kommt die Schleife niemals zur Ausführung.

```
?DO entspricht OVER OVER > IF DO
?LOOP entspricht LOOP ELSE DROP DROP
    THEN
```

Bei den Definitionen

```
: ?DO (hier ist der Fehler) OVER OVER > IF
    DO;
```

würde das gleiche Problem auftreten wie bei loop. Die korrekte Definition sieht folgendermaßen aus:

```
:?DO
  COMPILE OVER
  COMPILE OVER
  COMPILE >
  [COMPILE] IF
  [COMPILE] DO
;IMMEDIATE

:?LOOP
  [COMPILE] LOOP
  [COMPILE] ELSE
  COMPILE DROP
  COMPILE DROP
  [COMPILE] THEN
;IMMEDIATE
```

3) Die Programmstruktur CASE bietet – abhängig von einem Steuerwert – eine ganze Reihe von Ablaufmöglichkeiten. CASE ist zwar nicht standardmäßig vorgesehen, läßt sich aber leicht definieren. Hier das Format:

```
  Wert CASE
1.Möglichkeit OF . . .ENDOF
2.Möglichkeit OF . . .ENDOF
3.Möglichkeit OF . . .ENDOF
falls keine der Möglichkeiten eintrifft
ENDCASE
```

Die Variable „Wert" kann beispielsweise der ASCII-Code einer Taste sein. Damit lassen sich per Tastendruck unterschiedliche Abläufe aufrufen.

In der Definition von CASE benötigt jedes OF einen bedingten Sprung auf ein ENDOF und jedes ENDOF einen Sprung auf ENDCASE, während ENDCASE unter Umständen mehrere Sprungadressen einfügen muß. Da die meisten FORTH-Versionen für diese Aufgabe keine Wörter besitzen, müssen Sie diese Sprünge mit >MARK und >RESOLVE selbst ausarbeiten. >MARK schiebt dabei (wie IF) die Adresse des für die Sprungadresse reservierten Platzes auf den Stapel, während >RESOLVE sie wieder herunterzieht und die Adresse einfügt.

```
  : CASE (Laufzeit: Wert — Wert)
        ( compilieren: —0)
        0 (Zahl der OFs — bisher keine)
; IMMEDIATE
: OF ( Laufzeit: Wert, falls möglich — [bei
      Übereinstimmung])
( Wert, falls möglich — Wert [keine Überein-
      stimmung])
(compilieren: — Markierung für den Sprung
      auf ENDOF)
  COMPILE OVER
COMPILE =
COMPILE ?BRANCH >MARK
COMPILE DROP (Wert löschen falls
    Übereinstimmung)
; IMMEDIATE
: ENDOF (Laufzeit: —)
  ( compilieren: OF Zähler,
          OF Markierungen —
( Markierung für den Sprung auf ENDCASE,
  OF Zähler + 1)
COMPILE BRANCH >MARK
SWAP >RESOLVE (Sprungadresse von OF
    einsetzen)
SWAP 1+
; IMMEDIATE
: ENDCASE (Laufzeit: Wert —
    (compilieren: Markierung der ENDOFs,
    OF
    Zähler —)
  COMPILE DROP
  0 ?DO
    > RESOLVE
    ?LOOP
    ; IMMEDIATE
```

Vor dem Einsatz sollten Sie die Abläufe mit einigen Testzahlen überprüfen.

In Sachen Oric

Das englische Unternehmen Oric Products International wurde als Konkurrenz zu Sinclair finanziell von British Car Auctions und personell von Tangerine Computer Systems und Tansoft unterstützt.

Bei der Gründung von Oric im Jahr 1982 brachten Barry Muncaster als leitender Manager und Peter Harding als Vertriebschef die Marketingkenntnisse mit, während durch Dr. Paul Johnson und Paul Kaufman die nötige Erfahrung in der Hard- bzw. Softwareentwicklung dazukam. Das erste Produkt der Firma war der Oric-1, ein Gegenstück zum Sinclair Spectrum mit der 6502-CPU und wahlweise 16 oder 48 KByte Speicherkapazität. Im Vergleich zum Spectrum war der Oric-1 mit einer besseren Tastatur, soliderem Gehäuse und mehr Grafik- und Sound-Möglichkeiten ausgestattet, während als ROM-BASIC der übliche Microsoft-Dialekt Verwendung fand. Der Oric verfügt über eine Centronics-Schnittstelle für den Betrieb eines Druckers.

