Einsteigen - Verstehen - Beherrschen

DM 3,80 öS 30 sfr 3,80

computer kurs

Heft 64

Zwei neue Sprachsynthesizer

Begabte Interrupts

Sinclairs frische Kraft

Zukunft für CP/M

Ein wöchentliches Sammelwerk

computer kurs Heft 64

Inhalt

WIE SIE JEDE WOCHE IHR HEFT BEKOMMEN

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

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

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

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

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

Abonnement

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

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

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

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

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

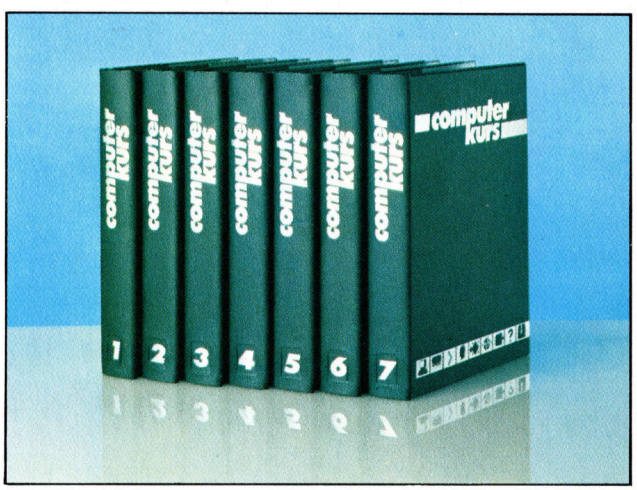
Schweiz: Der Sammelordner kostet sfr 15. Bitte wenden Sie sich an Ihren Kiosk; dort werden Sie jederzeit die gewünschten Exemplare erhalten.

INHALTSVERZEICHNIS

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

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

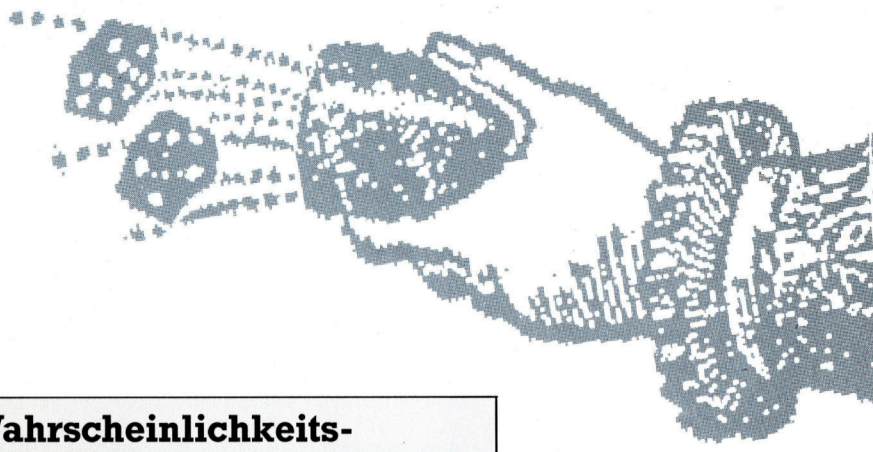
Vertrieb: Marshall Cavendish International Ltd., Heidenkampsweg 74, 2000 Hamburg 1



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

Wetten, daß . . .?

Wettspiele unterliegen den Gesetzen der Wahrscheinlichkeitsrechnung, doch beschäftigt sich kaum ein Spieler systematisch damit. Es ist aber durchaus lohnend, sich einmal in die Lage des Buchmachers zu versetzen und zu „computern".



Bei jedem Glücksspiel lassen sich die Chancen durch Wahrscheinlichkeitsrechnung abschätzen. Die gesamte Wahrscheinlichkeitsrechnung ist ein Feld voller Fallgruben und scheinbarer Widersprüche. Schon die Absicht, den Zufall präzise beschreiben zu wollen, wirkt zunächst absurd. Der Spieler von heute hat aber in seinem Computer einen fähigen Partner zum Berechnen der Chancen und kann damit bei einer gewissen Selbstbeherrschung sehr weit kommen, wenn er ein paar mathematische Grundlagen kennt. Dieser Artikel erläutert die erforderlichen Voraussetzungen und zeigt einige Punkte anhand eines Programmbeispiels auf.

