Einsteigen - Verstehen - Beherrschen

DM 3,80 öS 30 sfr 3,80

computer kurs

Heft 52

Ein wöchentliches Sammelwerk

Der Penman-Plotter

Elite der Großrechner

Schneider 6128

Bits und die Bibel

computer kurs

Heft 52

Inhalt

WIE SIE JEDE WOCHE IHR HEFT BEKOMMEN

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

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

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

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

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

Abonnement

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

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

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

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

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

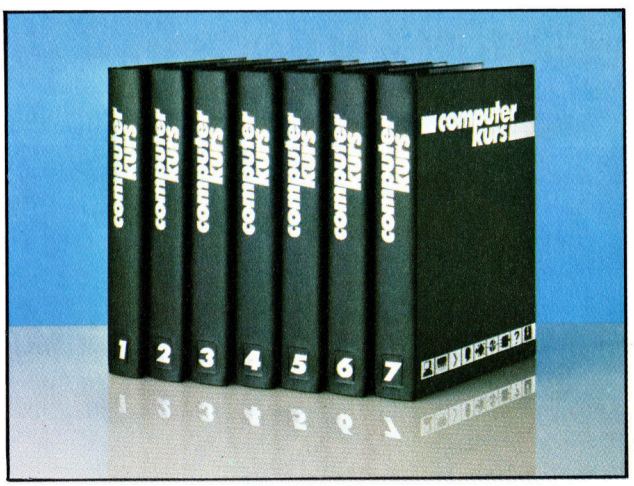
Schweiz: Der Sammelordner kostet sfr 15. Bitte wenden Sie sich an Ihren Kiosk; dort werden Sie jederzeit die gewünschten Exemplare erhalten.

INHALTSVERZEICHNIS

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

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

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



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

An der Leine

Relativ billige Bodenroboter („Schildkröten") sind vor allem in Schulen bei der Ausbildung im Programmieren beliebte Übungsobjekte. Auch der hier beschriebene „Penman Plotter" gehört in diese Kategorie.

Weil der Acorn B sehr flexible Schnittstellen bietet und in England in Schulen eingesetzt wird, gibt es dafür eine Vielzahl didaktisch orientierter Peripheriegeräte. Über etliches von diesem Zubehör wurde hier bereits berichtet, unter anderem über Plotter, Mäuse und Bodenroboter – der „Penman Plotter" erhebt den Anspruch, alles in einem zu sein.

Das Penman-Paket besteht aus Steuereinheit, Zeichenroboter, Netzteil, RS232-Anschlußkabel und Betriebssoftware. Bei Nichtgebrauch lassen sich Steuereinheit und Zeichengerät zu einem kompakten Kästchen von nur 55x128x335 mm Größe zusammensetzen. Um den Roboter aus seiner „Höhle" zu befreien, drücken Sie auf einen Clip an der Unterseite der Steuereinheit und ziehen den Einsatz aus dem Gehäuse heraus. Die Verbindung zwischen Steuereinheit und Roboter stellt ein Flachbandkabel her. Am Plotter selbst finden Sie drei Aufnahmebuchsen für Zeichenstifte und außerdem eine zentrale Bohrung, in die ein weiterer Stift für Turtle-Grafik paßt. Die 40 mm langen Filzstiftpatronen werden nach dem Einsetzen durch gefederte Klammern knapp über dem Papier in Bereitschaft gehalten. Wenn mit einem Stift gezeichnet werden soll, zieht ein Elektromagnet den betreffenden Federbügel herunter und die Schreibspitze drückt durch das Eigengewicht der Patrone aufs Papier.

An der Unterseite hat der Roboter drei Räder, von denen zwei angetrieben werden. Das dritte – eine kleine Kunststoffrolle mit losem Schwenklager – hält nur das Gleichgewicht. Die Metall-Antriebsräder tragen einen rauhen Belag, der die Reibung auf dem Papier verbessert. Vor beiden Rädern ist jeweils ein Lichtdetektor angebracht. Aufgrund des Helligkeitsunterschieds zwischen Papier und Unterlage wird so der Rand des Blatts erkannt.

Das Plottergehäuse enthält drei Elektromagnete zur Betätigung der Schreibstifte und die beiden Elektromotoren für den Radantrieb. Dabei handelt es sich nicht um Schrittmotoren, sondern um normale Gleichstromtypen, mit denen sich glatte Kurven anstelle von feinen Treppen malen lassen. Weil hier die Motorspannung ständig variiert werden muß, ist allerdings der Aufwand bei der Betriebssoftware höher als bei Schrittmotoren, wo nur die richtige Anzahl von Impulsen vorzugeben ist.

Der Penman besteht aus zwei Teilen: der Steuereinheit, die über eine RS232-Schnittstelle mit dem Rechner verbunden wird, und der beweglichen „Schildkröte". Das Flachbandkabel zwischen beiden dient zur Datenübertragung. Die Steuereinheit sendet darüber ihre Anweisungen für den Bewegungsablauf, und der Roboter meldet der Steuerung kontinuierlich seine Positionsänderungen.

Steuerung mit Rückmeldung

Auf einer kleinen Leiterplatte sind die Leistungselektronik und die Decodierer untergebracht, die die Entschlüsselung der Steuerungsinformationen für die Magnete und die Motoren besorgen. Jede Motorachse trägt eine Schlitzscheibe, die zwischen einer Lichtquelle und einer Fotodiode hindurchläuft. Dabei wird der Lichtstrahl periodisch unterbrochen. Und die Fotodiode gibt dementsprechend eine Impulsfolge an die Logik auf der kleinen Platine weiter, die dann der Steuereinheit die Informationen zur Bestimmung der Roboterposition übermittelt. Dank dieser Rückmeldung kann der Penman (bei abgeschaltetem Antrieb) ohne weiteres als Maus verwendet werden. Der Computer kennt Drehrichtung und -ge-

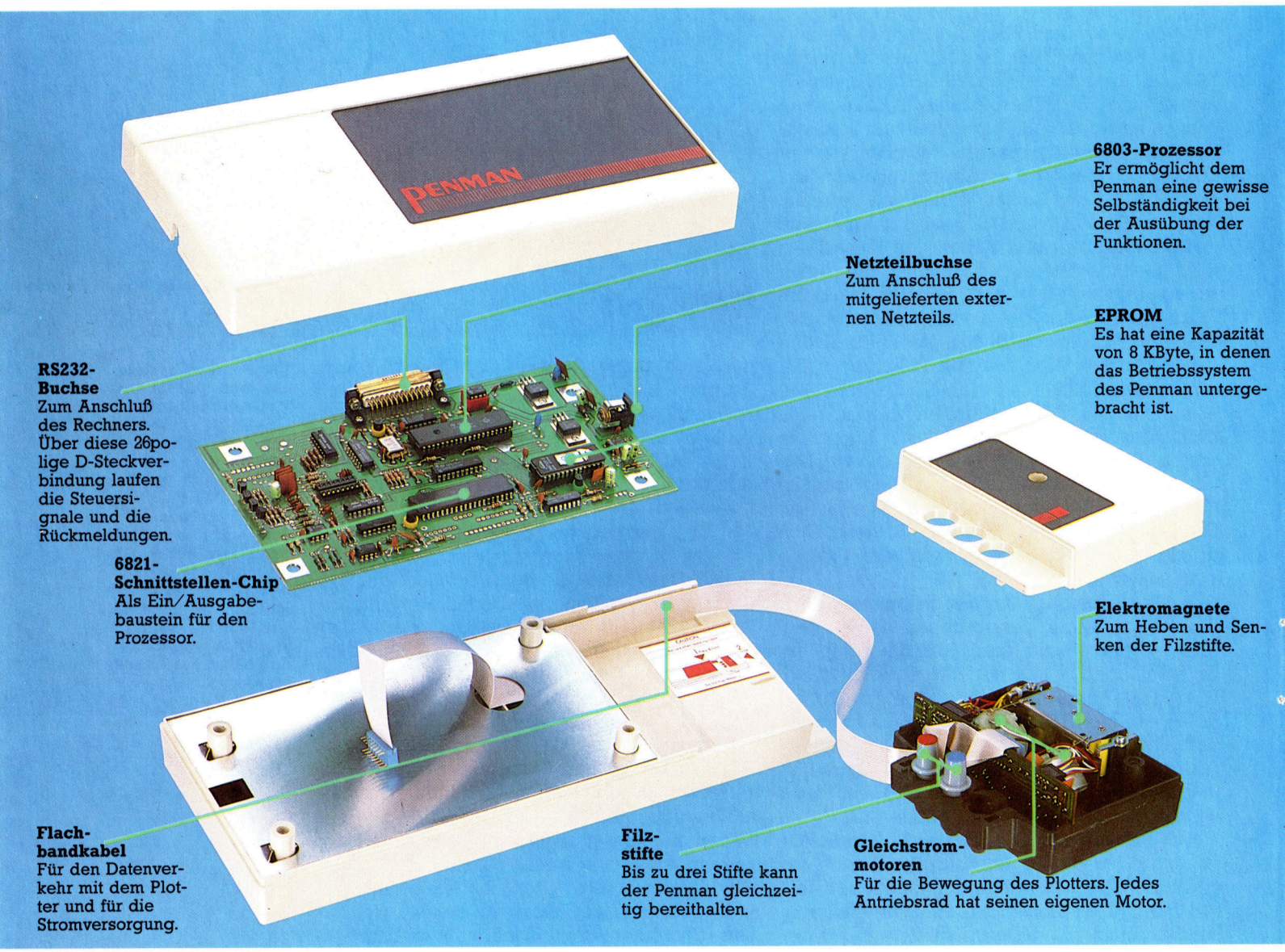
schwindigkeit der Räder und kann den Cursor auf dem Bildschirm entsprechend nachführen.

Das Innenleben des Steuergeräts besteht im wesentlichen aus einer großen Leiterplatte, auf der drei Chips besonders ins Auge fallen. Da ist zuerst der aus dem 6800 entwickelte Prozessorbaustein 6803, der sich durch hohe Flexibilität auszeichnet. Er enthält ein 128-Byte-RAM, das als „Merkzettel" für die Plotterposition verwendet wird. Der zweite auffällige Chip ist das Parallel-Interface 6821, über das der 6803 die Ein- und Ausgabe abwickelt. Und der dritte ist ein 8-KByte-EPROM mit den Demonstrationsprogrammen, die der Penman ohne Verbindung mit dem Rechner auch ganz allein ausführen kann.

Es sind die verschiedensten Betriebsarten vorgesehen. Im „Terminal Emulator"-Modus ist die unmittelbare Steuerung per Tastatur möglich. Dazu laden Sie die Pentlk-Software entweder direkt oder über das Hauptmenü, indem Sie den Roboter als Maus benutzen. Steht das Programm im Speicher, werden die Tastaturbefehle über die RS232-Schnittstelle in Form von ASCII-Zeichen an die Steuereinheit gesendet. Die Eingabe von PRINT'I' initialisiert den Penman. Bei der Befehlsübernahme erkennt das Gerät, welche der drei möglichen Übertragungsraten (300, 1200 oder 9600 Baud) eingestellt ist. Danach kann der Roboter durch Drükken von „H" (für Home) zum Aufsuchen der Startposition aufgefordert werden. Dies ist die linke untere Ecke des Blattes.

Die Home-Position

Der Plotter findet sie, indem er mit Hilfe der optischen Sensoren zunächst den unteren Rand des Papiers sucht. Dort dreht er nach rechts und forscht nach der linken Kante des Bogens. Um einen ausreichenden Kontrast zwischen Papier und Untergrund sicherzustellen, liefert die Herstellerfirma vorsorglich ein schwarzes Blatt als Unterlage mit. Nachdem die Home-Position erreicht ist, legt der Penman im Abstand von 50 mm vom unteren und vom



linken Papierrand den Koordinatenursprung fest (die 50 mm sind der Abstand der Mittelbohrung von der Turtle-Vorderkante).

Die Pentlk-Software unterstützt nicht nur die direkte Einzelbefehl-Eingabe, sondern auch den programmgesteuerten Betrieb durch Befehlsketten, die mit LOAD, SAVE und RUN gehandhabt werden. Im „Terminal Emulator"-Modus erfolgt die Bewegung in kartesischen Koordinaten, das heißt wie auf Millimeterpapier. Die Anweisung, den Punkt (500, 500) anzulaufen, bewirkt je 500 Zehntelmillimeter-Schritte in x- und in y-Richtung. Auf einem DIN-A4-Blatt sind maximal 2100 x- und 2970 y-Schritte möglich. Dem „Marschbefehl" MOVE ist ein A oder ein R voranzustellen, je nachdem ob absolute (auf den Ursprung bezogene) oder relative (auf die derzeitige Position bezogene) Zielkoordinaten angegeben werden. Die Schreibstiftsteuerung erfolgt über LOGO-ähnliche Kommandos wie „U" für „Pen Up" (Stift heben), „D" für „Pen Down" (Senken) und „P" für „Pen Select" (Auswahl des gewünschten Stiftes).

Auch ein Turtlegrafik-Betrieb ist bei direkter Eingabe möglich. Allerdings ist dies etwas umständlich, weil die Streckenlängen anders als kartesische Koordinaten hexadezimal eingegeben werden müssen. Das System greift dann unter Umgehung der Umrechnung unmittelbar auf die Speicherplätze zu, in denen die Steuerinformation steht. Sie können die Turtlegrafik aber mit Hilfe der mitgelieferten Software auch in LOGO programmieren.

Der Robotics-Modus

Im „Robotics"-Modus wird der Plotter direkt über die einzelnen Bits des User-Port-Datenregisters gesteuert. Zum Beispiel geben Bit 0 und 1 sowie Bit 2 und 3 die Spannung für den rechten bzw. linken Motor vor. Bit 4 und 5 sind für weitere Motorfunktionen wie das Ein- und Ausschalten zuständig, Bit 6 für die Schreibstift-Betätigung. Außerdem sendet der Penman über das Datenregister Rückmeldungen, etwa bei einer Fehlpositionierung oder beim Erreichen des Papierrands.

Sie können den Penman auch Texte schreiben lassen. Der Befehlsvorrat entspricht weitgehend dem der gängigen Printer/Plotter für Heimcomputer, und selbst in der Form der Zeichen gibt es kaum Unterschiede. Sie haben die Wahl zwischen Schrifthöhen von 1 bis 127 mm und vier verschiedenen Schreibrichtungen. Außerdem ist Schrägschrift möglich, was konventionelle Printer/Plotter meist nicht zu bieten haben.

Die mitgelieferte Dokumentation enthält eine vollständige Befehlsliste mit einigen Erläuterungen zur Implementierung und eine Beschreibung der Hardware. Allerdings setzt sie für den Anfänger vielleicht etwas zu hoch an. Die Dokumentation scheint sich an Benutzer zu richten, die sich mit ihrem Rechner schon gründlich auskennen.

Der Penman Plotter ist als ein weiterer Schritt in der Entwicklung eines universellen Turtle-Plotters anzusehen. Bei richtigem Gebrauch nähert sich seine Zeichengenauigkeit schon dem Bereich konventioneller Profi-Geräte. Für den kommerziellen Einsatz fehlt im Augenblick vor allem geeignete Software. Das derzeitige Paket ist vorwiegend für Ausbildungszwecke gedacht. Vom Benutzer wird erwartet, daß er sich die Programme für den Plotterbetrieb selbst schreibt, um dabei das Prinzip der Robotersteuerung kennenzulernen. Daran ist ein professioneller Anwender, der auf leichte Handhabung Wert legt, natürlich wenig interessiert.

Für die Ausbildung ist der Penman Plotter eine solide Sache. Es gibt auf dem Markt zwar billigere Geräte. Diese sind aber weniger vielseitig verwendbar. Mit dem Penman lassen sich die verschiedensten Anwendungsbereiche demonstrieren.

Penman Plotter

ABMESSUNGEN

335×128×55 mm

SCHNITTSTELLE

RS232C (bidirektional); erlaubt den Anschluß an eine Vielzahl gängiger Rechner.

AUFLÖSUNG

Schrifthöhe 1–127 mm; Grafik-Auflösung 0,1 mm.

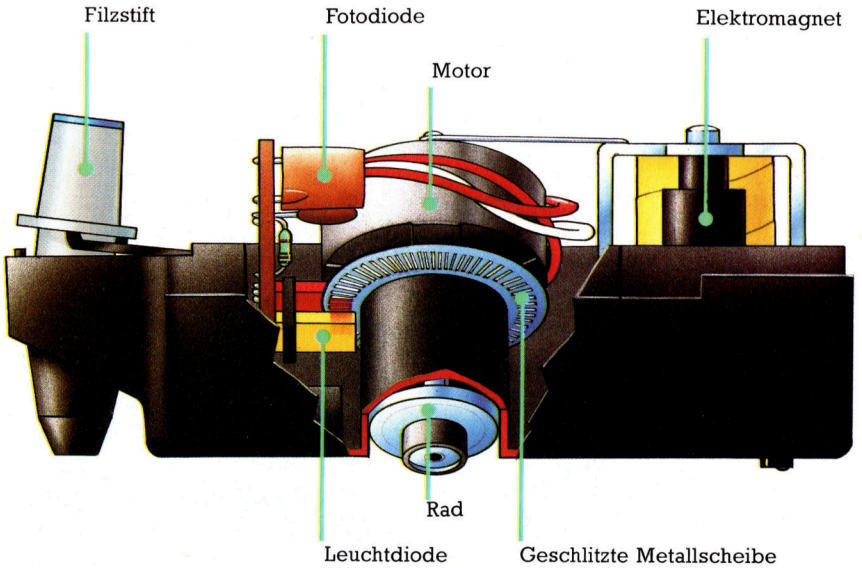
DOKUMENTATION

Die Anleitung enthält eine Menge technischer Details, die den fortgeschrittenen Benutzer alle Möglichkeiten des Penman ausschöpfen lassen. Für den Anfänger fehlt darin allerdings eine leicht verständliche Einführung.

Unter ständiger Kontrolle

Damit die Steuerung „weiß", wo sich der Plotter gerade aufhält, ist eine ständige Überwachung des Bewegungsablaufs unerläßlich. Auf jeder Achse ist eine radial geschlitzte Metallscheibe befestigt, die von einer Leuchtdiode angestrahlt wird.

Auf der gegenüberliegenden Seite der jeweiligen Scheibe ist eine Fotodiode angebracht, die die Lichtblitze beim Drehen der Scheibe in eine Folge elektrischer Impulse umsetzt. Aus der Anzahl der Impulse läßt sich errechnen, wieweit sich der Plotter bewegt hat.



Funktionelles

Wir zeigen, wie sich innerhalb von Listen andere Listen anlegen lassen, und definieren eigene Funktionen.

In der ersten Folge hatten wir untersucht, wie LISP veranlaßt wird, eine Liste als Datenliste und nicht als Funktion anzusehen. So ordnet

```
(SETQ X '(A B C D E F))
```

der Variablen X die Liste (A B C D E F) zu, wobei jedes Listenelement ein String mit einem einzigen Zeichen ist. Was aber passiert, wenn D, E und F Strings mit einzelnen Zeichen wären, A, B und C aber Variablen mit den Werten 2, 4 und 8? Für diesen Fall müssen wir X die Liste (2 4 8 D E F) zuordnen. Dies geschieht mit der neuen Funktion LIST, die eine Liste ihrer Argumente aufbaut.

```
(SETQ X '(LIST 'A 'B 'C 'D 'E 'F))
```

bringt das gleiche Ergebnis wie das vorige Beispiel, während:

```
(SETQ X '(LIST A B C 'D 'E 'F))
```

die Liste (2 4 8 D E F) zuordnet. Durch das Fehlen der Anführungsstriche werden A, B und C bewertet. Ebenso weist

```
(SETQ X (LIST 1 2 4 '(PLUS 4 4))
```

der Variablen X die Liste (1 2 4 (PLUS 4 4) zu, wobei das vierte Listenelement selbst wiederum eine Liste mit den drei Elementen PLUS, 4 und 4 ist. PLUS wurde nicht bewertet, da ein Anführungszeichen vor dem Ausdruck steht.

```
(SETQ X (LIST 1 2 4 (PLUS 4 4))
```

ordnet X jedoch die Liste (1 2 4 8) zu, da PLUS bewertet wurde. Ein einziges Anführungszeichen erzeugt diesen Unterschied. Mit dem Konzept läßt sich auch eine Liste von Listen aufbauen:

```
(SETQ PERSON '((KLAUS BLOCK)  (17 1
1960)))
```

ordnet der Variablen PERSON eine Datenliste zu, deren zwei Elemente wiederum Listen sind. Die erste Liste enthält zwei Elemente, die aus Zeichen bestehen, und die zweite Liste drei numerische Elemente. Innerhalb der Liste lassen sich weitere Listen anlegen.

Bisher haben wir uns die Funktionen SETQ, LIST, PLUS und TIMES angesehen. LISP verfügt jedoch noch über viele weitere Funktionen, deren Einsatz von der Anwendung abhängt.

Bevor wir uns ansehen, wie Funktionen angelegt werden, untersuchen wir zunächst drei Grundfunktionen, die in allen LISP-Versionen zu finden sind: CAR, CDR und CONS. Der Name CONS steht für das Wort CONStruct, während CAR und CDR aus einer der ersten Anwendungen stammen und Abkürzungen für „Contents of Address Register" und „Contents of Decrement Register" sind.

CAR und CDR

CAR und CDR müssen in jedem Fall ein einziges Argument haben, das eine Liste mit mindestens einem Element sein muß. Ausgeschlossen ist die „leere Liste" oder NIL, die

```
()
```

geschrieben wird. Die Funktion CAR liefert als Ergebnis das erste Element des Listenarguments.

```
(CAR '(1 2 3 4 5))
```

hat daher als Ergebnis die ganze Zahl 1. Das Ergebnis kann aber auch eine Liste sein. In dem Ausdruck

```
(CAR '((1 2) (3 4) 5))
```

ist das erste Element die Liste (1 2).

Die Funktion CDR liefert alle Elemente einer Liste außer dem ersten. So ergibt

```
(CDR '(1 2 3 4 5))
```

die Liste (2 3 4 5) und

```
(CDR '((1 2) (3 4) 5))
```

die Liste ((3 4) 5).

Die Funktion CONS fügt das erste Argument an das zweite an.

```
(CONS '(1 2) '((3 4) 5))
```

ergibt daher die Liste ((1 2) (3 4) 5).

Die Funktionen CAR und CDR werden oft ineinander verschachtelt. So ergibt

```
(CDR '((1 2) (3 4) 5))
```

die Liste ((3 4) 5) und

```
(CAR (CDR '((1 2) (3 4) 5)))
```

(3 4) – eigentlich (CAR '((3 4) 5)) –, während

```
(CAR (CAR (CDR '((1 2) (3 4) 5))))
```

den Wert 3 liefert – (CAR '(3 4)).