Zu Anfang hatte die Firma mit etlichen Widrigkeiten wie Entwicklungsfehlern und nachfolgenden Lieferproblemen zu kämpfen: Bei den ersten Rechnern gab es Schwierigkeiten mit dem Laden von Cassette und mit der Bildwiedergabe. Außerdem enthielten die BASIC-ROMs einige Fehler. Erschwerend wirkte sich dazu noch die eigenwillige Oric-Grafik aus, die auf minimalen Speicherbedarf ausgelegt ist. Das alles war nicht gerade verkaufsfördernd, zumal das System dann auch noch mehr kostete als der Sinclair Spectrum.

Der Oric-1 überlebte nur 15 Monate; dann entschloß sich die Firma zu einer anderen Verpackung und brachte den Oric Atmos im rot-schwarzen Gewand mit Schreibmaschinentastatur und überarbeitetem BASIC-ROM heraus. Auch der Vierfarben-Printerplotter wurde dem neuen Design angepaßt.

Als Folge des zögernden Absatzes kam auch die Software-Produktion zunächst nicht richtig in Gang. Später änderte sich dieser beklagenswerte Zustand, und Tansoft sorgte für eine ausreichende Softwarebasis. Von der zugesagten Entwicklung einer kompletten Peripherie mit Floppylaufwerk, Modem und Drukker wurde jedoch anfangs nur ein Vierfarben-Printerplotter Realität.

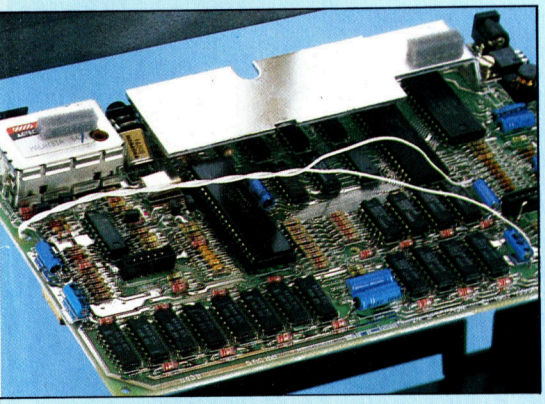
Nach diesem verhaltenen Start faßte Oric langsam Fuß auf dem Markt. Eindrucksvoll entwickelte sich vor allem der Exportanteil: Von den über 170 000 im Jahr 1983 hergestellten Rechnern ging mehr als die Hälfte ins Ausland. Besonderer Beliebtheit erfreut sich der Oric in Frankreich, wo er 1983 zum „Top Micro" gewählt wurde. Sogar in Japan konnten spezielle Oric-Versionen verkauft werden.

Fünfzehn Monate nach der Vorstellung des ersten Rechners brachte das Unternehmen unter der Bezeichnung Oric Atmos ein neues Modell mit einer vernünftigen Tastatur (QWERTZ-Format) und verbessertem BASIC heraus; im übrigen handelte es sich aber um die alte Maschine in neuem Gehäuse. Zudem wurde die Palette der Peripheriegeräte erheblich erweitert. Inzwischen bietet Oric unter anderem eine 3-Zoll-Diskettenstation, ein Modem für Bildschirmtext und direkte Rechnerkommunikation sowie einen 80-Zeichen-Matrixdrucker im charakteristischen Atmos-Design an.

Fachwörter von A bis Z

Keypunch = Locher
Lochkarten oder -streifen gehören zu den klassischen Datenträgern. Bei Karten werden Buchstaben und Ziffern als Lochmuster auf einem Hartpapierblatt mit zwölf Zeilen à 80 Spalten dargestellt. Dieses Format war als „Hollerith-Code" schon vor der Jahrhundertwende bei Tabelliermaschinen – den ersten Bürorechnern – gebräuchlich.

Keyword = Schlüsselwort
Um das Programmieren benutzerfreundlicher zu machen, arbeiten alle höheren Programmiersprachen mit sogenannten „Schlüsselwörtern" – das sind Vokabeln, die mit genau definierter Bedeutung wesentliche Strukturelemente der Sprache darstellen. Wenn der Compiler oder Interpreter auf ein solches „Keyword" trifft, verfährt er nach einer bestimmten vorgegebenen Routine. Auch der BASIC-Befehlsvorrat ist aus derartigen Schlüsselwörtern aufgebaut: PRINT, LET, GOTO, IF...THEN usw. haben festgelegte Funktionen und dürfen deshalb nicht als Variablennamen verwendet werden.



Die beiden Leitungen, die hier über der Leiterplatte des Sinclair Spectrum liegen, führen zu einer separaten Reset-Taste und verwandeln den Spectrum in einen Spectrum+. Die (kostspielige) Alternative für diesen typischen „Kluge" hätte darin bestanden, eine neue Platine zu entwerfen, um die zusätzlichen Leiterbahnen unterzubringen.