Als erstes sollte sich der informierte Spieler über den Zusammenhang zwischen Chancen und Wahrscheinlichkeiten im klaren sein. Beim Wetten kann entweder für oder gegen das Eintreffen eines bestimmten Ereignisses gesetzt werden. Die Buchmacher notieren im allgemeinen Einsätze „dagegen", etwa gegen den Rennsieg eines Pferdes, wenn es nicht ausdrücklich „Odds on" (dafür) heißt. Die Übersichtlichkeit wird dadurch erschwert, daß die Gewinnchancen durch Zahlenverhältnisse wie „6:4" angegeben werden, wo 3:2 eigentlich durchaus logischer erschiene.

Zur Vereinfachung ist es daher zweckmäßig, das Chancenverhältnis einheitlich durch eine einzige Zahl auszudrücken. Hierzu lassen sich die beiden untenstehenden Formeln verwenden, wobei „f" die Chancen für das Ereignis, „a" die abschlägigen Chancen bedeuten, die der Buchmacher notiert, und „F" bzw. „A" die zugehörigen Chancenverhältnisse angeben:

$F = f / a$; $A = a / f$

Wenn die Chancen gegen ein Ereignis 100:30 stehen, ist a=100 und f=30. Das Chancenverhältnis für das Ereignis ist dann F=30/100=0,30 und das Verhältnis dagegen A=100/30=3,33. Die Chancen 7:2 wären dasselbe wie 3,5:1, und 11:4 entspräche 2,75:1 – die einheitliche Angabe erleichtert den Überblick.

Der nächste Schritt besteht in der Umsetzung der Chancen in Wahrscheinlichkeiten, und zwar mit Hilfe der Formeln

Wahrscheinlichkeitsrechnung

Die mathematische Wahrscheinlichkeitsanalyse kam durch eine Pechsträhne beim Spielen in Gang. Ein gewisser Chevalier de Mere hatte sich im 17. Jahrhundert die Taschen gefüllt, indem er immer wieder wettete, er schaffe bei viermaligem Würfeln mindestens eine Sechs.

Leider führte der Erfolg des Chevalier dazu, daß kein Mensch mehr gegen ihn setzen wollte. Deshalb behauptete er nun, er würde bei 24 Versuchen mit einem Würfelpaar mindestens eine doppelte Sechs werfen.

Das einzig bekannte Verfahren zur Ermittlung der Wahrscheinlichkeiten bestand damals darin, so oft wie möglich zu würfeln und die Ergebnisse zu protokollieren; eine zeitraubende und unzuverlässige Methode. Deshalb schrieb der Chevalier kurz vor dem Bankrott an den Mathematiker Pascal.

Die Chancen bei der ersten Wette standen mit knapp 14:13 günstig für de Mere. Bei der zweiten Wette ergibt sich jedoch ein Chancenverhältnis von rund 28:29, ein Zuschußgeschäft!

P=F/ (F+1) oder P= 1/(A+1)

P gibt die Wahrscheinlichkeit eines Ereignisses an, bei dem die Chancen F:1 für das Eintreffen oder A:1 dagegen stehen. Aus 100:30 oder 3,33:1 wird so die Wahrscheinlichkeit P = 1/4,33 = 0,231. P ist dabei immer die Wahrscheinlichkeit für das Eintreten des Ereignisses, z. B. für den Sieg eines bestimmten Rennpferds. Die Wahrscheinlichkeit Q für das Ausbleiben des Ereignisses ergibt sich einfach aus der Formel

Q = 1 – P

Es gilt also stets P+Q=1, weil die Wahrscheinlichkeiten für das Eintreten und das Ausbleiben desselben Ereignisses zusammen immer 1 ergeben müssen.

Der Buchmacher offeriert niedrigere Gewinnchancen, als es den realen Chancen entspricht, anders ausgedrückt: Er läßt die Ereignisse, gegen die gesetzt wird, wahrscheinlicher aussehen, als sie tatsächlich sind. Die Gründe dafür liegen auf der Hand – der Buchmacher muß schließlich auch seine Kosten und seinen Lebensunterhalt hereinholen. Zur Erläuterung ein fiktives Pferderennen:

Rennpferd	Gesetzt	Chancen	Auszahlg.
Flinker Schneider	DM 4000	5:6	DM 7333
Atari Sayonara	3000	13:8	7875
Acorn Junior	800	8:1	7200
Roter Apple	200	33:1	6800
	8000		

Hier sind insgesamt 8000 Mark gesetzt, die Hälfte auf den „Flinken Schneider", der Rest wie angegeben. Die Gewinnchancen sind so gewählt, daß die Verbindlichkeiten des Buchmachers stets unter dem Gesamteinsatz bleiben, egal wer das Rennen macht. Heißt der Sieger „Acorn Junior", bekommen seine Anhänger ihren Einsatz von 800 Mark und dazu das Achtfache davon ausgezahlt, also 7200 Mark. Dabei bleiben als Profit 800 Mark oder 10 % des Einsatzes für den Buchmacher. In einem ausgeglichenen Wettbuch müßten „Schneider" mit 1:1 und „Acorn" mit 9:1 notiert sein – aber wer wollte damit noch Buchmacher spielen?