Da diese Verschachtelungen recht lästig werden können, erlaubt LISP Abkürzungen. Die Funktionsnamen fangen dabei zwar immer noch mit C an und enden mit R, können dazwischen aber jede beliebige Anzahl As (für CAR) oder Ds (für CDR) enthalten. Das Acornsoft-LISP läßt jedoch nur bis zu drei Einfügungen zu. Der letzte Ausdruck ließe sich nun auch

```
(CAAR (CDR '((1 2) (3 4) 5)))
```

schreiben oder

```
(CAR (CADR '((1 2) (3 4) 5)))
```

oder auch

```
(CAADR '((1 2) (3 4) 5))
```

Alle Versionen liefern das Ergebnis 3.

Funktionen werden mit der Funktion DEFUN definiert, die drei Argumente braucht und folgendes Format hat:

```
(DEFUN a (b) (c))
```

Dabei ist a der Name der Funktion, b die Parameterliste (ähnlich wie in PASCAL oder Acorn-B-BASIC) und c eine Listenstruktur mit dem Hauptteil der Funktion. Wenn wir beispielsweise oft mit 8 multiplizieren müssen, könnten wir natürlich jedesmal schreiben:

```
(TIMES 8 N)
```

wobei N die zu multiplizierende Zahl darstellt. Eine selbstdefinierte Funktion könnte jedoch die gleiche Aufgabe ausführen:

```
(DEFUN TIME8 (N) (TIMES 8 N))
```

Die Funktion TIME8 nimmt N als Argument, multipliziert diese Zahl über die Standardfunktion mit 8 und liefert das Ergebnis.

```
(TIME8 11)
```

ergibt 88.

Bevor wir weitere Funktionen definieren, wollen wir jedoch erst einmal ein wichtiges Programmierkonzept untersuchen: die Bedingung (COND). Sie ist etwa so wichtig wie der IF...THEN-Befehl in BASIC.

Auch Bedingungen haben in LISP die Form von Funktionen:

```
(COND (Bedingung1 Ausdruck1)
      (Bedingung2 Ausdruck2)
                "     "
      (BedingungX AusdruckX))
```

Beachten Sie, daß diese Funktion sich über mehrere Zeilen erstrecken muß, um überhaupt ausreichend Platz zu haben. Diese Anordnung wird in LISP oft eingesetzt, da die Sprache an der letzten Klammer erkennt, wo das Ende des Ausdrucks liegt.

```
(DEFUN EQUAL (A B) (COND
   ((EQ A B) T)
   ((OR (ATOM A) (ATOM B)) NIL)
   ((EQUAL (CAR A) (CAR B)) (EQUAL (CDR A) (CDR B))) (T NIL)))
(DEFUN FIND__AUTO (X L) (COND
   ((NULL L) '(KEINE AUTONUMMER))
   (EQUAL X (CDAR L)) (CAR L))
   (T (FIND__AUTO X (CDR L)))))

(SETQ AUTOS '((SCHMITT HHCS1254) (MEIER PIOS2354)
              (SCHANZE BIDD1984) (STEIN BTT336))
(PRINTC (FIND__AUTO '(BIDD1984) AUTOS))
```

Dieses einfache Programm zum Auffinden von Daten zeigt den Einsatz der Funktionen CDR und EQ. Dabei liefert eine vom Anwender eingegebene Autonummer – falls in der Datenbank des Programms vorhanden – diese Nummer und den Nachnamen des Besitzers.

Die COND-Funktion ähnelt dem CASE-Befehl von PASCAL. Dabei wird Bedingung1 bewertet. Ist sie wahr, dann liefert Ausdruck1 das Ergebnis. Ist sie falsch, wird Bedingung2 getestet etc. Einige LISP-Versionen (auch Acornsoft-LISP) verlangen, daß zumindest eine der Bedingungen („Aussagen" genannt) wahr ist. Dies läßt sich jedoch leicht durch das Hinzufügen einer Endbedingung erreichen:

```
(COND (Bedingung1 Ausdruck1)
      (Bedingung2 Ausdruck2)
                "     "

(COND (T     Letzter Ausdruck))
```

Das Zeichen T stellt hier „True" (wahr) dar. Der letzte Ausdruck (falls er erreicht wird) ist immer bewertbar. In LISP ist:

```
T = True = Nicht Null
```

und

```
F = False = Null = ()
```

Die Klammern stellen die leere Liste dar. Wir können nun schon kompliziertere Funktionen definieren.

Fast alle BASIC-Dialekte kennen die Funktion ABS, die als Ergebnis den positiven Wert ihres Arguments liefert. Das heißt: ein positives Argument bleibt unverändert, ein negatives wird umgekehrt. In LISP hat diese Funktion folgendes Format:

```
(DEFUN ABS (X)
   (Funktionsinhalt))
```

wobei X ein ganzzahliges Argument ist. Mit der Bedingungsfunktion COND sieht die vollständige ABS-Funktion folgendermaßen aus:

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

Die Definition enthält zwei neue Funktionen. Die Testfunktion MINUSP liefert einen Wert, wenn das Argument eine negative Zahl ist. Bei einer positiven Zahl ist das Ergebnis „False". Ist die Bedingung wahr, negiert die Funktion MINUS ihr Argument, wenn nicht, liefert T (die immer wahr ist) den ursprünglichen Wert.

Testfunktionen
Die folgende Liste zeigt Testfunktionen, die in LISP oft eingesetzt werden:

(OR Arg1 Arg2 . . .)	Liefert „True", wenn mindestens eins der Argumente wahr ist
(ONEP Arg)	Liefert „True", falls Arg = 1
(ZEROP Arg)	Liefert „True", falls Arg = 0
(LISTP Arg)	Liefert „True", falls Arg eine Liste ist, und „false", wenn es ein Atom ist
(ATOM Arg)	Liefert „True", falls Arg ein Atom ist, und „False", wenn es eine Liste ist
(LESSP Arg1 Arg2)	Liefert „True", wenn Arg1 < Arg2
(GREATERP Arg1 Arg2)	Liefert „True", falls Arg1 > Arg2
(EQ Arg1 Arg2)	Liefert „True", falls Arg1 = Arg2

Bits und Bibel

„The Word" ist ein speziell auf den englischsprachigen Theologiestudenten ausgerichtetes Programm mit Such- und Indexfunktionen. Obwohl dieses Paket sich an einen begrenzten Anwenderkreis richtet, enthält es Programmiertechniken, die auch in vielen anderen Bereichen eingesetzt werden könnten.

The Word
Für MS-DOS-Geräte

HERSTELLER
Access Software Ltd, 36 Aybrook Street, London W1M 3JL

AUTOREN
Bible Research Systems, Austin, Texas

FORMAT
Disketten

Einige Berufe oder Studiengänge erfordern häufig das Durchsuchen langer Texte nach bestimmten Wörtern. So arbeiten Bibelstudenten mit sogenannten Konkordanzen (Spezialindizes) für die 39 Bücher des Alten und die 27 Bücher des Neuen Testaments – insgesamt etwa 750 000 Wörter.

Im allgemeinen eignen sich Computer bestens für das Durchsuchen großer Textmengen, doch setzen die relativ kleinen Speicher- und Diskettenkapazitäten der Heimgeräte hier oft enge Grenzen. Speziell für Microcomputer gibt es jetzt jedoch ein Programm, das das Auffinden von Bibelstellen erleichtert. „The Word" findet dabei nun nicht nur bestimmte Textstellen, sondern legt auch Indizes an, die die Suchzeit erheblich verkürzen.

Der Umfang der von den Programmierern geleisteten Arbeit wird schon beim Öffnen des Paketes deutlich. Das Handbuch ist zwar nur ein dünnes Büchlein von 36 Seiten, das Programm mit den biblischen Textdateien besteht jedoch aus sieben doppelseitigen Disketten, die alle kopiert werden müssen. Bei einem einseitigen Diskettenlaufwerk mit 40 Spuren sind sogar 14 Disketten für Arbeitskopien nötig.

Es werde Licht

„The Word" muß sorgfältig installiert werden. Obwohl die Programmierer nicht davon ausgehen konnten, daß die meisten Theologiestudenten sich auch mit Computern gut auskennen, ist das BASIC-Programm recht kompliziert aufgebaut. Zum Beispiel akzeptiert es nur PC-DOS. Der BASIC-Interpreter und die Systemspuren müssen von der DOS-Diskette auf die Programmdiskette kopiert werden. Dann kann der Student wählen, ob er das Programm TWP aufrufen will (wenn er einen IBM PC besitzt) oder die Version ATWP (bei einem IBM-kompatiblen Gerät) oder eine Spezialversion für Computer mit einem Arbeitsspeicher von nur 64 KByte: TWP 64. Ein erfahrener Anwender würde nun davon ausgehen, daß nach Installation der Systemspuren, dem BASIC und den Programmen die Diskette automatisch booten würde. Doch es gibt kein Programm hierfür.

Schon beim Aufrufen der Titelseite ist erkennbar, daß es sich nicht um gewöhnliche Software handelt, da außer dem üblichen Copyrightvermerk („Diese Software ist durch die Copyrightgesetze der USA geschützt") auch ein passendes Bibelzitat aufgeführt wird (hier in deutsch wiedergegeben):

EXODUS 20 : 15 „Du sollst nicht stehlen."

Danach folgt das Hauptmenü (im Original natürlich englisch):

<F1> — HILFSTEXTE AUFBLENDEN
<F2> — STEUEROPTIONEN SETZEN
<F3> — DRUCK IST ABGESCHALTET. ANSCHALTEN
<F4> — BEREICH:
<F5> — ENDE DER BEARBEITUNG
<F6> — INDEX AUFBAUEN
<F7> — INDEX ANZEIGEN ODER VERÄNDERN
<F8> — INDIZES VERBINDEN
<F9> — INDEX LÖSCHEN
<F10> — TEXTE ANZEIGEN

<F2> blendet ein Untermenü mit neun Steuermöglichkeiten auf, darunter die Bildschirmbreite (40 oder 80 Zeichen), die linke und rechte Randbreite des Druckers, Zeilenabstand, Anzahl der Zeilen pro Seite, maximale Indexgröße, Anzeige eines Textes (oder einer Reihe von Texten im Zusammenhang) und die Namen aller Indizes auf der aktuellen Diskette.

<F4> gibt die Möglichkeit, mit der großen Datenmenge fertigzuwerden, indem ein beschränkter Bereich festgelegt wird, der bearbeitet werden soll. Mit einer aus drei Buchstaben bestehenden Kennzeichnung der Bibelbücher (von „Gen" bis „Rev" für „Offenbarung") und den Zahlen für Kapitel und Vers kann Anfang und Ende des Suchbereichs angegeben werden. Die Kapitel- und Verszahlen können weggelassen werden, wenn ein ganzes Buch bearbeitet werden soll.

Nach dem Einlegen der entsprechenden Diskette können die Kapitel betrachtet werden. <F10> blendet den ersten Vers des Suchbereichs auf. Mit den auf dem unteren Bildschirmrand aufgeblendeten Funktionstasten kann man nun im Text vor- und zurückblättern. Die Steueroption <F8> veranlaßt, daß nicht nur ein Vers angezeigt wird, sondern auch die angrenzenden Texte im Zusammenhang.

Mit <F2> lassen sich Suchkriterien einrichten. Wie bei den meisten anderen Datenbanken kann jedes Vorkommen eines Wortes gesucht werden, selbst in zusammengesetzten

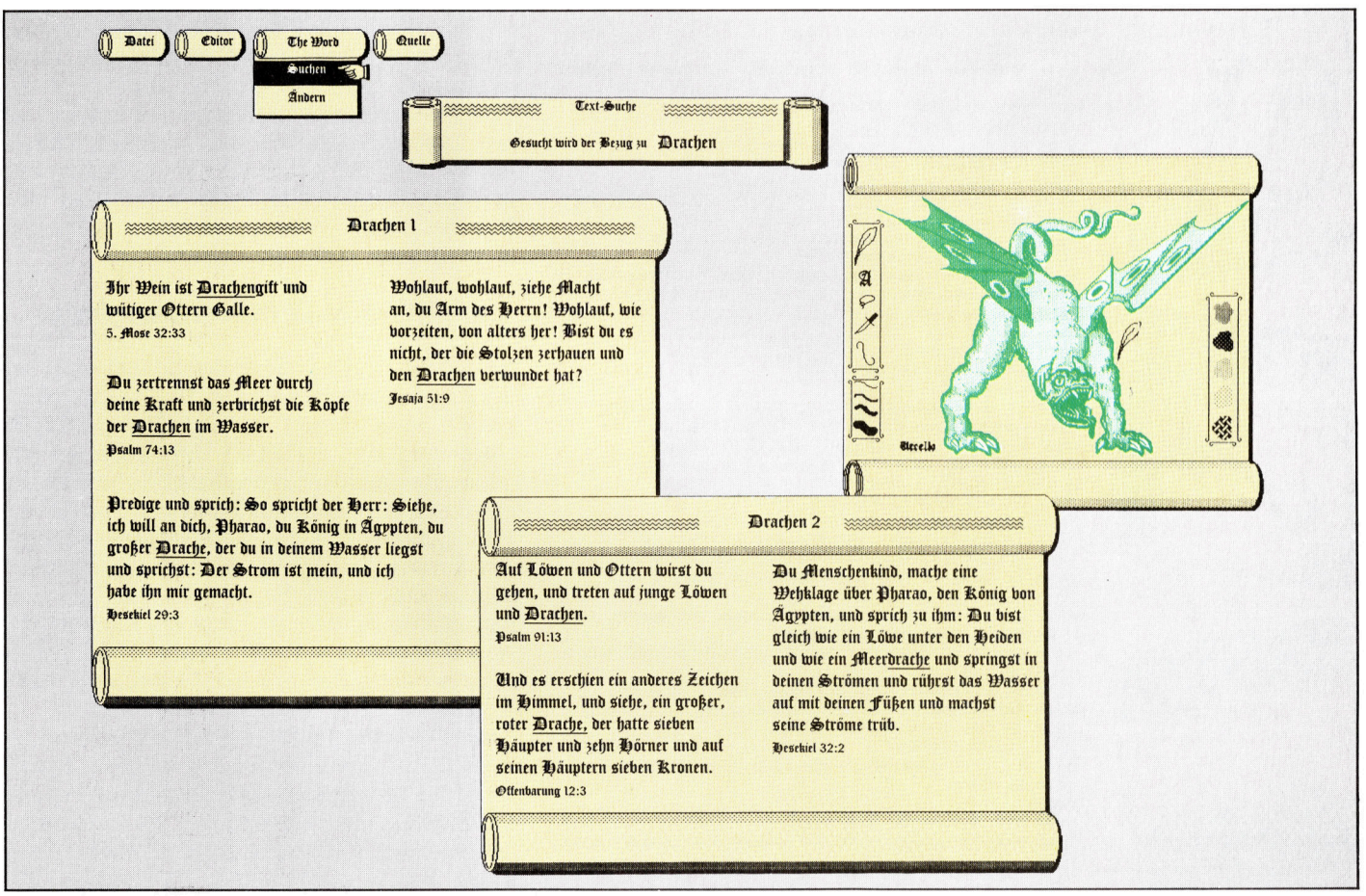
Wörtern. Die Suche nach dem Wort „man" würde dabei auch „demand" zutage fördern. Es kann jedoch auch nur nach dem vollständigen einzelnen Wort gesucht werden. Auch sogenannte „Wild Cards" sind möglich. Dabei werden alle Wörter gefunden, die mit der gesuchten Buchstabenkombination anfangen oder enden. Nach Eingabe der Kriterien werden die Textstellen entweder einzeln oder im Zusammenhang angezeigt.

Mit <F6> werden automatisch oder per Hand Indizes angelegt. Die Indizes lassen sich editieren (beispielsweise die Löschung nicht gewünschter Verweise) und zusammenlegen (zwei oder mehr) – vorausgesetzt, die Grenze von 1020 Einträgen pro Index wurde nicht überschritten. Die Grenze kann jedoch mit <F2> bis auf 54 040 heraufgesetzt werden. Enthält ein Index mehr als 1020 Einträge, wird er stückweise gespeichert. So entsteht eine Pause zwischen dem Eintrag 1020 und 1021, da der nächste Teilindex erst von der Diskette geladen werden muß. Der Anwender kann auch eigene Indizes mit den Themen der Verse eingeben.

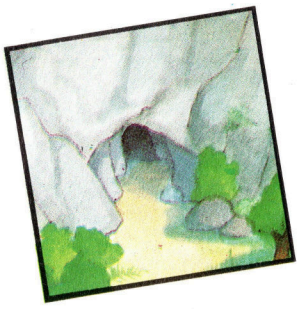
Abgesehen von den Problemen bei der Verwaltung großer Textmengen, hat The Word noch mit völlig anderen Schwierigkeiten zu kämpfen. Das Programm basiert auf der (englischen) King-James-Version von 1611. In den letzten drei Jahrhunderten hat sich jedoch die Bedeutung vieler Wörter radikal verändert. Wie die Übersetzer der revidierten Version von 1952 angeben, sind über dreihundert Bedeutungsänderungen eingetreten. Dabei sind Fehler, die seit dem 17. Jahrhundert durch die Entdeckung genauerer, ursprünglicherer Texte zutage traten, noch nicht berücksichtigt.

Das Zusatzprogramm, „The Greek Transliterator", beseitigt dieses Problem jedoch zumindest für das Neue Testament. Das Programm durchsucht den griechischen Text nach einem bestimmten Wort der King-James-Version. Dabei erscheint außer dem ursprünglichen griechischen Wort (in römischer Schrift, nicht in griechischen Buchstaben) auch die Definition und – falls notwendig – die Häufigkeit des Vorkommens und die Bedeutungen, die es in verschiedenen Zusammenhängen angenommen hat. Indizes, die mit TWP aufgebaut wurden, funktionieren auch mit dem Zusatzprogramm, und auch griechische Indizes sind möglich.

Bis zum heutigen Tage existiert noch kein Programm für das hebräische Alte Testament, obwohl in Jerusalem bereits ein Großcomputer den gesamten Talmud und zugeordnete Werke speichert. Ein sinnvolles computerisiertes Bibelstudium läßt sich eher mit einer öffentlichen Datenbank als mit einem individuellen Textsystem durchführen. Die in The Word eingesetzten Techniken sind jedoch sehr interessant und lassen sich vermutlich auch für die kritische Betrachtung anderer großer Textmengen einsetzen: die Werke von Shakespeare, Goethe oder auch die von Marx und Engels.

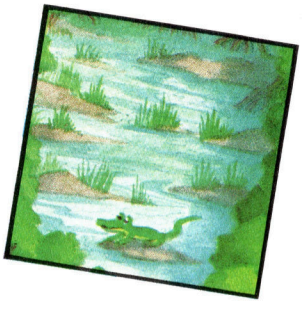


Bei der Suche nach dem Wort „Drachen" fand The Word diese und andere Textstellen. Wie eine Konkordanz vereinfacht das Programm die Suche nach identischen biblischen Ideen, Symbolen und Persönlichkeiten. Dem System liegt die (englische) King-James-Version der Heiligen Schrift zugrunde.



Tod im Sumpf?

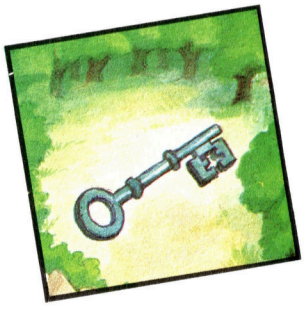
Unser Adventure-Projekt nähert sich langsam dem Ende. Wir besprechen in diesem Kapitel die Ausarbeitung der letzten beiden Orte von Haunted Forest – den Sumpf und das Dorf –, und das Listing für unser Digitaya-Abenteuer wird vervollständigt.

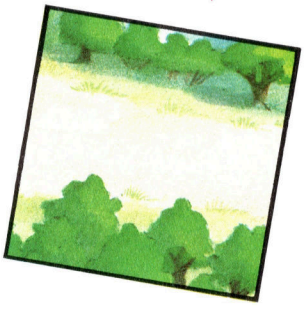


Als erstes geht es um den Sumpf von Haunted Forest. Auch hier muß eine Rahmenhandlung festgelegt werden, bevor mit dem Programmieren begonnen werden kann. Diese Handlung kann nach Belieben kompliziert oder einfach gehalten sein. Eine zu komplexe Handlung kann aber einen enormen Programmieraufwand und eine übermäßige Belegung des verfügbaren Speicherplatzes bewirken.

Es wird folgende Rahmenhandlung angenommen: Sobald der Spieler den Sumpf betritt, versinkt er. Alle „normalen" Befehle sind verfügbar, den Sumpf aber kann er nur verlassen, indem er ihn durchschwimmt.

Schwimmen ist nur möglich, wenn weniger als zwei Objekte mitgeführt werden, da deren Gewicht den Spieler nach unten zieht. Notfalls muß ein Objekt abgelegt werden, das damit verloren ist und später nicht mehr aufgenommen werden kann.





```
4870 REM **** SWAMP S/R ****
4875 SF=1
4880 SN$="YOU START TO SINK INTO THE SWAMP.":GOSUB
5500
4885 PRINT:INPUT"INSTRUCTIONS";IS$
4890 GOSUB2500:REM SPLIT INSTRUCTION
4895 IF F=0 THEN 4885:REM INVALID
4900 GOSUB3000:REM NORMAL INSTRUCTIONS
4910 IF VB$="LOOK" THENGOSUB2000:GOTO4885
4915 IF VB$="DROP"THEN IV$(F,2)="-2":REM OBJ LOST
FOREVER
4917 IF VF=1 THEN 4885:REM NORMAL COMMAND
4920 REM ** NEW COMMANDS **
4925 IF VB$<>"SWIM"THEN SN$="I DON'T UNDERSTAND":G
OSUB5500:GOTO4885
4930 REM ** SWIM **
4932 F=0
4935 FOR I=1 TO 2
4940 IF IC$(I)<>""THEN F=F+1
4950 NEXT I
4955 IF F<2 THENGOSUB5035:RETURN:REM SWIM AWAY
4960 GOSUB 5000:RETURN:REM TWO OBJS HELD
5000 REM **** TWO OBJECTS HELD S/R ****
5010 SN$="THE OBJECTS ARE WEIGHING YOU DOWN AND YO
U ARE SINKING.":GOSUB5500
5012 PRINT:INPUT"INSTRUCTIONS";IS$
5015 GOSUB2500:REM SPLIT INSTRUCTION
5020 IF VB$<>"DROP" THENGOSUB5080:REM SINK
5025 GOSUB3900:IV$(F,2)="-2":REM DROP OBJ
5030 IF HF=0 OR F=0 THEN 5080:REM SINK
5035 REM **** SWIM AWAY ****
5040 SN$="YOU CAN NOW SWIM THROUGH THE SWAMP. WHIC
H WAY WILL YOU GO?":GOSUB5500
5050 EX$(2)="00080605":GOSUB2300:REM DEFINE AND DI
SPLAY EXITS
5055 PRINT:INPUT"INSTRUCTIONS";IS$
5060 GOSUB2500:REM SPLIT INSTRUCTION
5062 IF F=0 THEN 5055:REM INVALID
5065 GOSUB3500:REM MOVE
5067 EX$(2)="00000000":REM ZERO EXIT DATA
5070 RETURN
5075 :
5080 REM **** SINK S/R ****
5085 SN$="YOU SINK INTO THE SWAMP AND DROWN":GOSUB
5500
5090 END
```

Die Zeilen 4870 bis 4917 ermöglichen unter Verwendung der bisher erstellten Unterroutinen eine Handhabung aller normalen Befehle. Will der Spieler am Rande des Sumpfes ein Objekt ablegen, wird die DROP-Routine ausgerufen. Diese Routine setzt die Position des abgelegten Objektes jedoch in der Hauptübersicht IV$(,) unter Verwendung des aktuellen Wertes des Ortszählers P ein: Das Objekt verbleibt im Sumpf, solange das Programm arbeitet. Soll ein Objekt, das in den Sumpf geworfen wurde, gelöscht werden, müssen wir den Inhalt des Strings IV$(,) aktualisieren.