Kludge = Zwischenlösung
Sind bei einem Gerät während des Fertigungsanlaufs plötzlich wegen irgendwelcher Probleme noch Änderungen erforderlich, hilft nur Improvisation. Die „Zwischenlösung", die dann schnellstens aus vorhandenen Teilen zusammengebaut wird, bezeichnen die Engländer als „Kludge" Bei den ersten QL-Modellen beispielsweise konnte das Betriebssystem nicht in den vorgesehenen ROMs untergebracht werden. Als Zwischenlösung wurde ein zusätzliches EPROM eingebaut.

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.

Label = Markierung
„Label" sind Kennzeichnungen zur Identifizierung oder Abgrenzung von Informationen in der Datenverarbeitung. Bei der Assembler-Programmierung zum Beispiel werden Label zur Markierung von Anweisungen und als symbolische Adressen benutzt, etwa in folgender Form:

Label-Feld	Anweisungs-Feld	Operanden-Feld
LOOP	LDA	TEST
EXIT	BNE	LOOP

LOOP, TEST und EXIT sind hier Label, die bei der Übersetzung des Programms in den Maschinencode durch die entsprechenden Adressen ersetzt werden.

Eine andere Label-Art sind kommentierende Zusätze im Programm, zum Beispiel REMarks in BASIC, aus denen hervorgeht, welche Aufgabe der folgende Block hat. In dieser Form sind Label wesentliche Elemente der strukturierten Programmierung, die dem Programmierer die Übersicht erleichtern.

Die strukturierte Programmierung macht auch von „Prozeduren" Gebrauch. Das sind Unterprogramme, auf die mit Hilfe eines Label Bezug genommen werden kann.

Ein Label kann auch eine kurze Information auf Cassette oder Diskette zur Charakterisierung von Merkmalen der gespeicherten Dateien sein. Ein „Volume Label" beschreibt den gesamten Datenträger, während ein „Header Label" oder Dateikopfsatz nur für eine einzelne Datei zuständig ist.

Schließlich gibt es bei komfortablen Tabellenkalkulationsprogrammen noch die Möglichkeit, Gruppen von verwandten Tabellenfeldern durch Label zu markieren und dann Rechnungen mit sämtlichen Feldern auf einmal durchzuführen, indem die Label in eine Formel eingesetzt werden. Sind etwa die Verkaufsziffern für ein Produkt ein Jahr lang notiert worden, läßt sich daraus eine Verkaufsprognose für das nächste Jahr erstellen, indem man bei den Vorjahreszahlen prozentuale Auf- und Abschläge berücksichtigt. Ohne die Verwendung von Labels müßte man für jedes betroffene Tabellenfeld eine Gleichung der Art
A4 × 110% = B4
ansetzen, die Beziehung also viele Male hinschreiben. Wenn Label zugelassen sind, muß man stattdessen nur einmal die Anweisung geben
Alter Umsatz × 110% =
Neuer Umsatz
und die zugehörigen Feldbereiche entsprechend markieren. Die Formel spricht dann alle Felder mit dem Label „Alter Umsatz" an und überträgt die Ergebnisse in den Datensatz mit dem Label „Neuer Umsatz". Dieses Verfahren erleichtert den Umgang mit Tabellensystemen in erheblichem Maße.

Bildnachweise
1625: Johannes Heidenhain
1626, 1627: Kevin Jones
1630, 1637: Ian McKinnell
1638, 1639: Chris Stevens
1641: Tony Sleep
1642: Kevin Jones
1643: Ian McKinnell
1644, 1645: Kevin Jones
1646: David Higham, Kevin Jones
1650: Caroline Clayton
U3: Chris Stevens

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

computer kurs Heft 60

Textumbruch am Computer: Zeitungsherstellung heute

Technik macht Druck

Die Zeitungsherstellung hat gewaltige Fortschritte gemacht. Texte werden in Satzcomputer eingegeben, mit denen gleichzeitig das Layout gemacht wird. So spart man Zeit.

Bildfolgen

Eine Maschinencoderoutine, mit der man Bildschirminhalte umlegen und auch speichern kann. Auf dem Commodore läuft das sogar problemlos. Viel Spaß beim Ausprobieren.

FORTH-Arrays

Wir zeigen, wie man FORTH-Strukturen definieren kann. Standardmäßig geht's nicht.

Videosignale

Videosignale können mit kompakten und erschwinglichen Geräten digitalisiert werden. Interessante Möglichkeiten für Hobbyfilmer.