Chancen und Wahrscheinlichkeit

Die Umrechnung von Chancen in Wahrscheinlichkeiten sei anhand eines Wettscheins mit vorgegebenen Gewinnchancen illustriert, wie er in England bei der Fußballwette üblich ist. Für das Spiel Leicester gegen Everton sind folgende Quoten notiert:

15/8 Leicester 5/2 Everton 1/1

Dies bedeutet, daß die Chancen gegen einen (Heim-)Sieg von Leicester 15:8 stehen, die Chancen gegen ein Unentschieden 5:2 und die Chancen gegen einen (Auswärts-)Sieg von Everton 1:1. Übrigens fand dies Spiel im Sommer 1985 tatsächlich statt, wobei Everton zwar zunächst in Führung ging, dann aber doch einen 3:1-Erfolg der Platzherren hinnehmen mußte. Zunächst sind die Chancen nach der Formel A = a/f zu normieren:

Heimsieg	15/8	= 1.875
Unentschieden	5/2	= 2.500
Auswärtssieg	1/1	= 1.000

Daraus folgen die Wahrscheinlichkeiten P=1/(A+1):

Heimsieg	1/2.875	= 0.3478
Unentschieden	1/3.5	= 0.2857
Auswärtssieg	1/2	= 0.5000
Summe		= 1.1335

Daß hier die Summe der Wahrscheinlichkeiten mit 1.1335 größer als 1 wird, ist zwar mathematisch, nicht aber juristisch bedenklich. Es bedeutet lediglich, daß der Buchmacher die Wahrscheinlichkeitssumme 1 bei diesem Spiel um 13.35 % überzogen hat. Wenn Sie sich vergegenwärtigen, daß höhere Wahrscheinlichkeiten (dafür) kleineren Chancen (dagegen) entsprechen, ist das für ihn ein einträgliches Geschäft.

Um Ihnen Schreib- (und Denk-)arbeit zu ersparen, ist unten ein Programm aufgelistet, das die Rechnerei einschließlich der Ermittlung der Buchmacher-Prozente übernimmt.

```
 10 REM ***********************************
 20 REM **WAHRSCHEINLICHKEITSBERECHNUNGEN**
 30 REM ***********************************
100 PRINT'"Wahrscheinlichkeitsrechner"
110 PRINT "Chancen als Zahlenpaar angeben;"
120 PRINT "etwa 3,1 fuer 3:1 usw."
130 PRINT "Beendigung der Eingabe mit 0,0"'
140 at%=&0102040A : REM Format
150 N=0
160 TP=0 : TA=0
190 REM **Hauptschleife**
200 REPEAT
202   @%=4
210   N=N+1
220   PRINT "Rennpferd ";N;" heißt ";
222   INPUT NAME$
225   PRINT "Chancen dagegen ";
230   INPUT A,F
240   IF A<=0 AND F<=0 THEN GOTO 310
250   P = F/(F+A)
260   A = A/F
270   TP=TP+P : REM *Wahrsch.-Summe
280   TA=TA+A : REM *Chancen-Summe
290   PRINT "Fuer Pferd Nr. ";N;", ";NAME$
295   @%=at%
296   PRINT "ist die Wahrsch.:  ";P
300   PRINT
310   UNTIL A<=0 OR F<=0
320 N=N-1
322 @%=at%
330 PRINT
333 PRINT "Einsatz-Auszahlung: ";100/TP;"%"
335 PRINT "Buchmacheraufschlag:";100*(TP-1);
    "%"
360 IF TP<1 THEN PRINT "Sie scherzen wohl!";
    CHR$7
999 END
```

Bei unserer fiktiven Wette ist zumindest eins lebensecht: Den Außenseitern werden relativ schlechte Chancen eingeräumt. Deshalb steht sich der Buchmacher meist günstiger, wenn mal ein Schuß ins Blaue trifft; setzen Sie also nicht auf unbeschriebene Blätter! Sie subventionieren damit nur die Favoriten, weil die Buchmacher dort nicht wesentlich unter gleichen Einsatz gehen können, ohne die Bieter zu vergrämen.