IV$(F,2) enthält die Position von Objekt F. In der Regel ist dies die Ortszahl oder −1, wenn der Spieler das Objekt trägt. Um das Objekt aus dem Spiel zu entfernen, darf IV$(F,2) nicht mehr als Ort bzw. als Anzeige für den Spieler interpretiert werden. In Zeile 4915 erhält IV$(F,2) den Wert −2, wobei F bereits durch die DROP-Routine als relevantes Objekt bezeichnet wurde.

Wählt der Spieler SWIM aus, verzweigt das Programm zur Zeile 4930. Hier wird eine Inventur vorgenommen, um zu prüfen, wieviele Objekte der Spieler bei sich trägt. Bei weniger als zwei Objekten ruft das Programm die Unterroutine SWIM AWAY auf, durch die der Spieler den Sumpf verlassen kann. Trägt der Spieler zwei Objekte, kann er eines ablegen. Nimmt er diese Chance nicht wahr, versinkt er.

SWIM-AWAY öffnet Ausgänge

Die SWIM-AWAY-Routine ermöglicht eine Angabe der Schwimmrichtung. Die Ausgangsdaten für den Sumpf in der Original-DATA-Anweisung sind 00000000. Dies heißt, es gibt keinen Ausgang aus dem Sumpf. Zeile 5050 definiert nun die Ausgangsdaten neu und ruft die Unterroutine auf, in der die Ausgänge angegeben werden. Der Spieler kann einen Ausgang wählen und so den Sumpf verlassen. Die Ausgangsdaten werden am Ende der Routine wieder auf 0 gesetzt. Bei einem neuen Betreten des Sumpfes kann der Spieler diesen nicht mehr verlassen.

Das Dorf liegt am Fluchtweg aus Haunted Forest, obwohl der Spieler noch mehrere Aufgaben lösen muß, bevor das Spiel beendet ist. Als Rahmenhandlung verwenden wir folgendes: Das Dorf ist von einer hohen, unüberwindlichen Mauer mit einem bewachten Tor umgeben. Um

in das Dorf zu gelangen, muß der Spieler zuerst die Wache mit dem Gewehr töten. Dann muß er das Tor mit einem Schlüssel öffnen, um den Wald verlassen zu können.

Die DORF-Routine besteht aus zwei Teilen. Zuerst muß die Wache beseitigt und dann das Tor mit dem Schlüssel geöffnet werden. Es ist nicht schwer, sich vorzustellen, wie der Spieler mit einem Gewehr das Dorf erreicht und die Wache tötet, um dann festzustellen, daß er zum Betreten des Dorfes einen Schlüssel braucht. Hat er den Schlüssel nicht, muß er den Ort verlassen, um ihn zu suchen. Kommt der Spieler dann wieder zum Dorf zurück, darf die Wache hier nicht mehr zu sehen sein. Nach dem Schuß muß also die Wache vom Programm entfernt werden, und bei einem neuerlichen Durchgang darf sie nicht wieder erscheinen.

Die Wache lebt

Das klingt schwerer als es ist. Mittels eines Flags kann erkannt werden, ob die Wache tot ist. Dazu wird die Variable GF verwendet, die anfangs auf 0 steht (Wache lebt). Tötet der Spieler die Wache, wird die Variable auf 1 gesetzt. Der Wert von GF kann beim Betreten des Dorfes abgefragt werden, um festzustellen, ob die Wache lebt oder nicht. Ist keine Wache vorhanden, kann der Spieler sofort ans Tor gehen. Mit den bisher erstellten Unterroutinen können nun Anweisungen aufgeteilt und gehandhabt werden. Hat der Spieler den Schlüssel bei sich und öffnet damit das Tor, ist er am Ziel angekommen, und das Spiel ist beendet.

```
5100 REM **** VILLAGE S/R ****
5102 SF=1
5105 SN$="THE VILLAGE IS SURROUNDED BY A TALL WALL
.":GOSUB5500
5106 IF GF<>0 THEN GOSUB5190:RETURN:REM GATE
5107 SN$="A GUARD IS AT THE ENTRANCE GATE TO THE V
ILLAGE":GOSUB5500
5115 PRINT:INPUT"INSTRUCTIONS";IS$
5120 GOSUB2500:IF F=0 THEN 5115:REM INVALID
5125 GOSUB3000:REM NORMAL INSTRUCTIONS
5130 IF VB$="LOOK"THEN GOSUB2000:REM DESCRIBE
5135 IF VB$="GO" AND MF=1 THEN RETURN
5140 IF VF=1 THEN 5115:REM NEXT INSTRUCTION
5145 IF VB$<>"KILL"THENSN$="I DON'T UNDERSTAND":GO
SUB5500:GOTO5115
5150 REM ** KILL **
5155 SN$="WHAT WILL YOU USE TO KILL THE GUARD?":GO
SUB5500
5160 SN$="ENTER OBJECT OR <I> FOR INSTRUCTION":GOS
UB5500
5162 INPUT IS$:IF IS$="I" THEN 5115
5165 GOSUB2500:REM SPLIT
5167 IF F=0 THEN 5160:REM INVALID
5170 GOSUB5300:IF F=0 THEN SN$="THERE IS NO "+W$:G
OSUB5500:GOTO5160
5172 OV=F:GOSUB5450:REM IS OBJECT HELD
5174 IF HF=0 THEN SN$="YOU DO NOT HAVE THE "+IV$(F
,1):GOSUB5500:GOTO5160
5175 IF F<>1 THEN SN$="THE "+IV$(F,1)+" IS NO USE"
:GOSUB5500:GOTO5160
5180 SN$="YOU KILL THE GUARD":GOSUB5500:GF=1
5185 :
5190 REM **** LOCKED GATE S/R ****
5195 SN$="YOU MOVE FORWARD AND TRY TO OPEN THE GAT
E TO THE VILLAGE"
5200 SN$=SN$+" BUT THE GATE IS LOCKED AND WILL NOT
 MOVE":GOSUB5500
5205 PRINT:INPUT"INSTRUCTIONS";IS$
5210 GOSUB2500:IF F=0 THEN 5205:REM INVALID
5215 GOSUB3000:REM NORMAL INSTRUCTIONS
5220 IF VB$="LOOK"THEN GOSUB2000:REM DESCRIBE
5225 IF VB$="GO" AND MF=1 THEN RETURN
5230 IF VF=1 THEN 5205:REM NEXT INSTRUCTION
5232 IF VB$="USE"THEN 5240
5234 IF VB$="UNLOCK"THENSN$="HOW?":GOSUB5500:GOTO5
205
5235 SN$="I DON'T UNDERSTAND":GOSUB5500:GOTO5205
5240 GOSUB5300:REM VALID OBJECT
5242 OV=F:GOSUB5450:REM IS OBJ CARRIED
5244 IF F=0 THEN SN$="THERE IS NO "+W$:GOSUB5500:G
OTO5205
5246 IF HF=0 THEN SN$="YOU DO NOT HAVE THE "IV$(F,
1):GOSUB5500:GOTO5205
5248 IF F<>3 THEN SN$="THE "+IV$(F,1)+" IS NO USE"
:GOSUB5500:GOTO5205
5250 REM ** THROUGH GATE AND SAFE **
5255 SN$="YOU UNLOCK THE GATE, AND DISGUISING YOUR
SELF IN THE DEAD"
5260 SN$=SN$+" GUARD'S CLOTHES, WALK UNNOTICED THR
OUGH THE VILLAGE"
5265 SN$=SN$+" AND THE SAFETY OF THE OUTSIDE WORLD
.":GOSUB5500
5270 END
```

Um beide Ort-Routinen aufrufen zu können, wird Zeile 2720 der Unterroutine, die entscheidet, ob es sich um einen besonderen Ort handelt, wie folgt geändert:

```
2720 ON P GOSUB4590,4870,5100,4590
```

BASIC-Dialekte

Spectrum
Im Listing Haunted Forest wird SN$ durch S$, IS$ durch T$, VB$ durch B$, IV$(,) durch V$(,), EX$() durch X$() und IC$() durch I$() ersetzt. Folgende Zeilen werden ersetzt:

```
2720 IF P=1 THEN GOSUB 4590
2722 IF P=2 THEN GOSUB 4870
2724 IF P=3 THEN GOSUB 5100
2726 IF P=4 THEN GOSUB 4590

4937 LET A$=IC$(I):GOSUB7000
4940 IF A$<>"" THEN LET F=F+1

5175 IF F<>1 THEN LET S$="THE "
     :LET A$=V$(F,1):GOSUB7000
5177 IF F<>1THEN LET S$=S$+"IS NO
     USE":GOSUB5500:GOTO5160

5248 IF F<>3 THEN LET S$="THE"
     :LET A$=V$(F,1):GOSUB7000
5249 IF F<>3 THEN LET S$=S$+
     "IS NO USE":GOSUB5500:GOTO5205
```

Ersetzen Sie im Digitaya-Listing SN$ durch S$, IS$ durch I$, VB$ durch B$, IV$(,) durch V$(,) und IC$() durch I$(). Ändern Sie folgende Zeilen:

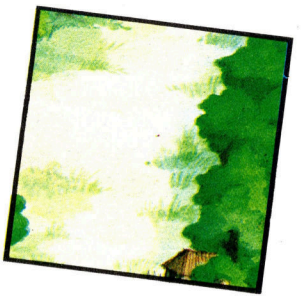
```
3650 LET S$="YOUR ":LET A$=V$(F,1):
     GOSUB 7000
3655 LET S$=S$+" IS USELESS, THE
     FORCE INCREASES"

4520 LET V$(4,2)=STR$(INT(RND(1)
     *40+8)
```

Acorn B
Ergänzen Sie folgende Zeile im Listing Haunted Forest:

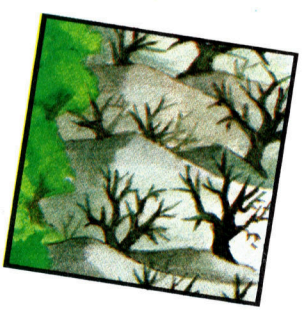
```
190 LET GF=0
```

Im Digitaya-Listing wird folgende Zeile ausgetauscht:

```
4520 IV$(4,2)=RND(40)+8
```








Digitaya

```
2960 REM **** USER PORT S/R ****
2970 SF=1
2980 SN$="ESCAPE IS AT HAND BUT THE DDR BOOKING CL
ERK"
2990 SN$=SN$+" BARS YOUR WAY. HE TELLS YOU THAT HE
 HAS BEEN INSTRUCTED TO"
3000 SN$=SN$+" ACCEPT INPUTS ONLY. HOWEVER HE DOES
 TAKE ALL MAJOR"
3010 SN$=SN$+" CREDIT CARDS."
3020 GOSUB 5880:REM FORMAT PRINT
3030 :
3040 PRINT:INPUT"INSTRUCTIONS";IS$
3050 GOSUB1700:REM ANALYSE INSTRUCTIONS
3060 GOSUB1900:REM NORMAL ACTIONS
3070 IF MF=1 THEN RETURN:REM MOVE OUT
3080 IF VF=1 THEN3040:REM NEXT INSTRUCTION
3090 IF VB$<>"GIVE" THENPRINT"I DON'T UNDERSTAND":
GOTO3040
3100 :
3110 REM ** INSTRUCTION IS GIVE **
3120 GOSUB5730:REM IS OBJECT VALID
3130 IFF=0THENPRINT"THERE IS NO ";NN$:GOTO3040:REM
 NEXT INSTRUCTION
3140 :
3150 REM ** IS OBJECT CREDIT CARD **
3160 IF F<>5THENPRINT"HE ONLY ACCEPTS CREDIT CARDS
":GOTO3040
3170 :
3180 REM ** IS CARD CARRIED **
3190 OV=5:GOSUB5830
3200 IFHF=0THENPRINT"YOU DO NOT HAVE THE ";IV$(5,1
):GOTO 3040
3210 :
3220 SN$="THE CLERK TAKES THE CARD AND SAYS 'THAT
WILL DO NICELY, SIR'"
3230 GOSUB5880:REM FORMAT PRINT
3240 SN$="YOU ARE ALLOWED TO PASS THE BARRIER AND
ENTER THE USER PORT"
3250 GOSUB5880:REM FORMAT PRINT
3260 :
3270 REM ** IS DIGITAYA CARRIED **
3280 OV=6:GOSUB5830
3290 IF HF=1 THEN 3380:REM SUCCESS
3300 :
3310 REM ** FAILURE **
3320 SN$="WELL DONE YOU HAVE SUCCEEDED IN ESCAPING
 FROM THE CLUTCHES"
3330 SN$=SN$+" OF THE MACHINE, BUT HAVE FAILED IN
YOUR MISSION"
3340 SN$=SN$+" TO BRING BACK THE MYSTERIOUS DIGITA
YA"
3350 GOSUB5880:REM FORMAT PRINT
3360 END
3370 :
3380 REM ** SUCCESS **
3390 SN$="CONGRATULATIONS, YOU HAVE SUCCEEDED IN Y
OUR MISSION"
3400 SN$=SN$+" TO RESCUE THE WONDEROUS DIGITAYA FR
OM THE"
3410 SN$=SN$+" CLUTCHES OF THE MACHINE.
3420 GOSUB5880:REM FORMAT PRINT
3430 END
3440 :
3450 REM **** CASSETTE PORT S/R ****
3460 SF=1
3470 SN$="YOU FEEL AN IRRESTISTIBLE FORCE PULLING
YOU TOWARDS"
3480 SN$=SN$+" PERMANANENT MAGNETIC SUSPENSION"
3490 GOSUB5880:REM FORMAT
3500 NS=0:REM START COUNTING INSTRUCTIONS
3510 REM ** INSTRUCTIONS **
3520 NS=NS+1:IFNS>3THEN3770:REM SUCKED OUT
3530 PRINT:INPUT"INSTRUCTIONS";IS$
3540 GOSUB1700:REM ANALYSE INSTRUCTIONS
3550 GOSUB1900:REM NORMAL ACTIONS
3560 IFMF=1THENMF=0:PRINT"YOU CAN'T MOVE...YET":GO
TO3510
3570 IFVF=1THEN3510:REM NEXT INSTRUCTION
3580 IFVB$<>"USE"THENPRINT"I DON'T UNDERSTAND":GOT
O3510
3590 REM ** INSTRUCTION IS USE **
3600 GOSUB5730:REM IS OBJECT VALID
3610 IFF=0THENPRINT"THERE IS NO ";NN$:GOTO3510
3620 :
3630 REM ** IS OBJECT BUFFER ACTIVATOR **
3640 IF F=8 THEN3680:REM OK
3650 SN$="YOUR "+IV$(F,1)+" IS USELESS, THE FORCE
INCREASES"
3660 GOSUB 5880:GOTO3510:REM NEXT INSTRUCTION
3670 :
3680 OV=8:GOSUB5830:REM IS BUFF ACT HELD
3690 IFHF=0THENSN$="YOU DON'T HAVE THE "+IV$(8,1):
GOSUB5880:GOTO3510
3700 :
3710 REM ** SAVED **
3720 SN$="YOU USE THE BUFFER ACTIVATOR TO COUNTER
THE PULL"
3730 SN$=SN$+" INTO MAGNETIC OBLIVION. THE FORCE S
UBSIDES"
3740 GOSUB5880:REM FORMAT
3750 RETURN
3760 :
3770 REM ** SUCKED OUT **
3780 SN$="THE FORCE BECOMES TOO STRONG AND YOU ARE
 PULLED OUT"
3790 SN$=SN$+" THROUGH THE CASSETTE PORT INTO MAGN
ETIC NOTHINGNESS."
3800 GOSUB 5880:REM FORMAT
3810 END
4180 REM **** TRI-STATE DEVICE S/R ****
4190 SF=1
4200 SN$="A LARGE SIGN SAYS 'I/O THIS WAY' BUT AS
YOU MOVE TOWARDS IT"
4210 SN$=SN$+" A TICKET COLLECTOR SHOUTS 'TICKETS
PLEASE'
4220 GOSUB5880:REM FORMAT
4230 :
4240 REM ** INSTRUCTIONS **
4250 PRINT:INPUT"INSTRUCTIONS";IS$
4260 GOSUB1700:GOSUB1900:REM ANALYSE
4270 IFMF=1 THEN RETURN
4280 IFVF=1THEN4240:REM NEXT INSTRUCTION
4290 IFVB$<>"GIVE"ANDVB$<>"OFFER"THENPRINT"I DON'T
 UNDERSTAND":GOTO4240
4300 REM ** INSTRUCTION IS GIVE **
4310 GOSUB5730:REM IS OBJECT VALID
4320 IFF=0THENPRINT"THERE IS NO ";NN$:GOTO4240:REM
 NEXT INSTRUCTION
4330 :
4340 REM ** IS OBJECT TICKET **
4350 IF F=4 THEN4400:REM OK
4360 SN$="THE TICKET COLLECTOR SHAKES HIS HEAD AND
 SAYS"
4370 SN$=SN$+" 'I CANNOT ACCEPT THIS "+IV$(F,1)
4380 GOSUB5880:GOTO4240:REM NEXT INSTRUCTION
4390 :
4400 OV=4:GOSUB5830:REM IS TICKET HELD
4410 IFHF=0THENPRINT"YOU DO NOT HAVE THE TICKET":G
OTO4240
4420 :
4430 REM ** OK **
4440 SN$="THE TICKET COLLECTOR ACCEPTS YOUR TICKET
 AND ALLOWS YOU"
4450 SN$=SN$+" TO PASS THE BARRIER."
4460 GOSUB5880:REM FORMAT
4470 REM ** DEL TICKET FROM LIST **
4480 F=0
4490 FORJ=1TO4
4500 IF IC$(J)=IV$(4,1)THENIC$(J)="":J=4
4510 NEXT J
4520 IV$(4,2)=STR$(INT(RND(TI)*40+8)):REM REALLOCA
TE TICKET POSITION
4530 P=15:MF=1:RETURN
```

Leihbücherei

Die Entwicklung zeit- und arbeitsparender Techniken kann für Programm-Projekte sehr nützlich sein. Einen Weg dazu wollen wir heute darstellen: den Aufbau einer Bibliothek, deren Einzelroutinen sich in die unterschiedlichsten Programme einbauen lassen.

Vielleicht erscheint Ihnen das strukturierte Programmieren sehr mühsam und umständlich – Sie können damit aber beim Programmieren selbst und bei der Fehlersuche sehr viel Zeit sparen. Direkt an der Tastatur entwickelte Programme haben häufig unnötig komplizierte Strukturen und Algorithmen. Solche Programme lassen sich meist nur unvollständig testen und sind zudem wegen ihrer Unübersichtlichkeit schwer zu entstören. Frühzeitige Planung vereinfacht dagegen die Programmstruktur und vermeidet dadurch Fehler.

Wer sorgfältig plant, gerät kaum in die Sackgasse eines unzureichenden Programm- oder Fileaufbaus. Diese Sackgasse wird erst später sichtbar – etwa wenn nicht genügend Platz für ein Feld innerhalb eines Files gelassen wurde. Solche Probleme führen oft dazu, daß große Teile eines nahezu fertigen Programms völlig neu geschrieben werden müssen.

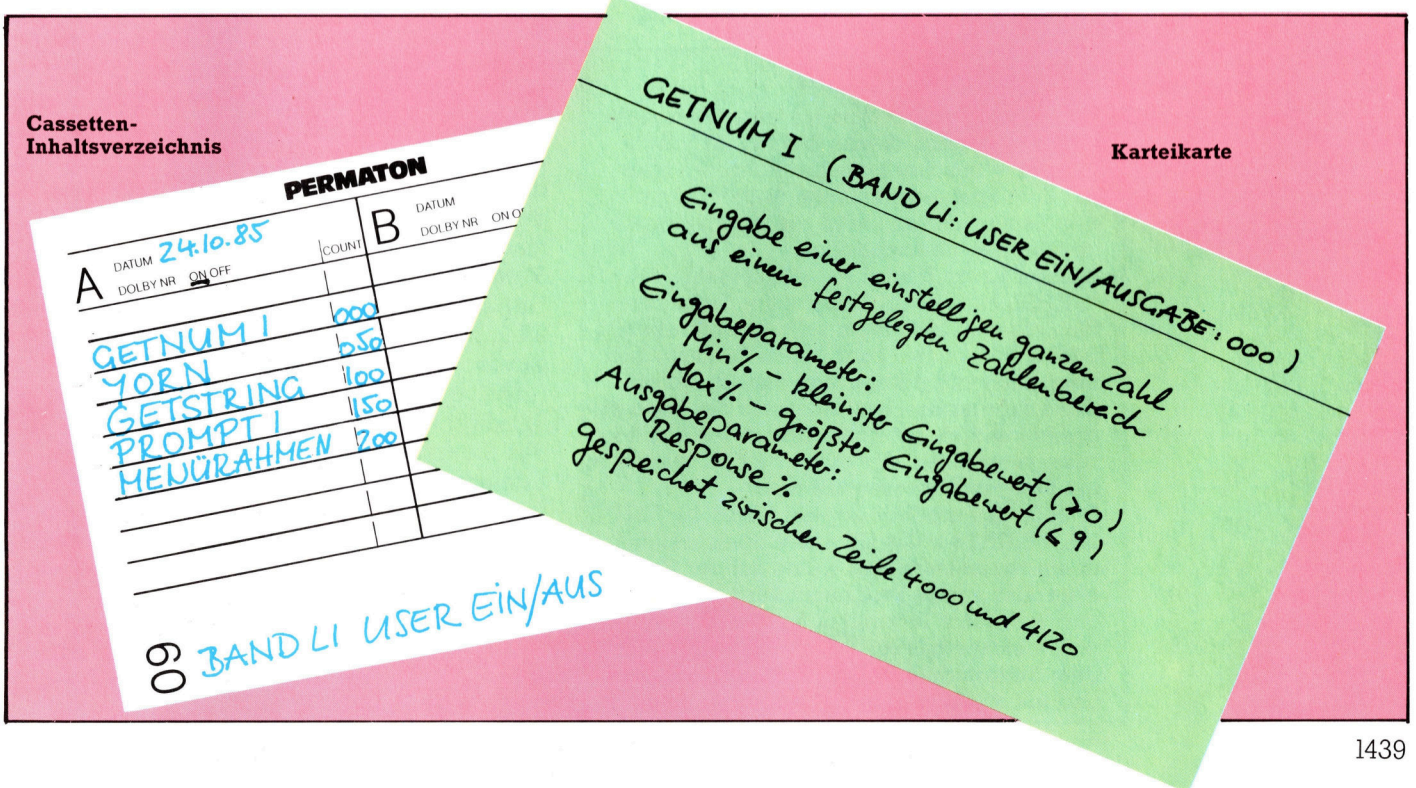
Wer einen Rechner mit einer richtigen Schreibmaschinentastatur hat, kann zwar das Zehnfingersystem lernen – die Programmeingabe läßt sich damit jedoch nur unwesentlich beschleunigen. Aber es gibt andere Wege, schneller zu programmieren. Der erste Schritt dazu ist, beim Programmieren bestimmte Regeln einzuhalten: die Verwendung bestimmter Namenstypen für die Bezeichnung lokaler Variablen, die sie von Variablen des Hauptprogramms unterscheidbar machen, der Beginn jedes Unterprogramms mit Zeilennummern, die auf 000 enden und der Abschluß aller Unterprogramme mit RETURN in einer eigenen Zeile. Außerdem sollte jeder besondere Unterprogramm-Typ in festgelegten Zeilenblöcken liegen (Unterprogramme für das Filehandling etwa zwischen 9000 und 9999, Utilities von 50000 aufwärts und so weiter).

Komfortprogramme

Das Einhalten dieser Regeln bringt eine Menge Vorteile: Kein Suchen mehr nach der Menü-Routine – sie steht immer wieder am selben Platz. Nie wieder die bange Frage, ob etwa der gleiche Variablenname im Haupt- und im Unterprogramm vorkommt – schon der Name macht Lokalvariablen sofort identifizierbar.

Die genannten Techniken sind auch beim Erstellen Ihrer Programmbibliothek sinnvoll. Bei guter Organisation können Sie damit Programme in der Hälfte der üblichen Zeit fertigstellen. Am besten bauen Sie Ihre Bibliothek

Ohne übersichtliche und einheitliche Dokumentation sind Programmbibliotheken unbrauchbar. Das trifft besonders für Anwender zu, die mit dem Cassettenrecorder arbeiten: Es ist eine sehr undankbare Aufgabe, eine Cassette ohne Inhaltsverzeichnis mit LOAD und LIST zu durchsuchen!



aus Teilen bereits vorhandener Programme auf, die wichtige Abläufe behandeln (Ein/Ausgabeprogramme, Datums-Unterprogramme, Umwandlung von Groß- in Kleinbuchstaben etc.). Jede Routine wird als einzelnes File gespeichert, wobei man die Programmteile ihrer Funktion entsprechend zu Gruppen zusammenfassen und gemeinsam auf einer Cassette unterbringen kann. In einer ausführlichen Kartei bzw. Datenbank sollten Variablennamen, Eingabeparameter und Funktionsweise der einzelnen Unterprogramme dokumentiert werden.

Jedes Programm der Bibliothek muß selbstverständlich gründlich getestet sein. Weil die Routinen innerhalb „fremder" Programme laufen, müssen sie unzulässige Dateneingaben sicher erkennen. Auch die ausgegebenen Daten dürfen im Gesamtprogramm keine Probleme erzeugen. Die Unterprogramme müssen ausreichend interne Dokumentationen enthalten, damit Sie ihre Funktion auch später noch verstehen. Ihre Bibliothek sollten Sie nur bei Bedarf ausbauen – Programmteile „auf Verdacht" zu erstellen, lohnt erfahrungsgemäß nicht. Die Zeilennummern Ihrer Bibliotheksprogramme müssen immer der von Ihnen gewählten Festlegung entsprechen – das erspart beim „Zusammenbau" der Programmteile ein zeitraubendes RENUMbern. Sie müssen natürlich nicht jedes Programm selbst schreiben. Computerzeitschriften drucken häufig neben kompletten Programmen, aus denen selbstverständlich auch nur Teile verwendet werden können, auch Listings nützlicher Unterprogramme ab.

Um die Bibliothek zu nutzen, ist eine Methode zum Einbau der Routinen in ein Gesamtprogramm nötig. Wer mit komplizierten Sprachen arbeitet, besitzt meist auch den dazugehörigen „link-loader", der einzelne kompilierte Module zu einem kompletten Programm vereinigt. Sofern BASIC-Programmierer ohne Compiler arbeiten, können sie für den gleichen Zweck eine Kombination aus RENUMber- und MERGE-Befehl einsetzen. Um eine Bibliotheksroutine mit einem anderen Programm zu koppeln, wird zuerst das Hauptprogramm geladen und der Platz für das Unterprogramm festgelegt. (Ein ausreichend großer Block ungenutzter Zeilennummern muß vorhanden sein.) Falls es notwendig ist, das Bibliotheksprogramm dann mit RENUMber so verändern, daß es in den freien Block hineinpaßt. Mit MERGE werden die beiden Programme zusammengefaßt. Wenn alles nach Wunsch funktioniert, kann das neue Programm geSAVEd werden.

Oft arbeiten Computerfreunde beim Erstellen neuer Programme in einer Schulgruppe oder im User-Club zusammen. Gerade für die Teamarbeit sind Programmplanung und effizientes Programmieren besonders wichtig. Die Grundlagen der strukturierten Programmierung wurden bei dem Versuch gelegt, die Entwicklung kommerzieller Programme auf mehrere Mitarbeiter zu verteilen. Jeder Programmierer bearbeitete nur einen Teilbereich, das komplette Programm setzte sich später aus diesen Komponenten zusammen.

Bindende Regeln

Nur mit bindenden Regeln kann man auch BASIC-Programme auf diese Weise erstellen. Steht die allgemeine Programmstruktur fest, muß der Programmierer eines einzelnen Moduls folgendes wissen:

1) welche Files vorhanden sind und wie sie organisiert sein sollen,
2) welche Vereinbarungen bezüglich der Variablenbenennung getroffen wurden. (Die Namen der wichtigsten Variablen, etwa im gesamten Programm genutzter Felder, werden vorab bestimmt. Auch die Bezeichnung lokaler Variablen sollte festgelegt sein. Variablen, die in mehreren Programm-Modulen verwendet werden, sollten entweder vorher benannt oder etwa durch Anhängen der Nummer des erzeugenden Moduls als Index deutlich erkennbar gemacht werden.),
3) welche Bibliotheks-Unterprogramme der Gruppe zugänglich sind (Formate, Variablennamen, spezielle Funkionen und Fehlerfreiheit),
4) wie die Fehlerbehandlung organisiert ist (etwa, ob jedes Teilprogramm Fehler eigenständig bearbeitet oder durch ein „Error-Flag" das Steuerprogramm aufruft),
5) die genauen Funktionen aller geplanten Programmteile,
6) Bereich und Typ aller Daten, die die einzelnen Programm-Module als Eingabe verlangen bzw. ausgeben.

Das Arbeiten nach dieser Methode verkürzt die Programmierzeit zugunsten der gemeinsamen Strategieplanung. Die darauffolgende Testphase werden wir noch behandeln.

Der MERGE-Befehl

- Der Sinclair Spectrum hat eine besonders einfache Version dieses Befehls: Er mischt (merge = mischen) die bezeichneten Files mit dem Speicherinhalt. Bei übereinstimmenden Zeilennummern wird die Programmzeile im Speicher überschrieben.
- Beim Acorn B wird mit dem Befehl *SPOOL eine ASCII-Version der Programmfiles erzeugt. Mit einem selbstgeschriebenen BASIC-Programm (oder einem Textverarbeitungsprogramm) kann man danach auf einzelne Programmzeilen innerhalb der Files zugreifen und sie zu einem dritten ASCII-File zusammensetzen, das durch *EXEC wieder in ein Programm verwandelt wird.
- Beim Commodore kopiert OPEN 1,1: CMD1: LIST: PRINT#1: CLOSE 1 den Speicherinhalt in ein namenloses ASCII-File auf der Cassette. Das zweite Programm wird geladen und mit einem Programm für INPUT gekoppelt, welches das ASCII-File zeilenweise auf dem Bildschirm ausgibt. Programm anhalten und mit RETURN die Programme vereinigen.

Allzweck-gerät

Die ersten Schneider-Computer ließen sich wegen ihres kleinen Arbeitsspeichers nur bedingt einsetzen. Der neue CPC 6128 hat einen größeren Speicher und ist somit auch für den kommerziellen Einsatz interessant.

Der CPC 6128 baut auf dem Erfolg des CPC 464 und des 664 auf. Er ist mit seinen Vorgängern softwarekompatibel, verfügt aber über einen erweiterten Arbeitsspeicher mit 128 KByte. Unter dem verbesserten CP/M des 6128 laufen die klassischen kommerziellen CP/M-Pakete.

Schon ein Jahr nach der Vorstellung des CPC 464 hatte der Schneider in Großbritannien 25 Prozent des Heimcomputermarkts erobert. Die Cassettenmaschine war nach dem Erfolgskonzept der Hi-Fi-Produkte von Amstrad gebaut worden: Alle Systemkomponenten sind in einem Gehäuse untergebracht, alle vom Anwender gewünschten Einsatzmöglichkeiten sind vorhanden, und der Preis stimmt.

Der CPC 664 ist eine erweiterte Version des CPC 464. Er besitzt eine integrierte Diskettenstation und ein geringfügig erweitertes BASIC. Der CPC 664 und das als Erweiterung des 464 gedachte Diskettenlaufwerk DDI–1 arbeiten mit CP/M und verfügen über eine Version des Dr. LOGO von Digital Research.

Nur wenige Monate später stellte Schneider den CPC 6128 vor, der dem 664 fast völlig gleicht, aber mit 128 KByte RAM ausgerüstet ist. Der 6128 hat das Aussehen einer kommerziellen Maschine. Die Tastatur ist in einheitlichem Grau gehalten. Der 464 und der 664 verfügen über separate Tastenblocks für Cursorsteuerung und Zahleneingaben. Auf dem 6128 wurden diese Tasten in ein zusammenhängendes Tastenfeld integriert. Dadurch ist das Gerät etwa sechs Zentimeter schmaler als der 664. Die Bedienung wurde durch die Verringerung der Gehäusehöhe und die Verkürzung des Tastenweges leichter und komfortabler.

Wie beim 664 liegen die Schnittstelle für eine weitere Diskettenstation und die Centronics-Druckerausgänge auf der Rückseite der Maschine. Die Anschlüsse für Cassettenrecorder, Joystick und Lautsprecher wurden auf die linke Seite des Geräts verlegt, während der Netzschalter und der Lautstärkeregler sich statt auf der rechten Seite nun auf der Rückseite befinden. Die Diskettenstation hat das gleiche 3-Zoll-Format (einseitig) wie beim CPC 664, ist jedoch weitaus flacher. Auf dem Laufwerk befindet sich eine Tabelle, aus der sich die Schlüsselzahlen für die Bildschirmfarben ablesen lassen.

Zum Lieferumfang des CPC 6128 gehören zwei Disketten. Die eine enthält CP/M Plus – eine verbesserte Version von CP/M – und eine Reihe von Programmierhilfen, darunter ein Assembler, ein Disassembler, Diskettenmodule und CP/M-Hilfsprogramme wie PIP und SUBMIT. Auf der zweiten Diskette befindet sich eine 48-KByte-Version von Dr. LOGO, die mit mehr als 50 zusätzlichen Befehlen eine wesentliche Verbesserung gegenüber der Version des CPC 664 darstellt.

Doppelter Speicher

Die zweite Diskette enthält außerdem das GSX-Programm von Digital Research und ein HELP-Modul. GSX war der Vorläufer des GEM von Digital Research. Unter GSX geschriebene Grafikroutinen laufen auf jeder Maschine der Z80- oder 8086-Familie. Das komplette Paket fand bei Softwarehäusern jedoch nie große Verbreitung.

Obwohl Acht-Bit-Prozessoren eigentlich nur 64 KByte adressieren können, verdoppelt der mit einem Z80A ausgerüstete 6128 diesen Speicherraum. Dabei werden durch „Bank Switching" verschiedene Bereiche von ROM und RAM in den Adreßraum des Prozessors (64 KByte) hinein- und herausgeschaltet.

CP/M Plus

Die neue CP/M-Version des CPC 6128 hat wesentliche Vorteile gegenüber der CP/M-2.2-Version für den CPC 664. Auf dem 6128 ist die wichtigste Eigenschaft von CP/M Plus die Freisetzung von 61,5 KByte RAM für Anwendungsprogramme. Damit stehen drei bis vier KByte mehr zur Verfügung, als die meisten Standard-CP/M-Programme benötigen, und auf dem 6128 kann die Palette der kommerziellen CP/M-Software eingesetzt werden, darunter SuperCalc, dBase II und WordStar.

Bei vielen CP/M-Paketen, die für Doppellaufwerke oder doppelseitige Disketten geschrieben sind, kann der Aufruf eines nicht vorhandenen Laufwerks einen Programmabbruch auslösen. CP/M Plus fängt dieses Problem ab: Es erscheint die Aufforderung, entweder die entsprechende Diskette einzusetzen oder den Befehl zu stornieren.

CP/M Plus kann außerdem die Zeichensätze anderer Sprachen verwenden oder wie ein VT52-Terminal arbeiten.

Platz zum Arbeiten

Die im Lieferumfang des 6128 enthaltene Version von Dr. LOGO besitzt viele Befehle, die in der Version des CPC 664 fehlen. Durch zusätzliche Listenverarbeitung, erweiterte mathematische Möglichkeiten, Grafik, Editierung und Diskettenbefehle enthält das Gerät eine weitaus flexiblere Version dieser wichtigen Programmiersprache. Der Arbeitsbereich wurde von den 2105 Datenelementen des CPC 664 auf 3753 Elemente erweitert.

Praktisch sind die Zugriffsmöglichkeiten auf den Drucker und E/A-Schnittstellen, mit denen sich auch die LOGO-Schildkröte steuern läßt. Durch verbesserte Diskettenbefehle lassen sich Dateien umbenennen oder von LOGO aus andere Laufwerke ansprechen. Mit den erweiterten Befehlen für Editierung und Fehlersuche kann man Dateien direkt von der Diskette in den Bildschirmeditor von LOGO laden und globale Variablen und Prozeduren im Arbeitsspeicher auflisten.

BASIC-ROM
Dieser Chip enthält die BASIC-Version 1.1.

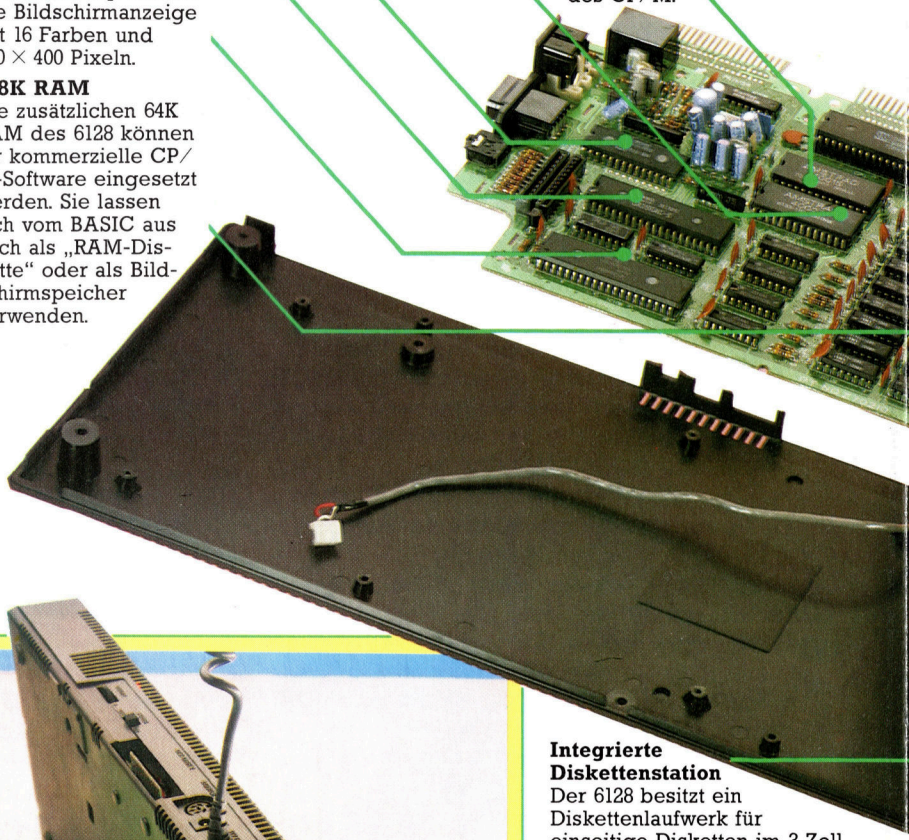
8912-Sound-Chip
Wie der CPC 464 und 664 enthält der 6128 einen Chip, der drei Stimmen erzeugen und die Hüllkurven für Ton und Lautstärke steuern kann.

PIO-Chip
Dieser Chip steuert die Schnittstelle für den Centronics-Drucker.

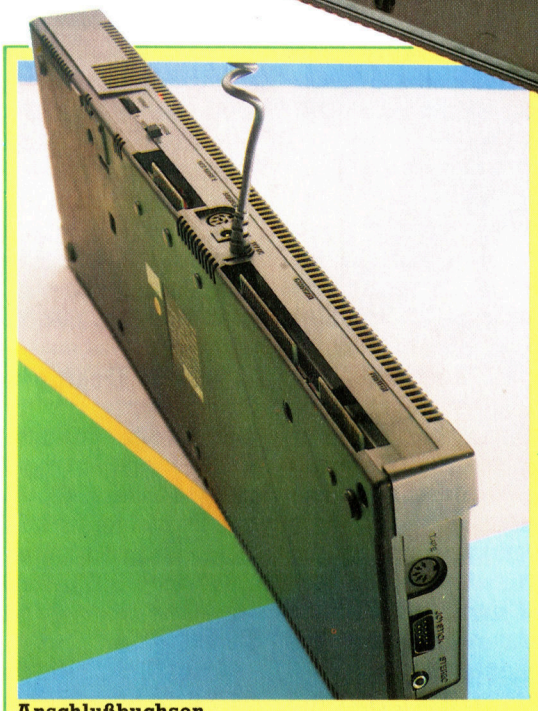
Video-Steuerung
Der Videochip steuert die Bildschirmanzeige mit 16 Farben und 640 × 400 Pixeln.

128K RAM
Die zusätzlichen 64K RAM des 6128 können für kommerzielle CP/M-Software eingesetzt werden. Sie lassen sich vom BASIC aus auch als „RAM-Diskette" oder als Bildschirmspeicher verwenden.

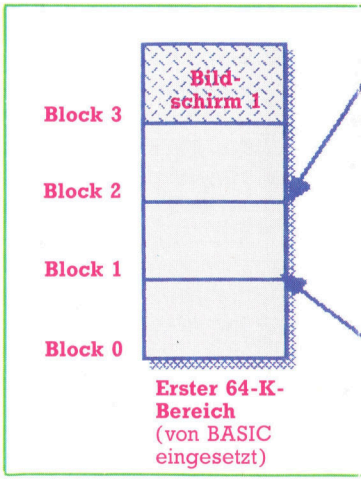
OS-Chip
Dieses ROM beinhaltet das Betriebssystem und Teile des CP/M.



Integrierte Diskettenstation
Der 6128 besitzt ein Diskettenlaufwerk für einseitige Disketten im 3-Zoll-Format, die jeweils 170 K speichern können.



Anschlußbuchsen
Der 6128 besitzt Ausgänge für einen Centronics-Drucker, eine Erweiterungsschnittstelle, ein zweites Diskettenlaufwerk, Joysticks, externe Lautsprecher und einen Cassettenrecorder sowie Monitorbuchsen.



Die Hilfsroutine BANKMAN stellt dem BASIC zusätzliche Befehle zur Verfügung, die den Umgang mit dem größeren Speicher möglich machen. Die 128 KByte des RAM sind in zwei Sektionen zu 64 KByte aufgeteilt. Da BASIC-Programme jedoch nur einen dieser Bereiche belegen können, lassen sich auch mit dem 6128 keine umfangreicheren Programme schreiben als mit dem CPC 464 oder 664. Die zweiten, nicht für BASIC-Programme einsetzbaren 64 KByte lassen sich jedoch vom BASIC aus mit den neuen Befehlen als zusätzliche Bildschirm- oder Datenspeicher einsetzen.

Da die Bildschirmanzeige 16 KByte belegt, können die zweiten 64 KByte vier zusätzliche Bildschirminhalte mit den Nummern 2 bis 5 speichern. Der Standardbildschirmspeicher

liegt jedoch in den 64 KByte, die auch das BASIC verwendet. Der Befehl SCREENCOPY,A,B kopiert den Bildschirm B in die 16 KByte von A und überschreibt dessen Inhalt, während SCREENSWAP,A,B den Inhalt der beiden Bildschirmbereiche gegeneinander austauscht. Ein Bildschirmaustausch kann bis zu einer halben Sekunde dauern. Es ist jedoch möglich, nur jeweils ein Vierundsechszigstel des Bildschirms zu kopieren oder auszutauschen. Für die Spieleprogrammierung bietet sich hier die Möglichkeit, kleine Bereiche des Bildschirms schnell verändern zu können.