Dann und wann schafft es ein Buchmacher nicht, seine Verbindlichkeiten abzudecken, weil er entweder zu langsam auf plötzliche Umschwünge im Wettverhalten reagiert oder mit unrealistischen Einsätzen eröffnet hat. Aber das sind Ausnahmen; schließen Sie daraus bloß nicht, Sie könnten die Buchmacher aufs Kreuz legen. Sie können allenfalls Ihren Mitspielern das Geld aus der Tasche ziehen, wobei der Buchmacher als Mittelsmann fungiert und dabei seinen Schnitt macht. Der Traum vom leichten Geld bleibt trotz Computer ein Traum.

Für das Ausfüllen von Totozetteln sind diverse Verfahren in Gebrauch. In England ist der „8 aus 10"-Tip am populärsten; dabei kreuzt jeder Teilnehmer zehn Spiele an. Wenn die Ergebnisse bekannt sind, werden die acht besten Tips ausgewertet, um die höchstmögliche Punktzahl zu erreichen. Die Auswahl der Spiele erfolgt nach den verschiedensten Methoden, wobei immer nur eine verschwindende Chance besteht, achtmal Unentschieden zu ziehen.

```
Wahrscheinlichkeitsrechner
Chancen als Zahlenpaar angeben;
etwa 3,1 fuer 3:1 usw.
Beendigung der Eingabe mit 0,0

Rennpferd 1 heißt ?Heimsieg
Chancen dagegen ?15,8
Fuer Pferd Nr.1, Heimsieg
ist die Wahrsch. : 0.3478

Rennpferd 2 heißt ?Unentsch.
Chancen dagegen ?5,2
Fuer Pferd Nr.2, Unentsch.
ist die Wahrsch. : 0.2857

Rennpferd 3 heißt ?Auswaerts.
Chancen dagegen ?1,1
Fuer Pferd Nr.3, Auswaerts.
ist die Wahrsch. : 0.5000

Rennpferd 3 heißt ?
Chancen dagegen ?0,0

Einsatz-Auszahlung: 88.2192%
Buchmacheraufschlag:13.3540%
>
>
>RUN
```

```
Wahrscheinlichkeitsrechner
Chancen als Zahlenpaar angeben;
etwa 3,1 fuer 3:1 usw.
Beendigung der Eingabe mit 0,0

Rennpferd  1 heißt ?Slip Anchor
Chancen dagegen ?9,4
Fuer Pferd Nr. 1, Slip Anchor
ist die Wahrsch. : 0.3077

Rennpferd  2 heißt ?Law Society
Chancen dagegen ?5,1
Fuer Pferd Nr. 2, Law Society
ist die Wahrsch. : 0.1667

Rennpferd  3 heißt ?Damister
Chancen dagegen ?16,1
Fuer Pferd Nr. 3, Damister
ist die Wahrsch. : 0.0588

Rennpferd  3 heißt ?Supreme Leader
Chancen dagegen ?10,1
Fuer Pferd Nr. 4, Supreme Leader
ist die Wahrsch. : 0.0909
```

```
Rennpferd  5 heißt ?Lanfranco
Chancen dagegen ?14,1
Fuer Pferd Nr. 5, Lanfranco
ist die Wahrsch. :0.0667

Rennpferd  6 heißt ?Reach
Chancen dagegen ?33,1
Fuer Pferd Nr. 6, Reach
ist die Wahrsch. :0.0294

Rennpferd  7 heißt ?Theatrical
Chancen dagegen ?10,1
Fuer Pferd Nr. 7, Theatrical
ist die Wahrsch. :0.0909

Rennpferd  8 heißt ?Phardante
Chancen dagegen ?40,1
Fuer Pferd Nr. 8, Phardante
ist die Wahrsch. :0.0244

Rennpferd  9 heißt ?Royal Harm
Chancen dagegen ?40,1
Fuer Pferd Nr. 9, Royal Harm
ist die Wahrsch. :0.0244

Rennpferd 10 heißt ?Snow Plant
Chancen dagegen ?100,1
Fuer Pferd Nr.10, Snow Plant
ist die Wahrsch. :0.0099

Rennpferd 11 heißt ?Petowski
Chancen dagegen ?33,1
Fuer Pferd Nr.11, Petowski
ist die Wahrsch. :0.0294

Rennpferd 12 heißt ?Seurat
Chancen dagegen ?33,1
Fuer Pferd Nr.12, Seurat
ist die Wahrsch. :0.0294

Rennpferd 13 heißt ?Shadeed
Chancen dagegen ?7,2
Fuer Pferd Nr.13, Shadeed
ist die Wahrsch. :0.2222

Rennpferd 14 heißt ?Main Reason
Chancen dagegen ?200,1
Fuer Pferd Nr.14, Main Reason
ist die Wahrsch. :0.0050

Rennpferd 15 heißt ?
Chancen dagegen ?0,0

Einsatz-Auszahlung: 86.5215%
Buchmacheraufschlag:15.5781%
>
>
```