RAM-Diskette

Die zusätzlichen 64 KByte lassen sich jedoch auch für eine Datei einsetzen. Dieser Bereich dient dabei als zusätzlicher Speicher, in dem Datensätze mit BASIC-Strings untergebracht werden. Da er sich wie eine Diskettendatei ansprechen läßt, wird er als „RAM-Diskette" bezeichnet. Die Datensätze dieser Datei müssen alle gleich lang sein. Sie werden mit dem Befehl „IBANKOPEN, n" eingerichtet, wobei n die Länge eines Datensatzes anzeigt und zwischen zwei und 255 liegen kann. Mit IBANKREAD und IBANKWRITE werden Stringdaten an die RAM-Disk übergeben oder von dort gelesen. Die Datensätze lassen sich aber auch einzeln ansprechen. Ist kein bestimmter Datensatz angegeben, so spricht das BASIC den zuletzt aufgerufenen (oder den ersten) Datensatz an und setzt seinen Zeiger danach automatisch auf den nächsten. Der Bereich läßt sich damit sowohl für sequentielle wie auch für wahlfreie Dateien verwenden.

Der CPC 6128 ist eine gut aufgebaute, kompakte Maschine, die viel für ihren Preis bietet.

Schneider CPC 6128

SPEICHERKAPAZITÄT

128 K RAM, 48 K ROM. Für BASIC-Programme stehen nur 64 K zur Verfügung, die zweiten 64 K lassen sich jedoch als „RAM-Diskette" oder für vier weitere Bildschirmspeicher einsetzen.

ZENTRALEINHEIT

Z80A mit 4 MHz.

DISKETTENFORMAT

Ein einseitiges 3-Zoll-Laufwerk, Anschlußbuchse für ein zweites Laufwerk.

SCHNITTSTELLEN

Erweiterungsbus, Schnittstellen für ein zweites Diskettenlaufwerk, Cassettenrecorder, Stereoausgang, Joysticks, Centronics-Drucker, Eingänge für 12V und 5V, Monitorbuchse.

MITGELIEFERTE SOFTWARE

Zwei Disketten enthalten CP/M Plus, Dienstmodule für die Programmierung, das erweiterte Dr. LOGO und GSX. Das CP/M 2.2 sorgt für die Kompatibilität mit dem 664 und dem 464.

DOKUMENTATION

Das Anwenderhandbuch behandelt die BASIC-Programmierung und Speicherverwaltung des erweiterten Systems. Die Kapitel über Dr. LOGO und CP/M enthalten die grundlegenden Abläufe.

STÄRKEN

Der CPC 6128 geht an die Grenzen der Acht-Bit-Technik und bietet viel für seinen Preis. Durch seine Fähigkeit, CP/M-Standardprogramme verarbeiten zu können, läßt sich die Maschine für kommerzielle Zwecke und für Spiele einsetzen.

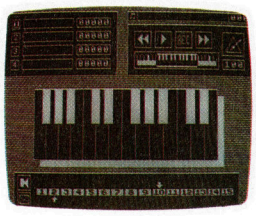
SCHWÄCHEN

Schneider verwendet 3-Zoll-Disketten statt des weitverbreiteten 3 1/2-Zoll-Formats, das von vielen kleinen kommerziellen Maschinen eingesetzt wird. Da die Laufwerke nur einseitige Disketten verarbeiten können und einige Programme beide Diskettenseiten ansprechen müssen, wird die Bedienung etwas umständlich.

Geänderte Adressen

Das BASIC des 6128 kann nicht alle 128 K RAM direkt ansprechen, sondern nur mit Programmen arbeiten, die in den ersten 64 K liegen. Die zweiten 64 K lassen sich jedoch als „RAM-Diskette" oder für vier zusätzliche Bildschirmspeicher einsetzen. Da der Z80A-Prozessor nur 64 K adressieren kann, wurde der zusätzliche Speicher in Blöcke zu je 16 K unterteilt.

Musik liegt in der Luft



Zu den herausragendsten Merkmalen des Music Systems gehört die Bildschirmdarstellung mit Piktogrammen und klar gestalteten Menüs. Die Aufnahme zeigt die „Keyboard Utility" auf dem Schirm.

Bei den meisten der bisher angebotenen Musikpakete muß der Benutzer komplexe Programme implementieren, um auch nur eine einzige Note zu erzeugen. „The Music System" löst dieses Problem und liefert ein professionelles Programm für Acorn B und C 64.

The Musik System von Island Logic, als Disketten- oder Cassettenversion erhältlich, umfaßt eine Folge von fünf Optionen. Das Laden von Cassette ist umständlich, da es zwei Bänder sind. Doch dank der Windows, Piktogramme und der Grafik im Macintosh-Stil ist das System anwenderfreundlich.

Das erste und meist benutzte der fünf Programme ist der Editor, den der Hersteller als ein „Notenverarbeitungssystem für Musiker" beschreibt. Es macht das Schreiben oder Verändern von Musik ungewöhnlich einfach. Auf dem Bildschirm sind Notenlinien für Oberstimme und Baßstimme dargestellt, auf die Noten geschrieben werden. Verändert man die Notenwerte, so werden die Taktstriche entsprechend dem vorgewählten Tempo automatisch eingesetzt. Hat man eine Melodie komponiert oder möchte man vorhergehende Takte hören, drückt man die TAB-Taste, wodurch das Audio-Playback ausgelöst wird und die Noten auf den Schirm „gerollt" werden.

Ein großer Vorteil dieses Systems ist, daß es sowohl für den Anfänger als auch für den erfahrenen Musiker eine große Hilfe darstellt. Der Anfänger sieht und hört genau, was sich in jedem Stadium des Komponierens ereignet, so daß Fehler leicht gefunden und rasch und genau korrigiert werden können.

Mit der Song- und Sound-Bibliothek steht eine Fülle von Grundrhythmen zur Verfügung. Ferner gibt es bekannte Melodien, die gespielt und verändert werden können, darunter Mozarts Klaviersonate in Es-Dur und den Hummelflug. Das Tempo läßt sich von 30 Takten pro Minute bis auf 200 pro Minute steigern. Damit sind beim Modifizieren einzelner Musikstücke ungeheuer aufregende Resultate möglich.

Die in Verbindung mit dem Editor und der Tastatur-Option arbeitende Synthesizer-Funktion wird zur Gestaltung beliebiger Klänge verwendet. Jeder Klang läßt sich durch Setzen der Parameter einer Hüllkurve definieren. Die Diskettenversion verfügt zudem über 15 verschiedene Percussion-Effekte. Wie bei den anderen Modulen ist das Editieren sehr einfach. Für jeden Parameter steht ein separates Piktogramm auf dem Bildschirm, und die Klänge hört man schon bei der Eingabe. Grafiken der Frequenzen und Amplituden lassen sich leicht darstellen. Synthesizer-Klänge werden in zwei Files gespeichert und können dann in den Editor oder das Keyboard geladen werden.

In der Keyboard-Option stehen die beiden mittleren Oktaven einer Klaviertastatur auf dem Bildschirm zur Verfügung, wobei die QWERTY-Tastenreihe des Rechners die Funktion der weißen, die Zahlentasten die der schwarzen Tasten haben. Unterhalb des Keyboards befindet sich ein Fenster, das die Piktogramme für Hüllkurve und Lautstärke enthält.

Die zwei ausführlichen Handbücher sollte man sorgfältig durcharbeiten, um das Programm optimal nutzen zu können.

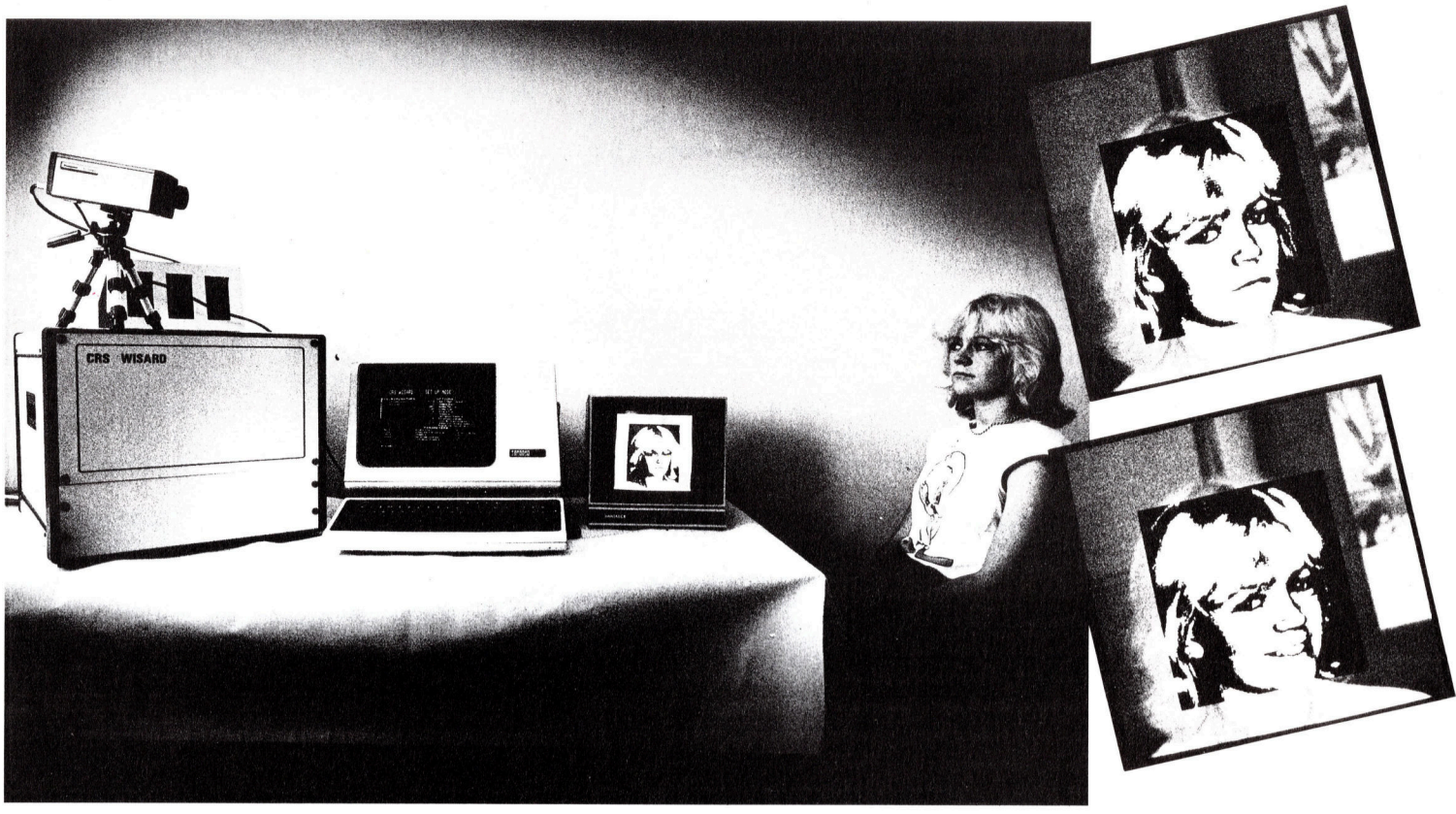
Das Music System bietet sowohl dem Anfänger als auch dem erfahrenen Musiker totale Kontrolle über die Klangmöglichkeiten des Acorn B oder des C 64. Das Programm auf Cassette wird in zwei Paketen geliefert: Das erste enthält Synthesizer- und Keyboard-Module, im zweiten sind Editor- und Ausdruck-Programm enthalten.

The Music System: Für Acorn B und C 64. Cassette Pack 1 – Synthesizer, Keyboard, Song & Sound Library, Cassette Pack 2 – Editor, Printout, Song & Sound Library

Hersteller: Island Logic Ltd., 22 St Peter's Square, London W6 9NW

Programm: Cassette und Diskette

Joysticks: Nicht erforderlich



Musterhaft

Wir beschäftigen uns weiterhin mit der Mustererkennung und entwickeln ein BASIC-Programm für diese Aufgabe.

Das am Imperial College, London, von Igor Aleksander entwickelte WISARD-System verwendet ein 512 x 512-Muster und ein RAM von einem Megabyte zur Erkennung von Mustern, die auf einem Fernsehschirm dargestellt werden. Es ist so empfindlich, daß es zwischen lächelnden und ernsten Gesichtern unterscheiden kann.

Sogenannte „bottom-up"-Mustererkennungs-Systeme versuchen, aus einem gegebenen Bild herausragende Merkmale zu isolieren und diese dann in ein einfacheres, abstrakteres Muster umzuwandeln. Eine Methode besteht darin, zum Abtasten lokale Operatoren zu verwenden. Jeder lokale Operator tastet einen kleinen Bereich ab und multipliziert die Graudichte der Punkte nach Gewichtungsfaktoren. Diese Faktoren sind so angelegt, daß sie hohe Zahlen erzeugen, sobald das gesuchte Merkmal lokalisiert wurde. Durch Verwendung einer Vielzahl unterschiedlicher lokaler Operatoren können die Positionen wichtiger Merkmale herausgelesen werden.

Das WISARD-Programm, das in der vorhergegangenen Folge beschrieben wurde, bedient sich des Prinzips der „Speicheradressen-Generierung", um sich ein Muster zu merken und es später wiederzuerkennen.

Der Vorteil von WISARD ist, daß alle RAM-Bänke parallel adressiert sind. So kann es mit hoher Geschwindigkeit scharfe Fernsehbilder verarbeiten. Unser Beispielprogramm in BASIC simuliert ein solches System sequentiell. Es ist erheblich langsamer, demonstriert aber die dabei berücksichtigten Techniken.

In diesem System werden „Versechsfacher" (Sextuple) als Diskriminatoren verwendet (die zwar weniger leistungsfähig als Verachtfacher sind, dafür aber weniger Speicherplatz benötigen). Dafür sind 20 480 (40 x 64 x 8) RAM-Bit oder 2560 Byte erforderlich. Der Grund ist: Wir haben 40 Versechsfacher, die sich in 64 verschiedenen Stadien befinden können und somit in der Lage sind, eine RAM-Bank von 64 Plätzen zu adressieren. Jeder Platz enthält acht Bits, womit es dem System möglich ist, zwischen acht Eingabe-Gruppen zu „diskriminieren" (zu unterscheiden).

Das Programm besteht aus zwei Phasen: Im Trainingsstadium werden dem Computer Beispiele für unterschiedliche Eingabegruppen beigebracht. In der zweiten Phase werden Muster erkannt, indem die Bilder mit den gespeicherten Beispielen verglichen werden.

Nehmen wir als Beispiel an, daß während der Trainingsphase der Versechsfacher Nummer 20 die Zahl 1100 10 (50) ausgibt, wenn ein Muster der Klasse 4 vorliegt. In diesem Fall wird das vierte Bit an Platz 50 von RAM-Bank 20 auf 1 gesetzt. Reagiert während der Wiedererkennungsphase der Diskriminator wieder mit einer 50, stellt er fest, daß das vierte Bit in die

betreffende Adresse gesetzt wurde. Daraufhin ist das Vorhandensein eines Musters der Nummer 4 anzunehmen.

Eine Reihe von Bytes, bei D beginnend, wird für die Speicherung der Muster-Daten verwendet. Das komplette Array wird vor jedem Durchlauf „gesäubert". Das erfolgt durch die Subroutine 1000. Die Subroutine 2000 ermöglicht Ihnen, 16 mal 16 Bit große Muster auf den Bildschirm zu zeichnen. Dabei werden die Tasten U, D, L und R für die Cursorsteuerung benutzt, und durch * und Leertaste wird das Muster erzeugt.

Das Programm kann derart modifiziert werden, daß das Speichern des Array D auf Diskette oder Cassette möglich wird. Sie könnten das System auch so abändern, daß es mit der „Außenwelt" durch einen A/D-Wandler verbunden ist (beispielsweise mittels Kamera).

Die Anordnung der Punkte zu Sechsergruppen ist zwar zufallsgesteuert, muß aber wiederholbar sein.

Das Musterbild wird in einem zweidimensionalen Array,I(,) festgehalten. Dabei stellt 1 einen Stern, 0 einen Leerraum dar. In der Trainingsphase werden Adressen durch zufällig ausgewählte I-(,)-Elemente generiert. Das entsprechende Bit wird in Zeile 210 gesetzt. In der Erkennungsphase muß dieselbe Sequenz zufällig gewählter Adressen generiert werden, um das zu testende Muster mit den zuvor erlernten Klassifikationen zu vergleichen.

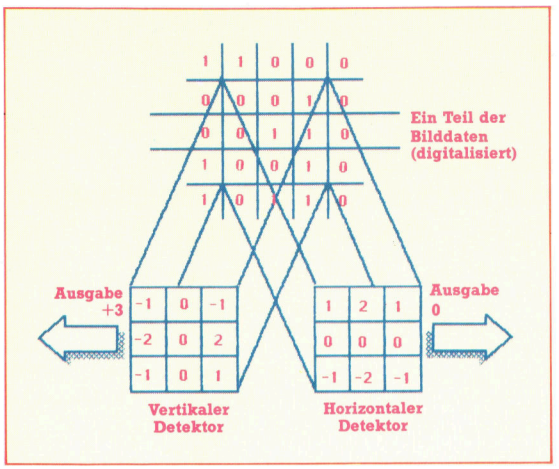
Mustererkennung

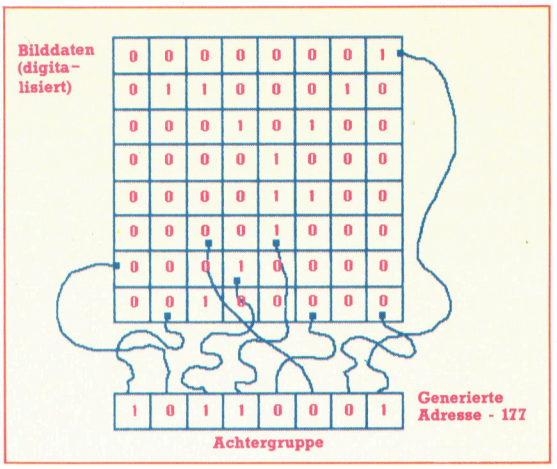
Acorn B

```
  10 REM ************************
  30 REM ** BBC WISARD-TYPE     **
  40 REM ** PATTERN RECOGNIZER **
  50 REM ************************
  55 MODE 7
  56 SX%=40: DS%=6: RB%=2^DS%
  57 MEMO%=SX%*RB%
  60 DIM I%(16,16), CS%(7)
  64 DIM D% MEMO%
  65 @%=4 : REM o/p format
  66 REM D% is a byte-array (BBC)
  70 REM -- I% is image array, D% is address array.
  80 REM -- Note: I% contains either 0 or 1
  85 PRINT "Total RAM size = ";MEMO%;" bytes."
  88 GOSUB 1000: REM clear D% array.
  90 REM -- First the Training Phase:
  92 MOOD$="Training"
  95 INPUT "Which class is it (0..7) ",C%
  96 IF C%<0 OR C%>7 THEN GOTO 90
 100 FOR I%=1 TO 16: FOR J%=1 TO 16: I%(I%,J%)=0
 101 NEXT: NEXT
 110 REM -- Now get user-painted image:
 120 GOSUB 2000
 130 R=RND(-1): REM random seed value
 140 FOR I%=1 TO SX%
 150 A%=RB% * (I%-1)
 155 REM A% is base address of RAM-bank.
 160 FOR J%=0 TO (DS%-1)
 170 R1%=INT(RND(1)*16 + 1)
 180 R2%=INT(RND(1)*16 + 1)
 190 A%=A%+ I%(R1%,R2%) * 2^J%
 200 NEXT J%
 202 REM R1% and R2% are 'random' co-ords.
 210 PROCdset(A%,C%)
 220 NEXT I%
 230 INPUT "Another training session (Y/N) ", A$:
IF A$="Y" THEN GOTO 90
 233 IF A$="y" THEN GOTO 90
 235 MOOD$="Recognition"
 236 REM -- Now the recognition phase:
 240 FOR I%=1 TO 16: FOR J%=1 TO 16
 244 I%(I%,J%)=0
 245 NEXT: NEXT
 250 REM -- Get an image for classification:
 260 GOSUB 2000
 270 R=RND(-1)
 280 FOR C%=0 TO 7: CS%(C%)=0: NEXT
 290 FOR I%=1 TO SX%
 300 A%=RB% * (I%-1)
 310 FOR J%=0 TO DS%-1
 320 R1%=INT(RND(1)*16 + 1)
 330 R2%=INT(RND(1)*16 + 1)
 340 A%=A%+I%(R1%,R2%) * 2^J%
 350 NEXT
 355 FOR C%=0 TO 7
 360 IF FNdget(A%,C%) > 0 THEN CS%(C%)=CS%(C%)+1
 370 NEXT C%
 380 NEXT I%
 382 CX%=0
 385 FOR C%=0 TO 7
 388 PRINT "Class";C%;" has a score of ";CS%(C%)/
SX%*100;SPC(8)
 390 IF CS%(C%)>CS%(CX%) THEN CX%=C%
 400 NEXT C%
 404 PRINT "Most likely class is no. ";CX%
 410 INPUT "Do you want to classify another image
(Y=Yes) ", A$
 420 IF A$="Y" OR A$="y" THEN GOTO 240
 440 PRINT: PRINT "Bye!"
 444 END
 999 :
1000 REM -- Memory Initialization:
1010 FOR I%=0 TO MEMO%
1020 REM -- uses byte-subscripting (?):
1030 ?(D%+I%)=0
1040 NEXT
1050 RETURN
1199 :
1200 DEF PROCdset(A%,C%)
1210 REM --------
1220 ?(D%+A%) = (D%?A%) OR 2^C%
1230 ENDPROC
1240 REM sets a bit in D% array
1244 :
1250 DEF FNdget(A%,C%)
1260 REM ------
1270 =(D%?A%) AND 2^C%
1280 REM extracts bit from D% array
1288 REM A% is address; C% is 0..7
1290 :
2000 REM -- Image-making Routine:
2010 CLS: PRINT TAB(0,20);"(Use: U, D, L, R, X, *
or space"
2012 PRINT "to define the image.)"
2013 PRINT MOOD$;" phase."
2015 PRINT TAB(1,1);
2020 T%=0: A$=" "
2030 H%=1: V%=1
2040 REPEAT  C$=INKEY$(222)
2050 IF C$="U" THEN V%=V%-1
2060 IF C$="D" THEN V%=V%+1
2070 IF C$="L" THEN H%=H%-1
2080 IF C$="R" THEN H%=H%+1
2090 IF H%>16 THEN H%=1
2095 IF H%<1 THEN H%=16
2100 IF V%>16 THEN V%=1
2105 IF V%<1 THEN V%=16
2110 IF C$=" " THEN T%=0: A$=C$
2120 IF C$="*" THEN T%=1: A$=C$
2130 I%(H%,V%)=T%
2140 PRINT TAB(H%,V%);A$;
2150 UNTIL C$="X"
2155 GOSUB 2200 : REM Re-display
2160 RETURN
2170 :
2200 REM -- display Routine:
2210 FOR H%=1 TO 16
2220 FOR V%=1 TO 16
2230 IF I%(H%,V%)>0 THEN PRINT TAB(H%,V%);":" ELS
E PRINT TAB(H%,V%);" ";
2240 NEXT: NEXT
2250 PRINT TAB(0,17);
2260 RETURN
2270 :
2300 REM 'painting' commands are:
2301 REM U & D for Up and Down;
2302 REM L & R for Left and Right;
2303 REM * & space for On and Off;
2304 REM X to exit (image done).
```