Objekt-Listing

In der letzten Folge haben wir die drei für Objekt-Manipulation erforderlichen Strukturen untersucht. Hier zeigen wir die entsprechenden Listings und nehmen einige Veränderungen vor, um „Dog and Bucket" noch lebendiger zu machen.

Unser Objekt-Manipulationsbaum kann durch Hinzufügen verschiedener Zeilen zum Quell-Listing (S. 1731) programmiert werden. Die Schlüsselzeilen sind 210, 220, 2430, 2440, 5030 bis 5090. Betrachten wir diese Teile genauer.

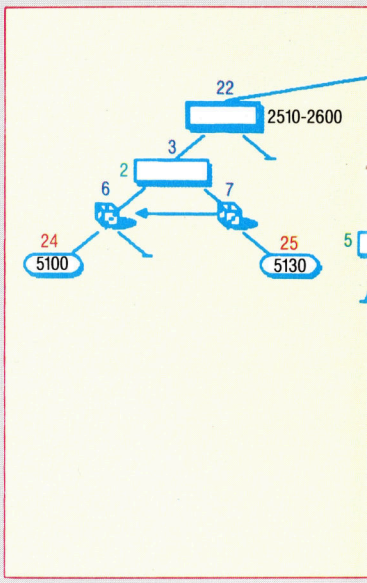
Die Zeilen 210 bis 220 setzen die für die Speicherung der Daten unseres Baumes erforderlichen Arrays. Dabei ist zu berücksichtigen, daß dieser Baum im Gegensatz zu vorher untersuchten Bäumen viele unterschiedliche Zustände überprüft, unabhängig von der Verzweigungstiefe oder Knotenzahl. Deshalb ist für jeden Knoten eine Aufzeichnung des Zustandswertes sowie der weiterführenden Wege erforderlich, abhängig davon, ob die entsprechende Bedingung richtig oder falsch ist. Das c-Array enthält die verschiedenen Zustandswerte. Jeder Knoten überprüft ein Element des Arrays. Die Zahl der zu überprüfenden Elemente wird in das k-Array gelesen.

Die Zeilen 2430 und 2440 initialisieren das c-Array. Diese Zeilen bilden eine Subroutine, die für jeden „Darsteller" aufgerufen werden muß, da der Zustandswert bei jeder Person zwangsläufig anders ist.

Zeile 5040 prüft die derzeitige Knotennummer. Handelt es sich um einen Endknoten (d. h. wenn die Zahl über 21 liegt), wird zu der Routine in den Zeilen 5070 bis 5090 verzweigt. Zeile 5050 prüft, ob der Knoten sich im Testzustand 12 befindet. Ist das der Fall, wird die Zufallszahl-Routine aufgerufen, um dem Zustand einen Wert (entweder eins oder zwei) zuzuordnen. Zeile 5060 führt daraufhin den wichtigsten Teil der Operation aus: Sie wählt aus dem t-Array die neue Knotenzahl.

Läßt man das vollständige Programm laufen, sieht man, wie das Figurenmodul funktioniert. Nach dem Prompt „Default values?" ist Y einzugeben. Der Figuren-Editor in Zeile 2350 ist mit dem Figurenmodul in der vorhandenen Form nicht voll kompatibel.

Folgende Zeilen müssen dem Initialisierungs-Modul hinzugefügt werden. Mit Zeile 190 wird ein Array t gesetzt, das zur Datenspeicherung für unsere drei Haupt-Baum-Strukturen benutzt wird. Das Array k enthält die unterschiedlichen Zustände, die in den verschiedenen Knoten überprüft werden müssen. Im Array c sind die Zustandswerte enthalten.