Commodore 64

```
 10 REM **** CBM 64 PATTERN ****
 20 REM ****   RECOGNISER   ****
 30 FORI=1 TO 25:DW$=DW$+CHR$(17):NEXT I
 56 SX=40:DS=6:RB=2^DS
 57 ME=SX*RB
 60 DIM I(16,16),C(8)
 64 D=12*4096:REM USE $C000- FOR D ARRAY
 85 PRINT"TOTAL RAM SIZE=";ME;" BYTES"
 88 GOSUB 1000:REM CLEAR D ARRAY
 90 REM **** THE TRAINING PHASE ****
 92 M$="TRAINING"
 95 INPUT"WHICH CLASS IS (1...8)";C
 96 IFC<1 OR C>8 THEN 95
100 FOR I=1 TO 16:FOR J=1 TO 16:I(I,J)=0
101 NEXT J:NEXT I
110 REM **** NOW GET USERPAINTED IMAGE ****
120 GOSUB 2000:REM IMAGE MAKER
```





Links außen sehen Sie ein Beispiel von Mustererkennung auf einer unteren Ebene. Das dreimal-drei-Raster enthält die sogenannten „lokalen Operatoren", die die Bilddaten abtasten. Die Operatoren werden über die Bilddaten bewegt, und jede Ziffer wird mit der Grauwertintensität der Musterauswahl multipliziert. Es entstehen hohe Werte, wenn das gewünschte Merkmal vorhanden ist.

```
130 R=RND(-1):REM RANDOM SEED VALUE
140 FORI=1 TO SX
150 A=RB*(I-1):REM CALC ADDRESS OF RAM BANK
160 FOR J=0 TO (DS-1)
170 R1=INT(RND(1)*16+1):R2=INT(RND(1)*16+1)
190 A=A+I(R1,R2)*2^J
200 NEXT J
210 POKE(D+A),(PEEK(D+A) OR 2^(C-1))
220 NEXT I
230 INPUT"ANOTHER TRAINING SESSION (Y/N)";A$
233 IF A$="Y" THEN 90:REM REPEAT
234 REM **** NOW RECOGINITION PHASE ****
235 M$="RECOGNITION"
240 FOR I=1 TO 16:FOR J=1 TO 16
244 I(I,J)=0
245 NEXT J:NEXT I
250 GOSUB 2000:REM GET AN IMAGE FOR CLASSIFICATION
270 R=RND(-1)
280 FORC=1 TO 8:C(C)=0:NEXT C
290 FOR I=1 TO SX
300 A=RB*(I-1)
310 FOR J=0 TO DS-1
320 R1=INT(RND(1)*16+1):R2=INT(RND(1)*16+1)
340 A=A+I(R1,R2)*2^J
350 NEXT J
355 FOR C=1 TO 8
360 IF (PEEK(D+A)AND 2^(C-1)))>0 THEN C(C)=C(C)+1
370 NEXT C
380 NEXT I
381 PRINTCHR$(147)
382 CX=0:X=0:Y=17:GOSUB 3000
385 FOR C=1 TO 8
388 PRINT"CLASS"C"HAS A SCORE OF";C(C)/SX*100
390 IF C(C)>C(CX) THEN CX=C
400 NEXT C
404 PRINT"MOST LIKELY CLASS IS NO.";CX
410 INPUT "CLASSIFY ANOTHER IMAGE (Y/N)";A$
420 IF A$="Y" THEN 240:REM REPEAT
444 END
999 :
1000 REM ***** INIT MEMORY ****
1010 FOR I=0 TO ME:POKE(D+I),0:NEXT I
1030 RETURN
1040 :
2000 REM **** IMAGE MAKER ****
2005 PRINTCHR$(147)
2010 X=0:Y=20:GOSUB 3000:PRINT"USE U,D,L,R,X,* OR
SPACE"
2012 PRINT"TO DEFINE THE IMAGE"
2013 PRINTM$;" PHASE"
2015 X=1:Y=1:GOSUB 3000
2020 T=0:A$=" ":H=1:V=1
2040 GET C$
2050 IF C$="U" THEN V=V-1:IF V<1 THEN V=16
2060 IF C$="D" THEN V=V+1:IF V>16 THEN V=1
2070 IF C$="L" THEN H=H-1:IF H<1 THEN H=16
2080 IF C$="R" THEN H=H+1:IF H>16 THEN H=1
2110 IF C$=" " THEN T=0:A$=C$
2120 IF C$="*" THEN T=1:A$=C$
2130 I(H,V)=T
2140 X=H:Y=V:GOSUB 3000:PRINTA$;
2150 IF C$<>"X" THEN 2040
2155 GOSUB 2200
2160 RETURN
2170 :
2200 REM **** DISPLAY ROUTINE ****
2210 FOR H=1 TO 16
2220 FOR V=1 TO 16
2230 X=H:Y=V:GOSUB3000:IF I(H,V)=0 THEN PRINT":":G
OTO2240
2235 PRINT" ";
2240 NEXT V:NEXT H
2250 X=0:Y=17:GOSUB 3000
2260 RETURN
2270 :
3000 REM **** TAB COMMAND ****
3010 PRINTCHR$(19);TAB(X);LEFT$(DW$,Y);
3020 RETURN
```

Gruppen von acht Punkten werden willkürlich zu Acht-Bit-Registern verbunden, um die Basis des WISARD-Mustererkennungssystems zu bilden. Jede Achtergruppe (octuple) entspricht einer Bank von 256 Speicheradressen. In der Trainingsphase stellt der Wert der Achtergruppe eine Adresse innerhalb seiner RAM-Bank dar. Eine 1 wird vor die Adresse gesetzt. Ist dasselbe Muster während der Erkennungsphase vorhanden, wird die RAM-Adresse generiert, und die vorhandene 1 zeigt die Erkennung an.

BASIC-Dialekte

Spectrum
Der Spectrum verfügt nicht über bitweise AND- und OR-Anweisungen. Wir haben deshalb einige kurze Maschinencode-Routinen geschrieben, die diese Anweisungen ersetzen. Sie sind in Form eines BASIC-Loaders gegeben. Dabei sind folgende Änderungen (bei der C64-Version) durchzuführen:

```
30 CLEAR 49999:GOSUB 4000

64 LET D=50018

130 RANDOMIZE 1

210 LET X1=PEEK(D+A):LET Y1=2↑(C-1)
    :GOSUB 6000
215 POKE(D+A),R1

270 RANDOMIZE 1

360 LET X1=PEEK(D+A) :LET Y1=2↑(C-1)
    :GOSUB 5000
365 IF R1>0 THEN C(C)=C(C)+1

381  CLS

2005 CLS

3010 PRINT AT Y,X;

4000 FOR I=50000 TO 50017
4010 READ A:POKE I,A:NEXT I
4020 RETURN
4030 DATA 62,0,14,0,161,6,0,79,201
4040 DATA 62,0,14,0,177,6,0,79,201

5000 REM **** AND ****
5010 POKE 50001,X1:POKE 50003,Y1
5020 LET R1=USR 50000:RETURN

6000 REM **** OR ****
6010 POKE 50010,X1:POKE 50012,Y1
6020 LET R1=USR 50009:RETURN
```

Minenspiel

Bisher haben wir in unserem Projekt alle Routinen für das Grundgerüst des Minenfeldspiels erstellt. Nun werden Verfeinerungen eingebaut, die das Spiel optisch verbessern und den Spielreiz erhöhen.

Zuerst wird eine „Heckenschützen"-Routine erstellt. Dazu wird ein Schütze simuliert, der über das Minenfeld schießt, um das Suchgerät oder den Assistenten zu treffen. Das Geschoß wird als hochauflösende Linie dargestellt, die vom linken Feldrand nach rechts gezogen wird. Um ein Zufallselement für die Richtung einzubauen, werden die Koordinaten der Start- und Zielpunkte mit der RND-Funktion definiert. Die Werte von xstart und xfinish werden in der Prozedur „initialise variables" gesetzt. Die Differenz zwischen den beiden Werten beträgt 1024 Grafik-Einheiten. Wenn überprüft werden soll, was getroffen wurde, muß ein kurzer Teil der Linie gezeichnet, dann der Bereich auf Vorkommen der logischen Farbe 1 (mittels des POINT-Befehls) hin überprüft und dann der nächste kurze Teil der Linie gezogen werden. Diese Sequenz wird so lange wiederholt, bis ein Treffer erzielt wurde oder die Linie den anderen Rand erreicht hat.

Wir müssen nun die Schrittlänge festlegen. Bei einer sehr kurzen Schrittlänge dauert es sehr lange, bis die Linie vollständig gezogen ist. Haben wir eine zu große Schrittlänge, können Treffer übersehen werden. Da jede Zeichenzelle 64 Grafikeinheiten entspricht, wäre eine Schrittlänge von einer halben Zeichenzelle (32 Grafikeinheiten) sinnvoll. Wird in der X-Richtung (dx) als Länge 32 festgelegt, kann die gesamte Linie in 1024/32 = 32 Schritten gezogen werden. Werden die Y-Koordinaten des Start- und Zielpunkts zufällig gesetzt, kann die Schrittlänge in Y-Richtung (dy) durch Division der Werte durch 32 errechnet werden.

Unser letztes Problem ist das Löschen der Linie. Die Lösung ergibt sich aus dem Konzept der logischen Farben und ihrer Fähigkeit, untereinander logische Operationen auszuführen. Im Modus 5 gibt es 4 logische Farben:

Logische Farbe	0	1	2	3
Binär-Äquivalent	00	01	10	11
Aktuelle Farbe	schwarz	rot	gelb	weiß

Mit GCOL können verschiedene logische Operationen zwischen der aktuellen Zeichenfarbe und der Hintergrundfarbe ausgeführt werden. Der Befehl hat zwei Parameter, wobei der zweite die logische Farbe angibt, in der gezeichnet wird. Der erste Parameter setzt die Zeichenmethode:

GCOL0	Zeichnet die angegebene Farbe
GCOL1	Führt eine OR-Operation durch
GCOL2	Führt eine AND-Operation durch
GCOL3	Führt eine Exklusiv-OR-Operation durch
GCOL4	NOT-Operation mit aktueller Farbe

Ist an der Position, an der wir zeichnen wollen, die Farbe Weiß und wir wollen in Rot zeichnen, dann können mit den einzelnen Methoden folgende Ergebnisse erzielt werden:

GCOL0,1	löscht Weiß und zeichnet Rot	
GCOL1,1	verknüpft Rot und Weiß mit OR, um Weiß zu erzielen	Rot 01 Weiß OR 11 Weiß 11
GCOL2,1	verknüpft Rot und Weiß mit AND, um Rot zu erzielen	Rot 01 Weiß AND 11 Rot 01
GCOL3,1	verknüpft Rot und Weiß mit EOR, um Gelb zu erzielen	Rot 01 Weiß EOR 11 Gelb 10
GCOL4,1	verknüpft Weiß mit NOT, um Schwarz zu erzielen	Weiß 11 Schwarz 00

Wie lösen wir nun unser Problem? Wir könnten die Linie in Weiß ziehen und mit Schwarz übermalen, um sie zu löschen. Ist jedoch zum Beispiel eine Mine unter der Linie, würde ein „Loch" zurückbleiben. Wir können aber Exklusiv-OR mit Rot und der Farbe verknüpfen, die sich unter der Linie befindet. Wenn die Linie eine weiße Fläche kreuzt, erhalten wir ein gelbes Liniensegment. Malen wir über dieselbe Fläche in Exklusiv-OR mit Rot, ist das Ergebnis:

Rot 01
Gelb 10
EOR __
Weiß 11

Die ursprüngliche Farbe wird also wieder erreicht. Sie können sicherstellen, daß über zwei Exklusiv-ORs immer die Originalfarbe ausgegeben wird. Diesen Umstand nutzen wir zum Löschen der Linie. Wenn wir die Originallinie mit EOR zeichnen und dann genau dieselbe Linie erneut mit EOR ziehen, löschen wir die Linie

und stellen den alten Fond wieder her. Nachfolgend sehen Sie das vollständige Listing.

```
3110DEF PROCsnipe
3120ystart=RND(750)+220
3130yfinish=RND(750)+220
3140dx=32:dy=(yfinish-ystart)/32
3150GCOL 3,3
3160PROCline
3170IF POINT(x,y)=1 THEN PROCexplode(x,y) ELSE PROCline
3180ENDPROC
```

Hier das Listing der Linien-Prozedur:

```
3450DEF PROCline
3460SOUND0,-8,4,5
3470x=xstart:y=ystart
3480MOVE x,y
3490REPEAT
3500DRAW x,y
3510x=x+dx:y=y+dy
3520UNTIL x>xfinish OR POINT(x,y)=1
3530ENDPROC
```

Wenn Sie eine musikalische Untermalung haben möchten, können sie ein kurzes Lied einfügen. Hier ein Beispiel-Listing:

```
4090DEF PROCmusic
4100REM ** 1ST BAR **
4110SOUND1,-8,213,5
4120SOUND1,-8,209,5
4130SOUND1,-8,213,5
4140SOUND1,-8,209,5
4150SOUND1,-8,213,5
4160SOUND1,-8,193,5
4170SOUND1,-8,205,5
4180SOUND1,-8,197,5
4190REM ** 2ND BAR **
4200SOUND1,-8,185,20
4210SOUND1,-8,165,5
4220SOUND1,-8,185,5
4230SOUND1,-8,193,20
4240REM ** 3RD BAR **
4250SOUND1,-8,165,5
4260SOUND1,-8,193,5
4270SOUND1,-8,197,20
4280ENDPROC
```

TITELSEITE: Wir können die Technik von EOR- und relativen Punkt-Zeichnungen für einen Titelvorspann verwenden. Folgende Prozedur zeichnet das Wort MINES in hochauflösender Grafik. Jede neue Zeile des Wortes wird in Relation zur vorigen Zeile gezeichnet. Deshalb können wir das gesamte Wort durch Angabe des Startpunktes an jeder beliebigen Stelle auf dem Schirm darstellen. Wenn wir das Wort zeichnen und dann mit EOR erneut zeichnen, bevor um eine Linie weitergegangen wird, scheint das Wort über den Bildschirm zu wandern. GCOL0,129 erzeugt einen roten Hintergrund. Durch ein nachgesetztes CLG wird der gesamte Schirm rot. Zugleich können wir das Lied durch Aufruf von PROCmusic ertönen lassen. Die Information in PROCmusic wird schneller verarbeitet, als sie abgespielt werden kann. Deshalb müssen die SOUND-Informationen in einem Buffer abgefangen werden, bis sie gespielt werden.
SCHWIERIGKEITSGRADE: Um das Spiel spannender zu machen, werden Schwierigkeitsgrade vorgegeben. Nach Darstellung des Titels wird eine Zahl zwischen 0 und 9 abgefragt, die in der Variablen „skill" abgelegt ist. Damit kann die Anzahl von Minen im Feld sowie die Häufigkeit der Schüsse über den Bildschirm erhöht werden. Der erste Faktor kann durch Verändern der setup-Prozedur bestimmt werden. Verändern Sie Zeile 1930 und 1940 wie folgt:

```
1930factor=skill*3+30
1940PROClay_mines(factor)
```

Wir müssen beim Neusetzen der Minen in der reset-Prozedur die verbleibenden Minen errechnen. Ändern Sie Zeile 3950 folgendermaßen:

```
3950mines_left=factor-score/150
```

Das Listing für die Titelseite ist:

```
1300DEF PROCtitle_page
1310GCOL 0,129
1320CLG
1330GCOL 3,3
1340PROCmusic
1350Y=100:X=0
1360REPEAT
1370X=X+20:Y=Y+50
1380FOR I=1 TO 2
1390PROCmines
1400NEXT I
1410UNTIL Y>700
1420:
1430PROCmines
1440PRINTTAB(0,20)"Skill factor (0-9)?"
1450PROCmusic
1460REPEAT
1470skill=GET-48
1480UNTIL skill>-1 AND skill<10
1490ENDPROC
1500:
1510DEF PROCmines
1520PLOT4,X,Y
1530REM ** LETTER M **
1540PLOT1,0,200
1550PLOT1,80,-100
1560PLOT1,80,100
1570PLOT1,0,-200
1580REM ** LETTER I **
1590PLOT0,40,0
1600PLOT1,80,0
1610PLOT0,-40,0
1620PLOT1,0,200
1630PLOT0,-40,0
1640PLOT1,80,0
1650REM ** LETTER N **
1660PLOT0,40,-200
1670PLOT1,0,200
1680PLOT1,120,-200
1690PLOT1,0,200
1700REM ** LETTER E **
1710PLOT0,160,0
1720PLOT1,-120,0
1730PLOT1,0,-200
1740PLOT1,120,0
1750PLOT0,-40,100
1760PLOT1,-80,0
1770REM ** LETTER S **
1780PLOT0,280,60
1790PLOT1,0,40
1800PLOT1,-120,0
1810PLOT1,0,-100
1820PLOT1,120,0
1830PLOT1,0,-100
1840PLOT1,-120,0
1850PLOT1,0,40
1860ENDPROC
```

Bisher haben wir ein zeitweise aufrufendes Programm zum Verbinden unserer Prozeduren verwendet. Da jetzt alle für die Hauptprogrammschleife des Spieles erforderlichen Prozeduren assembliert worden sind, löschen Sie das aufrufende Programm, und geben Sie folgendes Listing ein:

```
2020DEF PROCloop
2030REPEAT
2040PROCupdate_time
2050PROCtest_keyboard
2060rand=RND(50-skill)
2070IF rand=1 THEN PROCsnipe
2080 UNTIL TIME>12099 OR end_flag=1
2090ENDPROC
```

Jetzt kann unser Aufruf-Programm geschrieben werden. Geben Sie ein:

```
1060hi_score$="00000"
1110MODE5
1120REM ** TURN OFF CURSOR **
1130VDU23;8202;0;0;0;
1140PROCtitle_page
1150CLS
1160PROCsetup
1170:
1180PROCloop
```

Guter Tastsinn

Dieser Abschnitt enthält ein Programm, mit dem der Roboter eine vorgegebene Fläche abtastet und die Form eines darin befindlichen Gegenstands auf dem Bildschirm darstellt.

Zum Abtasten einer Fläche brauchen wir ein geeignetes Bewegungsmuster, mit dem sich das Gebiet völlig abdecken läßt. Eine Möglichkeit ist, den Roboter auf der Suche nach einem Gegenstand vor- und rückwärts über die Fläche fahren zu lassen, bis er auf ein Hindernis trifft. Vor dem Abtasten der restlichen Fläche könnte er den Gegenstand erst einmal von allen Seiten „vermessen". Dieser Algorithmus ist jedoch sehr schwer zu programmieren und würde, falls mehrere Hindernisse vorhanden sind, zu Problemen führen. Eine andere Möglichkeit ist ein Programm, das den Roboter beim Berühren eines Objekts nur die Fahrtrichtung ändern läßt, den Tastvorgang aber weiter nicht beeinflußt. Hier die Einzelschritte des Lösungsansatzes:

```
REPEAT
  REPEAT
  Vorwärts bis zu einem Hindernis
  UNTIL Seitliche Grenze der Suchfläche
  erreicht oder Gegenstand gefunden
Um die Breite eines Streifens versetzen
Wenden
UNTIL obere Grenze der Suchfläche erreicht
```

Bei der horizontalen Suche werden einige Teile der Fläche nicht überstrichen, wenn ein Hindernis im Wege ist – das Gebiet dahinter bildet noch einen „weißen Fleck auf der Landkarte". Also brauchen wir noch mindestens einen zweiten Suchvorgang im Winkel von 90 Grad zum ersten Abtasten.

Bevor der Roboter erfolgreich suchen kann, muß an seinen beiden vorderen Sensoren ein kleiner Umbau vorgenommen werden: Da die Räder des Roboters seitlich vorstehen, könnte er das Hindernis beim seitlichen Vorbeifahren berühren. Daher müssen die Sensoren eine vergrößerte Tastfläche erhalten. Jede dieser Tastflächen besteht aus einem Plastikrechteck, etwa 95 mm x 25 mm groß. Das Material muß leicht sein, damit die Sensorschalter nicht schon allein durch sein Gewicht geschlossen werden. Befestigen Sie die Tastschilder mit Klebestreifen oder Kontaktkleber so, daß sie seitlich mit der Breite der Räder abschließen. Danach die Sensoren testen, sie müssen sich weiterhin leicht öffnen und schließen!

Unser Programm für die Flächenabtastung mit dem Selbstbau-Roboter wurde für je eine horizontale und vertikale Abtastung geschrieben. Beim Abfahren jedes Einzelstreifens wird eine entsprechende Fläche auf dem Bildschirm eingefärbt. Nach dem Abtasten wird die gesamte Fläche – so, wie der Roboter sie „sieht" – auf dem Bildschirm dargestellt.

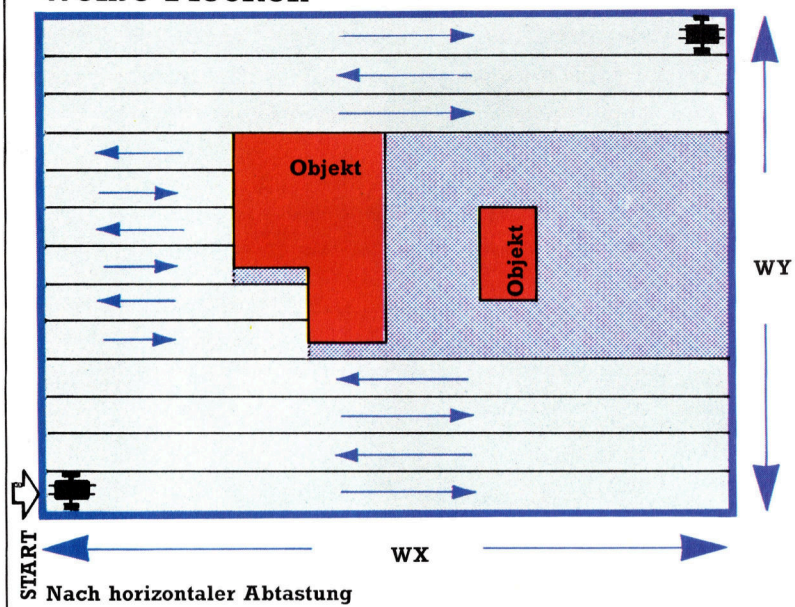
Bei beiden Programmversionen muß eingegeben werden, wie groß die abzutastende Fläche ist. Im mode 4 des Acorn B bzw. bei Verwendung der hochauflösenden Grafik beim C 64 entspricht jedes Bildschirm-Pixel einem 4 x 4 mm großen Quadrat im abgetasteten Gebiet, das somit bei beiden Rechnern maximal 1279 x 1023 mm groß sein darf. Die Breite eines Abtast-Streifens beträgt für beide Programme jeweils 40 mm.