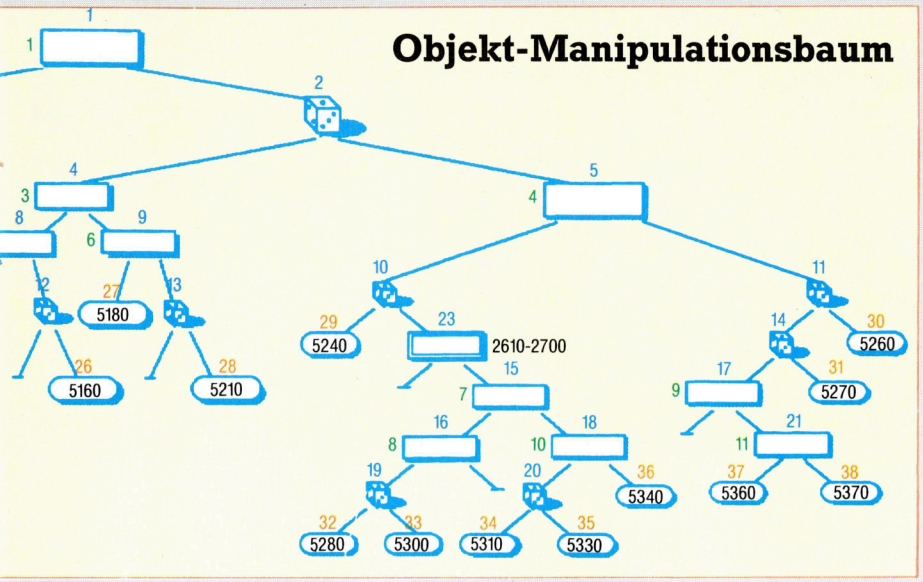

```
130 DEF FNb(y,z)=VAL(b$(y,z))
140 DEF FNc(y,z)=VAL(c$(y,z))
150 DEF FNm$(c$,d)=STR$((VAL(c$))-d)
160 DEF FNi$=b$(VAL(c$(c,3)),1)
180 REM setup trees
190 DIM t(3,25,2),k(3,30),c(25)
200 REM object tree
210 FOR n=1 TO 21: REM 21 choice nodes
220 READ k(1,n),t(1,n,2),t(1,n,1): NEXT n
```

Um den ersten Teil unseres Figurenmoduls einbeziehen zu können, müssen wir unsere Hauptprogramm-Schleife verändern. Mit den Zeilen 550 bis 800 wird nacheinander jeder „Darsteller" überprüft, um festzustellen, ob das Handlungs-Flag größer als Null ist (Zeile 560). Ist das der Fall, nimmt der Wert um eins ab, und die nächste Figur wird überprüft. Beträgt der Wert Null, wird der Steuerungsfaktor (c$(n,10)) durch Lesen der Daten in den Zeilen 6030 und 6040 zurückgesetzt. Zeile 580 prüft, ob der Standardwert Null beträgt. In diesem Fall bearbeitet das Modul den Darsteller nicht. Zeile 590 initialisiert die „Baumzustände" durch Aufruf der Subroutine in Zeile 2430.

Die Zeilen 2430 und 2440 ordnen dem c-Array verschiedene Zustandswerte zu. Das Figurenmodul ruft die Subroutinen in den Zeilen 2520, 2620 und 2720 an verschiedenen Punkten auf.

```
500 REM
510 REM test program loop
520 REM
530 GOSUB 2100: GOSUB 2150: GOSUB 2240:
PRINT: PRINT
540 REM character handler
550 FOR c=1 TO 6
560 IF FNc(c,10)>0 THEN c$(c,10)=FNm$(c$
(c,10),1): GOTO 800
570 RESTORE: FOR n=1 TO c%10+c-1: READ c
$(c,10): NEXT n: REM reset default handl
e value
580 IF FNc(c,10)=0 THEN GOTO 800
590 GOSUB 2430: GOSUB 5000: REM call obj
ect tree
800 NEXT c
810 GOSUB 4260: IF i$="" GOTO 550
820 GOSUB 2040: GOTO 530

2400 REM
2410 REM conditions
2420 REM
2430 h=FNc(c,8): i=FNc(c,3): j=FNc(c,6):
 c(1)=ABS(i>0): c(2)=ABS((FNb(j,2)=FNc(c
,2)) AND (q=1)): c(3)=ABS(b$(i,3)="y"):
c(4)=ABS(FNc(c,3)=FNc(c,6)): c(5)=ABS(b$
(j,4)="y")
2440 c(6)=ABS(i=3): c(7)=ABS(FNc(c,5)>5)
: c(8)=ABS(FNc(c,5)>2): c(9)=ABS(VAL(c$(
c,9))=1): c(10)=ABS(FNc(x,3)=0): c(11)=A
BS(FNc(h,2)=FNc(c,2)): c(12)=255
2500 RETURN
2510 REM
2520 REM check location for objects
2530 REM
2540 f=0: REM set 'found flag' to zero
2550 FOR b=1 TO 12
2560 IF FNb(b,2)=FNc(c,2) THEN f=1: b=12
2570 NEXT b
2580 IF f=1 THEN n=3: GOTO 2600
2590 n=39
2600 RETURN
2610 REM
2620 REM check for presence of owner of
object: if present, set x to character n
umber: jump back into tree
2630 REM
2640 f=0:x=0
2650 FOR m=1 TO 6
```



Verzweigungslinien
Hier wird der Objekt-Manipulationsbaum dargestellt, wobei die Wahlknoten blau, die Endknoten rot sind. Die Knoten 22 und 23 verzweigen zu Subroutinen, die die neue Knotennummer festlegen. Jeder Wahlknoten zeigt zudem (in grün) den Wert des c-Array-Elements, das den Wert des zu prüfenden Zustands enthält (Zeilen 2430 und 2440). Zufallsknoten sind durchweg dem Element 12 zugeordnet. Die Endknoten sind mit der Zeilennummer der Subroutine versehen, an die die Steuerung übergeben wird.

```
2660 IF (FNc(m,2)=FNc(c,2)) AND (FNc(m,6
)=FNc(c,3)) THEN f=1: x=m: GOSUB 2430: m
=6
2670 NEXT m
2680 IF f=1 THEN n=15: GOTO 2700
2690 n=39
2700 RETURN
2710 REM
2720 REM select object at random from ch
aracter's location
2730 REM
2740 b=0
2750 FOR s=1 TO 12
2760 IF FNb(s,2)<>FNc(c,2) THEN GOTO 278
0
2770 GOSUB 4180: IF q=1 THEN b=s: s=12
2780 NEXT s
2790 IF b=0 THEN GOTO 2750
2800 RETURN
```

Hier fügen wir eine Zufallszahl-Routine ein für die verschiedenen Zufalls-Wahl-Knoten, ferner eine kurze Routine, um die Codes von c$(n,8) und c$(n,9) auf Null zu setzen.

```
4150 REM
4160 REM random number routine
4170 REM
4180 q=INT(RND(1)*2)+1: RETURN
4190 REM
4200 REM zero character codes
4210 REM
4220 c$(c,8)="0": c$(c,9)="0": RETURN
4230 REM
4240 REM test to see if key pressed
4250 REM
4260 i$=INKEY$: RETURN
```