Parallele Striche

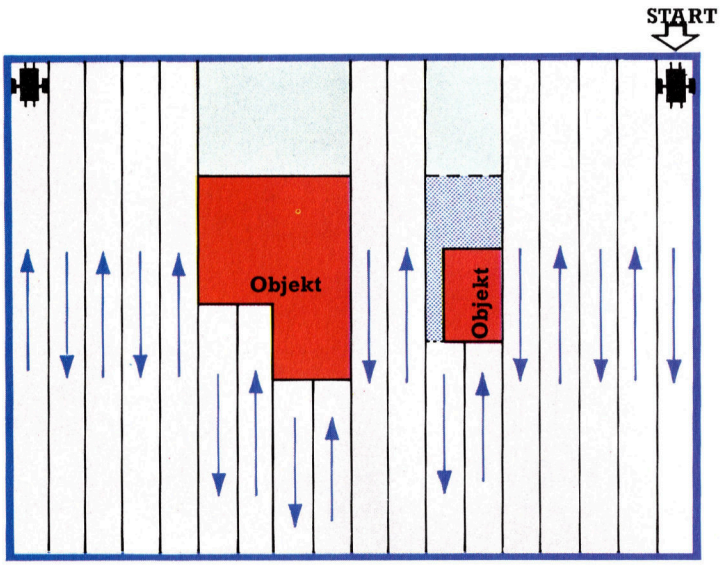
Wenn der Roboter einen Streifen abgetastet hat, wird die grafische Darstellung mit den Prozeduren (PROC) bzw. den Unterprogrammen XPLOT und YPLOT aktualisiert. Bei den meisten Computern wird der jeweilige Streifen einfach mit mehreren parallelen Strichen durch die Befehle MOVE und DRAW gezeichnet. Im Commodore-BASIC gibt es keine Befehle zum Zeichnen in hochauflösender Grafik. Wir haben daher zwei Maschinensprache-Routinen für diesen Zweck eingesetzt, die bereits früher im Kurs behandelt wurden. Vielleicht haben Sie noch Kopien der Objektfiles dieser Routinen, die mit Zeile 30 und 40 der Programmversion für den Commodore geladen werden können. Alternativ können Sie auch die BASIC-Loader beider Routinen verwenden. Jede Routine muß einzeln geladen und gestartet werden, bevor das gesamte Programm laufen kann. Zeile 30 und 40 fallen dann dabei weg.

Neben „weißen Flecken" können auch noch andere Probleme auftreten. So ist es möglich, daß das Programm bei unterschiedlichen Formen oder beim Abtasten in verschiedene Richtungen nicht immer gleich gut arbeitet. Am besten funktioniert es bei rechtwinkligen Formen, deren Geraden mit der Abtastrichtung parallel bzw. im rechten Winkel dazu stehen. Dreieckige und runde Formen erzeugen größere „weiße Flecken", weil das Abtastmuster ihre Umrisse in gerade Linien verwandelt. Schwierigkeiten gibt es auch, wenn der Roboter beim Umsetzen auf den nächsten Taststreifen mit einem Hindernis kollidiert. Zur Vereinfachung werden die Sensoren während dieses Manövers nicht abgefragt. Eine gute Übung ist, das Programm durch eine weitere vertikale Abtastung auszubauen.

Weiße Flecken



Nach horizontaler Abtastung



Nach vertikaler Abtastung

Bedeutung:

Horizontal abgetastete Flächen

Vertikal abgetastete Flächen

Nicht abgetastete Flächen

Am einfachsten kann eine rechteckige Fläche in zwei getrennten Schritten abgetastet werden, bei denen die Suchrichtung um 90 Grad gedreht wird. Bei Berührung mit den Kanten der Fläche oder mit einem Hindernis „geht" der Roboter um einen Streifen zur Seite und fährt in Gegenrichtung weiter.

Bei dieser Methode bleiben hinter jedem Gegenstand „weiße Flecken", die allerdings im zweiten Schritt zum Teil noch abgedeckt werden. Ganz exakt kann das Abtasten aber wegen der Breite der einzelnen Taststreifen nicht sein. Auch durch das gegenseitige Verdecken mehrerer Gegenstände treten Fehler leicht auf.

Acorn B

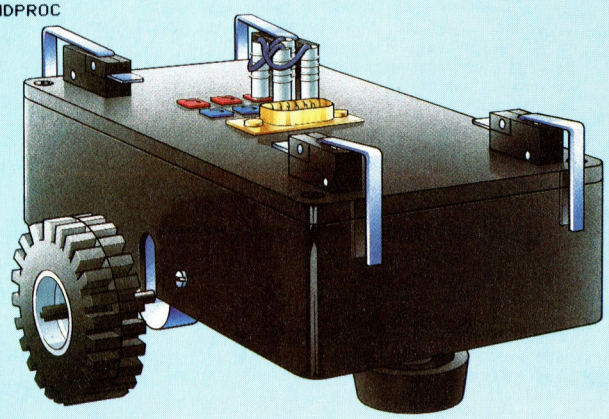
```
 10REM **** BBC SHAPE SCANNER ****
 20MODE 4
 30PROCinitialise
 40PROCscan_dimensions
 50PROChoriz_scan
 60PROCvert_scan
 70END
 80DEF PROChoriz_scan
 90 target=wx:sense=left:xstart=0
100REPEAT
110REPEAT
120PROCmove(forwards,dx):x=x+dx
130UNTIL(?DATREG AND 192)<>neither_bumpers OR x=target
140PROCxplot
150dx=-dx:y=y+dy:xstart=x
160IF target=0 THEN target=wx ELSE target=0
170IF y<wy THEN PROCnext_strip(sense)
180IF sense=right THEN sense=left ELSE sense=right
190UNTIL y>=wy
200ENDPROC
210:
220DEF PROCvert_scan
230IF x=0 THEN sense=left:dx=width ELSE sense=right:dx=-width
```

Commodore C 64

```
10 REM **** CBM SHAPE SCANNER ****
20 DN=8:REM IF CASS THEN DN=1
30 IF A=0 THEN A=1:LOAD"PLOTSUB.HEX",DN,1
40 IF A=1 THEN A=2:LOAD"LINESUB.HEX",DN,1
50 GOSUB1000:REM INITIALISE
60 GOSUB2000:REM SCAN DIMENSIONS
70 GOSUB5000:REM SWITCH TO HIRES MODE
80 GOSUB3000:REM HORIZONTAL SCAN
90 GOSUB4000:REM VERTICAL SCAN
100 GOSUB5100:REM OUT OF HIRES MODE
110 END
120 :
1000 REM **** INITIALISE S/R ****
1010 DDR=56579:DATREG=56577
1020 POKE DDR,15:POKE DATREG,1
1030 FW=4:BW=2:LF=6:RT=0
1040 PD=3.14446:PA=375/90
1050 RB=128:LB=64:BB=0:NB=192
1060 WD=40:DW=WD/10:X=0:Y=0:DX=DW:DY=WD
1070 REM ** M/C START ADRRESSES **
1080 HIRES=49422:LINESUB=49934
1090 RETURN
1100 :
```

Fortsetzung Acorn B

```
240dy=-dw:target=0:ystart=y
250PROCturn(sense,90)
260REPEAT
270REPEAT
280PROCmove(forwards,dy):y=y+dy
290UNTIL(?DATREG AND 192)<>neither_bumpers OR y=target
300PROCyplot
310dy=-dy:x=x+dx:ystart=y
320IF target=0 THEN target=wy ELSE target=0
330PROCnext_strip(sense)
340IF sense=right THEN sense=left ELSE sense=right
350UNTIL x>=wx OR x<0
360ENDPROC
370:
380DEF PROCnext_strip(way)
390PROCmove(backwards,30)
400PROCturn(way,90)
410PROCmove(forwards,width)
420PROCturn(way,90)
430ENDPROC
440:
450DEF PROCscan_dimensions
460INPUT"X DIMENSION IN MM";wx
470 IF wx>1279 THEN 460
480INPUT"Y DIMENSION IN MM";wy
490 IF wy>1023 THEN 480
500wx=width*(wx DIV width)
510wy=width*(wy DIV width)
520CLS
530ENDPROC
540:
550DEF PROCinitialise
560DDR=&FE62:DATREG=&FE60
570?DDR=15:REM LINES 0-3 OUTPUT
580?DATREG=1:REM TURN ON RESET BIT
590forwards=4:backwards=2:left=6:right=0
600pd_ratio=3.34446:pa_ratio=375/90
610right_bumper=128:left_bumper=64
620both_bumpers=0:neither_bumpers=192
630width=40:dw=width/10
640 x=0:y=0:dx=dw:dy=width
650MOVE 0,0
660ENDPROC
670:
680DEF PROCmove(dir,distance)
690?DATREG=(?DATREG AND 1)OR dir
700pulses=pd_ratio*ABS(distance)
710FOR I=1 TO pulses:PROCpulse:NEXT I
720ENDPROC
730:
740DEF PROCturn(dir,angle)
750?DATREG=(?DATREG AND 1)OR dir
760pulses=pa_ratio*angle
770FOR I=1 TO pulses:PROCpulse:NEXT I
780ENDPROC
790:
800DEF PROCpulse
810?DATREG=(?DATREG OR 8)
820?DATREG=(?DATREG AND 247)
830ENDPROC
840DEF PROCxplot
850FOR I=0 TO width STEP 4
860MOVExstart,y+I
870DRAWx,y+I
880NEXT I
890ENDPROC
900:
910DEF PROCyplot
920FOR I=0 TO width STEP 4
930MOVEx+I,ystart
940DRAWx+I,y
950NEXT I
960ENDPROC
```



Fortsetzung Commodore C 64

```
2000 REM **** SCAN DIMENSIONS ****
2010 INPUT"X DIMENSION IN MM";WX
2020 IF WX/4>319 THEN 2010
2030 INPUT"Y DIMENSION IN MM";WY
2040 IF WY/4>199 THEN 2030
2050 WX=WD*INT(WX/WD):WY=WD*INT(WY/WD)
2060 RETURN
2070 :
3000 REM **** HORIZONTAL SCAN ****
3010 TG=WX:SE=LF:XS=0
3020 DR=FW:DS=DX:GOSUB7000:REM MOVE
3030 X=X+DX
3040 IF(PEEK(DATREG)AND192)=NB AND X<>TG THEN 3020
3050 GOSUB9000:REM XPLOT
3060 DX=-DX:Y=Y+DY:XS=X
3070 IF TG=0 THEN TG=WX:GOTO 3090
3080 TG=0
3090 IF Y<WY THEN WA=SE:GOSUB8000:REM NEXT STRIP
3100 IF SE=RT THEN SE=LF:GOTO 3120
3110 SE=RT
3120 IF Y<WY THEN 3020
3130 RETURN
3140 :
4000 REM **** VERTICAL SCAN ****
4010 IF X=0 THEN SE=LF:DX=WD:GOTO 4030
4020 SE=RT:DX=-WD
4030 DY=-DY:TG=0:YS=Y
4040 DR=SE:AG=90:GOSUB7100:REM TURN
4050 DR=FW:DS=DY:GOSUB7000:REM MOVE
4060 Y=Y+DY
4070 IF(PEEK(DATREG)AND192)=NB AND Y<>TG THEN 4050
4080 GOSUB9100:REM YPLOT
4090 DY=-DY:X=X+DX:YS=Y
4100 IF TG=0 THEN TG=WY:GOTO 4120
4110 TG=0
4120 WA=SE:GOSUB8000:REM NEXT STRIP
4130 IF SE=RT THEN SE=LF:GOTO 4150
4140 SE=RT
4150 IF X<WX AND X>0 THEN 4050:REM REPEAT
4160 RETURN
4170 :
5000 REM **** ENTER HIRES ****
5010 POKE 49408,1:POKE 49409,1
5020 POKE 49410,1:SYS HIRES:RETURN
5030 :
5100 REM **** LEAVE HIRES ****
5110 POKE 49408,0:POKE 49409,0
5120 POKE 49410,1:SYS HIRES:RETURN
5130 :
6000 REM **** ENTER LINESUB ****
6010 MHI=INT(X1/256):MLO=X1-256*MHI
6020 NHI=INT(X2/256):NLO=X2-256*NHI
6030 POKE 49920,MLO:POKE 49921,MHI
6040 POKE 49922,NLO:POKE 49923,NHI
6050 POKE 49924,Y1:POKE 49925,Y2
6060 SYS LINESUB:RETURN
6070 :
7000 REM **** MOVE (DR,DS) ****
7010 POKE DATREG,(PEEK(DATREG)AND 1)OR DR
7020 PL=PD*DS
7030 FOR I=1 TO PL:GOSUB7200:NEXT I
7040 RETURN
7050 :
7100 REM **** TURN (DR,AG) ****
7110 POKE DATREG,(PEEK(DATREG)AND 1)OR DR
7120 PL=PA*DS
7130 FOR I=1 TO PL:GOSUB7200:NEXT I
7140 RETURN
7150 :
7200 REM **** PULSE ****
7210 POKE DATREG,PEEK(DATREG)OR 8
7220 POKE DATREG,PEEK(DATREG)AND 247
7230 RETURN
7240 :
8000 REM **** NEXT STRIP ****
8010 DR=BW:DS=30:GOSUB7000:REM MOVE
8020 DR=WA:AG=90:GOSUB7100:REM TURN
8030 DR=FW:DS=WD:GOSUB7000:REM MOVE
8040 DR=WA:AG=90:GOSUB7100:REM TURN
8050 RETURN
9000 REM **** XPLOT ****
9010 FOR I=0 TO WD
9020 X1=XS/4:Y1=(Y+I)/4:X2=X/4:Y2=Y1
9030 GOSUB6000:REM ENTER LINESUB
9035 NEXT I
9040 RETURN
9100 REM **** YPLOT ****
9110 FOR I=0 TO WD
9120 X1=(X+I)/4:Y1=YS/4:X2=X1:Y2=Y/4
9130 GOSUB6000:REM ENTER LINESUB
9135 NEXT I
9140 RETURN
```

Ein- und Ausgabe

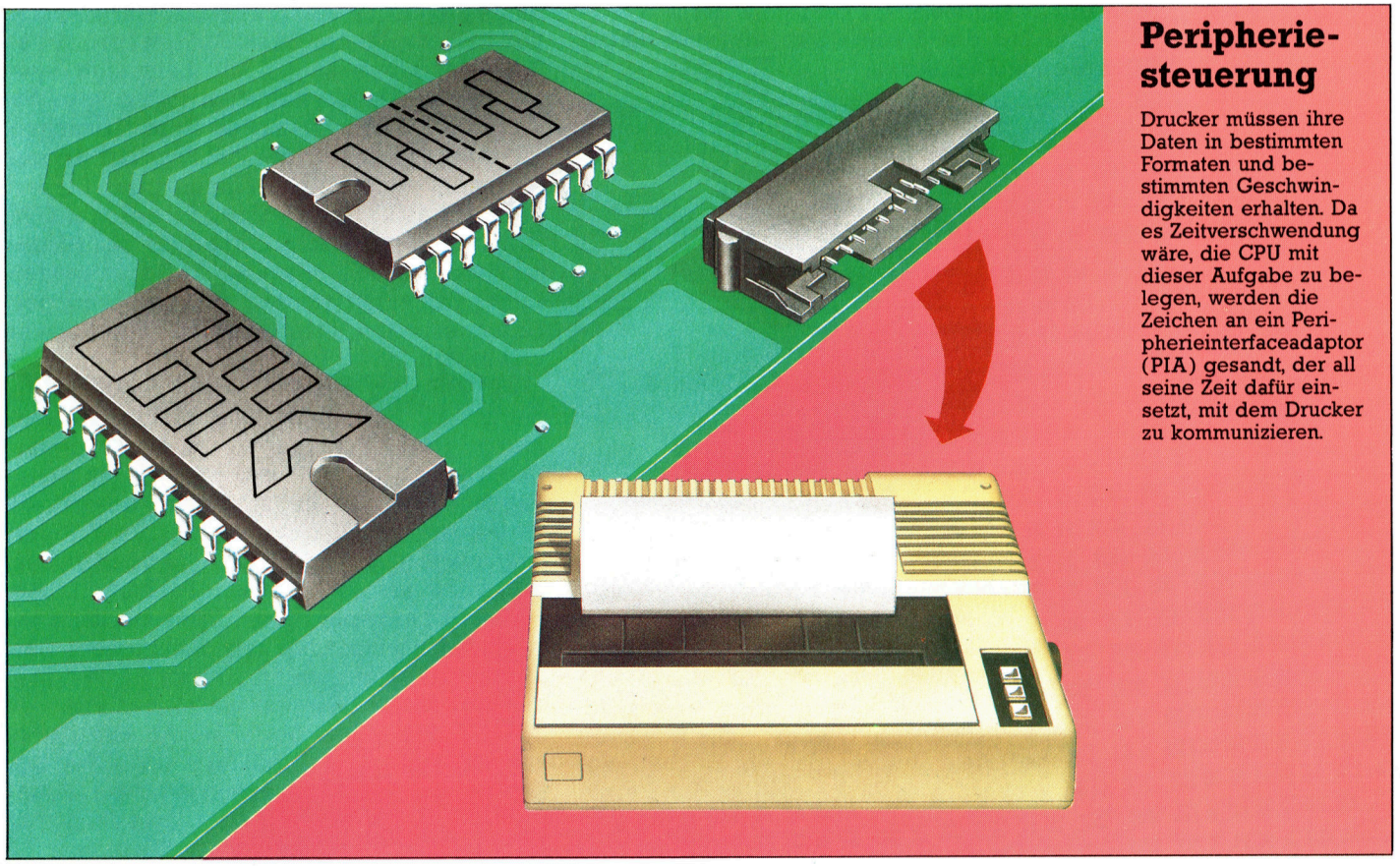
Die Steuerung von Ein- und Ausgabe ist ein wichtiger Aspekt der Assemblerprogrammierung. Wir sehen uns an, was in den beiden Schnittstellenchips 6820-PIA und 6850-ACIA vorgeht.

Der 6809-Prozessor hat wie der 6502 keinen getrennten Adreßraum für die Ein- und Ausgabe und kennt dafür auch keine speziellen Befehle. Die Ein- und Ausgabechips liegen im normalen Adreßbereich und werden wie Speicherstellen angesprochen. Diese Methode ist zwar schnell und einfach, blockiert aber einen Adreßbereich. Der 6809 kann daher trotz seines für 64 KByte ausgelegten Adreßbusses maximal nur 56 KByte einsetzen – ohne die Speicherverwaltung für Hard- und Software.

Zwar lassen sich einige Ein- und Ausgabegeräte direkt an den Systemdatenbus anschließen, doch sitzt normalerweise ein Schnittstellenchip dazwischen. Diese Chips sind hochentwickelte Schaltungen. Üblicherweise werden Chips der gleichen Prozessorfamilie eingesetzt, da der Anschluß und die Steuerung darauf abgestimmt sind. Mit dem 6809 wird für die parallele Ein- und Ausgabe meistens der 6820- (oder 6821-) PIA genommen und für die serielle E/A der 6850-ACIA. Beide Chips besitzen eine Reihe von Registern, die wie normale Speicherstellen angesprochen werden. Es gibt drei Registertypen:

- Steuerregister: In diese Register kann nur geschrieben werden. Die gespeicherten Werte programmieren den Chip für den gewünschten Vorgang, zum Beispiel zur Festlegung der Baud-Rate.
- Statusregister: Diese „Read-Only"-Register zeigen den Status des Chips an. Sie geben Auskunft darüber, ob Eingaben empfangen wurden, ob die letzte Ausgabe schon gesandt wurde oder ob Fehler aufgetreten sind.
- Datenregister: Hier liegen die Ein- und Ausgabedaten. Sie können als Schreib/Lese-Speicher funktionieren oder auch als reiner Lese- bzw. Schreibspeicher dienen.

Um Speicherplatz zu sparen, verwenden oft mehrere Register die gleiche Adresse. Wenn beispielsweise ein Statusregister und ein Steuerregister auf einer Adresse liegen, wird beim



Peripheriesteuerung

Drucker müssen ihre Daten in bestimmten Formaten und bestimmten Geschwindigkeiten erhalten. Da es Zeitverschwendung wäre, die CPU mit dieser Aufgabe zu belegen, werden die Zeichen an ein Peripherieinterfaceadaptor (PIA) gesandt, der all seine Zeit dafür einsetzt, mit dem Drucker zu kommunizieren.

Lesen das Statusregister angesprochen und beim Schreiben das Steuerregister. Ebenso teilen sich auch die Datenregister der Ein- und Ausgabe identische Adressen.

Der 6820-PIA enthält sechs Register und belegt vier aufeinanderfolgende Speicherbytes. Dabei versorgt der Chip zwei voneinander unabhängige Ausgänge, die beide je drei Register einsetzen. Die Peripherieseite des Chips hat für jeden Ausgang acht Datenleitungen und zwei Steuerleitungen zur Verfügung. Die beiden mit den entsprechenden Peripherieleitungen verbundenen Steuerleitungen geben Auskunft über den Status des Chips. Die Steuerleitung 1 ist für eingehende Signale reserviert, die Steuerleitung 2 kann Daten senden und empfangen.

Die drei Register haben folgende Aufgaben:

- Das Datenregister, dessen Bits sich einzeln setzen lassen, dient für die Ein- und Ausgabe.
- Das Datenrichtungsregister bestimmt, welche Bits des Datenregisters für die Eingabe und welche für die Ausgabe eingesetzt werden. Dafür werden die entsprechenden Bits des Richtungsregisters auf Eingabe (0) oder Ausgabe (1) gesetzt.
- Das dritte Register dient als Steuerungs- und Statusregister.

Das Datenrichtungsregister und das Datenregister liegen auf derselben Adresse. Ein Bit im Steuerregister bestimmt, welches der beiden Register dieser Adresse gerade angesprochen wird. Die nebenstehende Tabelle gibt die Offsets für die Registeradressen von der Basisadresse des Chips aus an.

Register	Offset
Daten A	0
Datenrichtung A	0
Steuer/Status A	1
Daten B	2
Datenrichtung B	2
Steuer/Status B	3

Die Bits des Steuer-/Statusregisters haben folgende Funktionen:

Bit	Funktion
7	Statusbit für Steuerleitung 1. Wird bei Empfang auf 1 gesetzt. Lesen des Datenregisters automatisch auf Null.
6	Statusbit für Steuerleitung 2; wie Bit 7
5	Bestimmt Einsatz der Steuerleitung 2 für Eingabe (0) oder Ausgabe (1)
4	Bestimmt Steuersignalart auf Leitung 2
3	Wenn Steuerleitung 2 auf Eingabe, wird hier mit 1 ein Interrupt von Bit 6 möglich. Hilft bei Ausgabe, Signalart festzustellen
2	Bestimmt, ob Daten- (1) oder Datenrichtungsregister (0) angesprochen werden
1	Bestimmt Steuersignalart auf Leitung 1
0	1 ermöglicht hier Interrupts von Bit 7

Der erste Teil unseres PIA-Beispielprogramms versetzt einen 6820-Chip in die Lage, einen Drucker über eine Standard-Centronic-Schnittstelle zu steuern, während der zweite Teil die Steuer- und Datenleitungen festlegt. Dabei signalisiert eine Steuerleitung („Strobe") dem Drucker, daß ein Zeichen „auf dem Weg ist". Der Strobe muß an die Steuerleitung 2 angeschlossen sein, die auf Ausgabe gestellt ist. Ein Steuersignal des Druckers (genannt „Acknowledge" – Bestätigung) zeigt an, daß er bereit ist, das nächste Zeichen zu empfangen. Dieses Signal sollte auf der Steuerleitung 1 ankommen. Die acht Datenleitungen werden dazu mit den acht Datenausgängen des PIA verbunden.

Für die Einstellung der PIA auf diese Aufgabe müssen wir das Datenrichtungsregister anwählen und alle acht Bits auf Ausgabe stellen, dann das Datenregister ansprechen und die Steuerleitung 2 ebenfalls auf Ausgabe stellen. Bei der Ausführung wird das Steuer-/Statusregister so lange gelesen, bis eine Eins in Bit 7 anzeigt, daß der Drucker empfangsbereit ist. Wir schreiben dann ein Zeichen in das Datenregister, das automatisch über die Steuerleitung 2 ein Steuersignal aussendet. Nach der Übertragung des Zeichens wird Bit 6 automatisch auf Eins gesetzt. Das Lesen des Datenregisters löscht nun Bit 6 und 7. Dieser Ablauf wird so lange wiederholt, bis das letzte Zeichen übermittelt ist.

Nehmen wir an, die Basisadresse der PIA liegt in einer Adreßtabelle bei $3000. Beim Aufruf der Druckroutine enthält das Prozessorregister A den Index zu dieser Tabelle und Y die Adresse des Strings, der gedruckt werden soll. Der String hat Normalformat, das heißt, sein erstes Byte ist das Längenbyte. Die erste Subroutine programmiert die PIA, die zweite druckt den String.

Der 6850-ACIA ist ein UART (Universal Asynchronous Receiver/Transmitter), der normalerweise für das RS232-Protokoll und für Modems eingesetzt wird. Er besitzt vier Register und belegt zwei Adressen. Auf der Peripherieseite des Chips liegen fünf Leitungen: eine Leitung für die übermittelten Daten, eine für die empfangenen Daten und drei Leitungen für den Handshaking-Betrieb. Zwei der Steuerleitungen empfangen eingehende Steuersignale – DCD (Data Carrier Detect) und CTS (Clear To Send), während die dritte das ausgehende Signal RTS (Request To Send) sendet. Diese Leitungen werden mit gleichnamigen Leitungen der Standard-RS232-Schnittstelle verbunden.

Die links am Rand stehende Tabelle zeigt die vier ACIA-Register. Mit Bit 7 des Steuerregisters werden Interrupts der empfangenen Daten ermöglicht. Bit 5 und 6 stellen die Übertragungsinterrupts an oder ab und bestimmen die Art des Steuersignals, das auf der RTS-Leitung ausgesandt wird. Bit 2, 3 und 4 bestimmen die Größe des übermittelten „Pakets". Für die serielle Übertragung eines Bytes werden normalerweise mindestens zehn Bits gesandt. Das Anfangsbit zeigt dem Empfänger an, daß Daten folgen. Die Daten selbst können aus sieben oder acht Bits bestehen. An die Daten kann ein Parity-Bit angehängt sein, das zur Feststellung von Übertragungsfehlern dient. Schließlich kommen ein oder zwei Stop-Bits. Insgesamt gibt es folgende Kombinationen:

Register	Offset
Steuerregister	0
Statusregister	0
Daten senden	1
Daten empfangen	2

Steuerregister			Anzahl der Daten-Bits	Parität	Anzahl der Stop-Bits
Bit 4	Bit 3	Bit 2			
0	0	0	7	gerade	2
0	0	1	7	ungerade	2
0	1	0	7	gerade	1
0	1	1	7	ungerade	1
1	0	0	8	keine	2
1	0	1	8	keine	1
1	1	0	8	gerade	1
1	1	1	8	ungerade	1

Die beiden niederwertigen Bits (0 und 1) setzen einen Teiler für die Taktfrequenz und bestimmen damit die Empfangs- und Übertragungsgeschwindigkeit. Da der 6850 keinen eigenen Taktgeber besitzt, wird ein externer verwandt, der normalerweise mit einer Frequenz von 1,760 Hz arbeitet.

Steuerregister		Teiler der Taktfrequenz
Bit 1	Bit 0	
0	0	1
0	1	16
1	0	64

Wenn beide Bits auf Eins stehen, wird im Chip ein „Master Reset" veranlaßt.

Die Bits des Statusregisters haben folgende Funktionen und werden gesetzt, wenn:

Bit	Funktion
7	Interruptanforderung vorliegt
6	bei Übertragung ein Paritätsfehler auftritt
5	Empfänger einen Überlauf hat (es wurden zu viele Bits empfangen, die die zuvor gesandten überschreiben)
4	beim Empfang Fehler in den Rahmenbits (die falsche Anzahl Anfangs- und Endbits) auftritt
3	Signal auf der CTS-Leitung empfangen
2	Signal auf der DCD-Leitung empfangen
1	Übertragungsdatenregister leer
0	Empfangsdatenregister leer

Unser zweites Programmbeispiel empfängt mit dem 6850 eine Zeichenkette (von einem Terminal), die mit Return abgeschlossen ist. Hier wird zunächst der Chip programmiert und dann mit einer Schleife abgefragt, ob das Empfangsdatenregister voll ist. Ist dies der Fall, wird das Datenbyte gespeichert und damit Bit 0 des Statusregisters auf Null gesetzt. Dieser Vorgang wird so lange wiederholt, bis ein Return (ASCII-Code 13) kommt. Das Programm ignoriert Übertragungsfehler, die sich jedoch leicht abfangen lassen, indem man den Inhalt des Statusregisters maskiert und testet, ob eins der Fehlerbits gesetzt ist. Wir setzen dabei ein weit verbreitetes Protokoll voraus: acht Datenbits, keine Parität, zwei Endbits und eine durch 16 geteilte Taktfrequenz. Die erste Subroutine programmiert den Chip, während die zweite die Daten empfängt.

PIA-Programm

```
TABLE   EQU   $3000
        ORG   $1000

        ASLA                    A links drehen, um mit Zwei zu multiplizieren (die Tabelle enthält Zwei-Byte-Adressen)
        LDX   #TABLE
        LDX   A,X               PIA-Adresse holen
        CLR   1,X               Zugang zum Datenrichtungsregister verschaffen
        LDB   #%11111111
        STB   ,X                Alle Bits auf Eingabe setzen
        LDB   #%00101100
        STB   1,X               Interrupts abschalten, Steuerleitung 2 auf Ausgabe setzen und Datenregister anwählen
        RTS

                                Subroutine für das Drucken eines Strings, dessen Adresse in Y steht
        ASLA                    A links drehen, um mit Zwei zu multiplizieren
        LDX   #TABLE            Basisadresse der PIA holen
        LDX   A,X
        LDA   ,Y+               Stringlänge in A speichern
LOOP1   BEQ   FINISH            Testen auf Länge Null
LOOP2   LDB   1,X               Bereit für nächstes Zeichen?
        ANDB  #%10000000        Alle Bits außer Bit 7 maskieren
        BEQ   LOOP2             Falls nicht bereit
        LDB   ,Y+               Nächstes Zeichen holen
        STB   ,X                Drucken
LOOP3   LDB   1,X               Testen, ob Zeichen übertragen
        ANDB  #%01000000        Bit 6 überprüfen
        BEQ   LOOP3             Schleife starten, falls nicht bereit
        LDB   ,X                Datenregister lesen, um die Statusbits zu löschen
        DECA                    Von der Länge Eins abziehen
        BRA   LOOP1             Nächstes Zeichen holen
FINISH  RTS
```

ACIA-Programm

```
TABLE   EQU   $3000             Subroutine für die Programmierung des 6850
        ORG   $1000             Subroutine zur Einstellung des ACIA

ACIAST  ASLA                    A links drehen, um mit Zwei zu multiplizieren (die Tabelle enthält Zwei-Byte-Adressen)
        LDX   #TABLE            Basisadresse des ACIA holen
        LDX   A,X
        LDA   #%00000011        Master Reset des ACIA
        STA   ,X                Steuerregister ansprechen
        LDA   #%00010001        ACIA programmieren (8 Datenbits, keine Parität, 2 Stopbits)
        STA   ,X
        RTS

                                Subroutine zum Aufnehmen eines Zeichenstrings
BUFFER  EQU   $4000             Hier wird der String gespeichert
CR      EQU   13                ASCII-Code für Return
        BSR   ACIAST            ACIA einstellen, das X-Register enthält die ACIA-Adresse
        LDY   #BUFFER           Bestimmungsort in Y
LOOP    LDB   ,X                Status holen
        ASLB                    Bit 7 aus dem B-Register ins Übertragungsflag des CCR schieben
        BCC   LOOP              Rückverzweigen, falls das Bit nicht gesetzt ist (keine Interruptanforderung)
        LDA   1,X
        STA   ,Y+               Datenbyte holen
        CMPA  #CR               Datenbyte speichern
        BNE   LOOP              Ist A ein Return?
        RTS                     Nächstes Zeichen
```

Absolut super

Die Cray-Familie der Supercomputer bildet die Elite der Großrechner. Industrie, Universitäten, Forschungslaboratorien der Regierungen sind Abnehmer dieser Rechner – ob nun Explosionen simuliert, Maschinen getestet oder Naturereignisse vorhergesagt werden sollen.

Das Erscheinungsbild des Cray-1 ist unverwechselbar. Der Computer befindet sich in einem kreisförmig angeordneten Gebilde. Die Stromversorgung liegt unter den runden Sitzbänken.

Ein Computer-Hersteller, der nach dem Verkauf von nur 16 Systemen zum Jahresende eine Rekordbilanz ziehen kann, muß ein recht ungewöhnliches Produkt herstellen. – So geschehen 1983, als Cray Research den Jahresbericht veröffentlichte und damit die Anzahl der weltweit installierten milliardenteuren Cray-Computer auf 65 gestiegen war.

Das Unternehmen wurde 1972 von Seymour R. Cray gegründet. Er sah damals eine wachsende Nachfrage nach leistungsfähigen Großrechnern voraus. Aufgrund dieser Erkenntnis setzte Cray Research 1983 rund 170 Millionen Dollar um, was einer Verzehnfachung des Umsatzes von vor fünf Jahren entsprach.

Dieser Erfolg basierte auf dem ersten Produkt des Unternehmens, dem Cray-1. Erstmals 1976 installiert, war dieser Rechner für Jahre der leistungsfähigste Computer der Welt. Rein äußerlich wirkte er sehr unauffällig.

Der billigste Cray-1/M-Großrechner kostet heute etwa 16 Millionen Mark. Einer der Hauptgründe für eine so kostspielige Investition ist seine Simulationsfähigkeit. Zeit und Kosten für das Konstruieren von einem Auto-, Schiff- oder Flugzeug-Modell nacheinander und die sich daran anschließenden Tests können gewaltige Dimensionen haben. Der Computer wird dazu eingesetzt, ein „Modell" zu konstruieren, das nur im Speicher des Rechners besteht, modifiziert wird und entsprechend bestimmter Variablenwerte verändert werden kann. Das so erzeugte Modell läßt sich unter allen erdenklichen Voraussetzungen und Bedingungen testen, selbst in Extremsituationen, die physikalisch nicht herstellbar sind. Zu den Cray-Kunden gehören Firmen wie Lockheed und General Motors. Die Systeme machten sich dort bereits nach wenigen Wochen bezahlt. Cray selbst stellt dazu fest, daß sich beispielsweise die Entwicklungskosten und -zeiten für ein neues Flugzeug mit einem seiner Rechner auf die Hälfte reduzieren lassen.

Simulationen sind auch für Wissenschaftler, die Vorhersagen von Naturereignissen treffen wollen, ungeheuer wichtig. So benutzt zum Beispiel auch das britische Wetteramt für seine Vorhersagen einen Cray-1. Aus Daten, die Lufttemperatur, Luftdruck, Feuchtigkeit etc. an verschiedenen Orten und in verschiedenen Höhen darstellen, berechnet der Computer die Bewegung der Luftmassen und die Veränderungen dieser physikalischen Größen über Stunden im voraus. Das beinhaltet eine Vielzahl von Berechnungen, deren Einzelergebnisse die Folgeergebnisse beeinflussen.

Ähnlich umfassend ist die Simulation bestimmter Geschehnisse für Atomforscher, Seismologen und Wissenschaftler in vielen anderen Bereichen. Cray Research ist der Überzeugung, daß die zunehmende Verwendung von Computersimulationen die herkömmliche physikalische Simulation (den Laborversuch) ablöst und so eine neue Ära der Forschung und des Ingenieurwesens einleitet.

Cray beschleunigt diese Revolution durch noch beeindruckendere Produkte. Die Cray-1/S-Serie wurde von der leistungsfähigeren Cray-1/M-Linie abgelöst. Die X-MP-Serie ist noch leistungsfähiger. Diese Rechner verfügen über ein, zwei, drei oder noch mehr Zentraleinheiten und kosten bis zu 50 Millionen Mark. Die Speicherkapazität umfaßt bis zu acht Millionen 64-Bit-Wörter. Der verwendete Massenspeicher wirkt sich sogar auf Entwicklungen im Heimcomputerbereich aus. Crays Diskettenstationen können bis zu 10 Megabyte Daten pro Sekunde übertragen.

Und wem das immer noch nicht schnell genug ist, der bedient sich Crays SSD (Solid-State Storage Device) mit einer Übertragungsgeschwindigkeit von bis zu 2000 Megabyte pro Sekunde aus dem Gigabyte-Speicher. Mit derartigen Maschinen wird Cray Research seine Spitzenposition als Computer-Wegweiser sicherlich weiter ausbauen.

Crays Fertigungsstätte in Chippewa Falls, Wisconsin. Hier werden hauptsächlich Chips hergestellt. Doch inzwischen beschäftigt sich Cray stark mit der Gallium-Arsenid-Technik.

Fachwörter von A bis Z

Identification = Benutzeridentifikation
In Rechnernetzen muß die Identität des Benutzers überprüft werden, um einen unberechtigten Zugriff zu vertraulichen Informationen zu verhindern. Die Identifikation erfolgt meist in der Form, daß der Benutzer sich mit Namen und Benutzernummer beim System anmeldet („einloggt") und zusätzlich ein Paßwort eintippt, das ihm den Zugang freigibt.

Weil sich gezeigt hat, daß dieses Verfahren keinen ausreichenden Schutz gegen Mißbrauch durch Hakker bietet, bemühen sich Benutzer wie Hersteller um zuverlässigere Maßnahmen. Dr. Kuno Zimmerman von der Universität Missouri-Columbia (USA) machte zum Beispiel den Vorschlag, daß der Benutzer auf einem kleinen Grafiktablett seinen Namenszug hinterlassen sollte, dessen charakteristische Merkmale dann mit dem im Rechner gespeicherten Original verglichen werden. Es gibt Unterschriftenfälscher ...

Handschriftliche Mitteilungen, auch Unterschriften zur Benutzeridentifikation, können mit einem Telefaxgerät telefonisch an die jeweilige Empfängerstation übermittelt werden. Die Übertragung erfolgt geräuschlos und verschlüsselt, so daß Abhörversuche sinnlos sind.

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.

IEEE = IEEE
Das amerikanische „Institute of Electrical and Electronic Engineers" ist 1963 aus dem „Institute of Radio Engineers" und dem „American Institute of Electrical Engineers" hervorgegangen. Dieser Verband mit weltweit über 200 000 Mitgliedern gibt technische Normen heraus.

Als interessierter und gestandener Heimcomputerbesitzer hatten Sie möglicherweise schon mit dem IEEE-488-Bus zu tun – ein verbreiteter Parallelschnittstellen-Standard für die Peripherie. Diese IEEE-Norm (gesprochen: „I triple E") schreibt vor, daß acht Leitungen für die byteweise Parallelübertragung der Daten vorliegen müssen, so viele Adreßleitungen, wie die höchste Systemadresse erfordert, und Steuerleitungen für Ein- und Ausgabe. Auch Quittungsbetrieb (Handshaking) ist vorgesehen, um eine korrekte Übertragung sicherzustellen. Der IEEE-Bus ist unter anderem zum Datenaustausch zwischen unterschiedlichen Rechnern geeignet. Für den Computerfreak also notwendig.

IF-THEN-ELSE = IF-THEN-ELSE
Die bedingte IF-THEN-Anweisung gehört zum Wortschatz fast aller Heimcomputer. Die ELSE-Erweiterung kennen leider nur die anspruchsvolleren Modelle. Das vollständige IF-THEN-ELSE ermöglicht die direkte Programmierung einer echten Verzweigung: Wenn die zum IF gehörige Bedingung erfüllt ist, führt der Rechner die auf THEN folgende Anweisung aus. Andernfalls springt er auf die mit ELSE eingeleitete Alternative. Diese kann beispielsweise im Aufruf eines Unterprogramms bestehen, das seinerseits weitere Bedingungen abfragt.

Häufig ist nur das IF-THEN zugelassen. Dann bearbeitet der Rechner bei nicht erfüllter Bedingung direkt die nächste Programmzeile, die er andernfalls auf dem Umweg über die THEN-Anweisung auch erreicht. Für eine echte Verzweigung müßten hier weitere IF-Befehle verwendet werden. Das ist umständlicher und macht die Fehlersuche schwieriger, als es bei der klaren Trennung mit IF-THEN-ELSE der Fall ist.

Impact Printer = Anschlagdrucker
Bei Anschlagdruckern entsteht das Schriftbild oder die Grafik durch mechanischen Anschlag von Druckwerkzeugen auf ein Farbband. Zu unterscheiden ist im wesentlichen zwischen Ganzzeichen- und Matrixdruckern. Ganzzeichendrucker erzeugen durch einmaligen Anschlag einer Relieftype gleich das vollständige Symbol, während bei Matrixdruckern jedes Zeichen aus einzelnen Punkten aufgebaut wird.

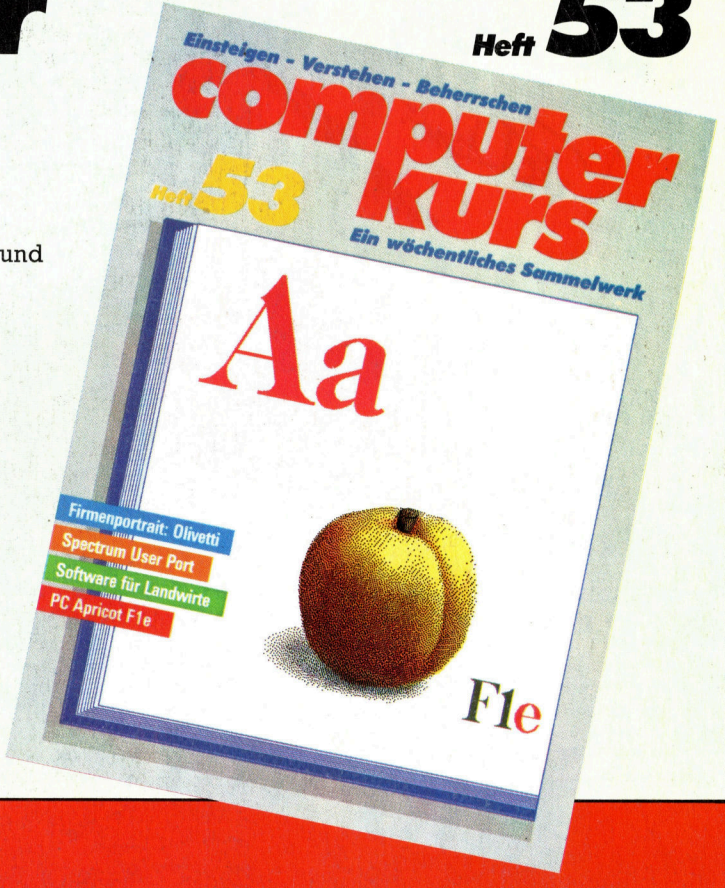
Der Druckkopf der Matrixgeräte enthält dazu eine Anzahl von Anschlagnadeln, die elektromagnetisch gegen Farbband und Papier geschleudert werden, so wie es das Zeichen erfordert. Auch Schreibmaschinen gehören im Prinzip zu den Anschlagdruckern, werden aber normalerweise nicht dazu gerechnet, weil die Ansteuerung über die eigene Tastatur erfolgt.

Bildnachweise
1429, 1441, 1442, 1443: Chris Stevens
1431, 1451, 1453: Kevin Jones
1435: Ian McKinnell
1439: Mike Clownes, Uta Brandl
1444: Marcus Wilson-Smith
1445: Tony Sleep, Computer Recognition System
1447: Caroline Clayton
U3: Steve Cross

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

computer kurs

Heft 53



Geistesblitze

Eine riesige Lücke klafft zwischen Theorie und Praxis bei der Entwicklung intelligenter Roboter. Eine Ideensammlung.

Ländliche Idylle

Kein Bauernhof ohne Computer: Hier die benötigte Software dazu.

Umbaumaßnahmen

Für den Spectrum ein Interface, das einem User Port entspricht. Ermöglicht Teilnahme am Roboter-Projekt.

Chic in Schale

Günstiger Preis und Einsatzmöglichkeiten als Personal- und Lerncomputer machen den Apricot F1e interessant.