Diese Zeilen laufen durch den Baum (5030 bis 5060) und wählen die verschiedenen Subroutinen, die durch die Endknoten (Zeile 5090) bestimmt werden. Zu beachten ist Zeile 5080, mit der zwei Subroutinen gewählt werden, die neue Knotenwerte erzeugen und dann wieder zum Entscheidungsbaum zurückspringen.

```
5000 REM object tree routines
5010 p=0: REM zero print flag
5020 IF FNc(c,2)=r THEN p=1
5030 n=1: REM start at node 1
5040 IF n>21 GOTO 5070
5050 k=c(k(1,n))+1: IF k(1,n)=12 THEN GO
SUB 4180: k=q
5060 n=t(1,n,k): GOTO 5040
5070 IF n>=24 GOTO 5090
5080 ON (n-21) GOSUB 2540,2640: GOTO 504
0
5090 ON (n-23) GOTO 5100,5130,5160,5180,
5210,5240,5260,5270,5280,5300,5310,5330,
5340,5360,5370,5430
5100 GOSUB 2740: c$(c,3)=STR$(b)
5110 IF p=1 THEN PRINT c$(c,1);" picks u
p";b$(b,1): PRINT
5120 b$(b,2)="0": c$(c,9)="4": RETURN
5130 c$(c,3)=c$(c,6)
5140 IF p=1 THEN PRINT c$(c,1);" picks u
p";FNi$: PRINT
5150 b$(VAL(c$(c,3)),2)="0": c$(c,9)="4"
: RETURN
5160 IF p=1 THEN PRINT c$(c,1);" takes a
 sip from";FNi$: PRINT
5170 c$(c,4)=FNm$(c$(c,4),-1): RETURN
5180 GOSUB 4180: IF (p=1) AND (q=1) THEN
 PRINT c$(c,1);" is eating the sandwich.
": PRINT
5190 c$(c,4)=FNm$(c$(c,4),-2): c$(c,9)="
6": GOSUB 4180: IF q=1 THEN GOSUB 4220
5200 RETURN
5210 IF p=1 THEN PRINT c$(c,1);" takes a
 tentative bite of the pasty, groans, an
d drops it on the floor.": PRINT
5220 g=c: REM set pasty eaten flag
5230 c$(c,3)="0": c$(c,4)=FNm$(c$(c,4),1
0): b$(3,2)=c$(c,2): RETURN
5240 IF p=1 THEN PRINT c$(c,1);" puts do
wn";FNi$: PRINT
5250 b$(VAL(c$(c,3)),2)=c$(c,2): c$(c,3)
="0": RETURN
5260 c$(c,5)=FNm$(c$(c,5),-1): RETURN
5270 GOSUB 5240: RETURN
5280 IF p=1 THEN PRINT c$(c,1);" throws"
;b$(VAL(c$(c,3)),1);" at ";c$(x,1): PRIN
T
5290 c$(x,4)=FNm$(c$(x,4),1): b$(VAL(c$(
c,3)),2)=c$(c,2): c$(x,8)=STR$(c): c$(x,
9)="5": c$(c,3)="0": RETURN
5300 GOSUB 4220: RETURN
5310 IF p=1 THEN PRINT "I think I've got
 your drink, says ";c$(c,1);" to ";c$(x,
1): PRINT
5320 c$(c,8)=STR$(x): c$(c,9)="2": RETUR
N
5330 c$(c,4)=FNm$(c$(c,4),2): RETURN
5340 IF p=1 THEN PRINT c$(c,1);" gives";
FNi$;" to ";c$(x,1): PRINT
5350 c$(x,3)=c$(c,3): c$(c,3)="0": c$(x,
8)=STR$(c): c$(x,9)="1": RETURN
5360 GOSUB 4220: RETURN
5370 IF p=0 GOTO 5420
5380 IF p=1 THEN PRINT c$(c,1);" is drun
kenly thanking ";c$(VAL(c$(c,8)),1);" fo
r returning ";
5390 IF p=1 AND c$(c,7)="f" THEN PRINT "
her ";: GOTO 5410
5400 PRINT "his ";
5410 PRINT "drink": PRINT
5420 GOSUB 4220: RETURN
5430 RETURN
```

Durch Zeile 220 sind die Werte in Dreiergruppen eingebracht, die für jeden Knoten folgende Zuordnung treffen: Definition des c-Array-Elements, das die zu testende Kondition enthält; die Nummer des zu verzweigenden Knotens, wenn der Zustand wahr ist; der Knoten, zu dem zu springen ist, wenn der Zustand falsch ist.

```
6200 REM
6210 REM object tree data
6220 REM
6230 DATA 1,2,22,12,5,4,2,7,6,3,9,8,4,11
,10,12,39,24,12,6,25,5,12,39,6,13,27,12,
23,29,12,30,14,12,26,39,12,28,39,12,31,1
7,7,18,16,8,39,19,9,21,39,10,36,20,12,33
,32,12,35,34,11,38,37
```

Kooperation

In dieser Folge untersuchen wir, wie BASIC mit dem Betriebssystem des Acorn B zusammenarbeitet, und stellen USR und CALL vor.

Es wurde bereits erwähnt, daß viele OS-Routinen auch vom BASIC eingesetzt werden. So ruft der BASIC-Befehl SOUND für die Tonerzeugung OSWORD mit A=7 auf, während ENVELOPE OSWORD mit A=8 auf die gleiche Weise verwendet. Die BASIC-Abläufe lassen sich bis zu einem gewissen Grad beeinflussen, wenn man das Betriebssystem ändert – das heißt durch Vektorveränderungen und durch eigenen Code, der anstelle der bereits besprochenen OS-Routinen abläuft.

OSWRCH, OSASCII und OSNEWL werden von einer Reihe Bildschirmsteuerroutinen eingesetzt. Oft ist jedoch Maschinencode, der Grafik mit OS-Routinen erzeugt, nicht wesentlich schneller als die entsprechenden BASIC-Befehle. Daran läßt sich messen, auf welche Weise der BASIC-Interpreter OS-Routinen bei Befehlen wie MOVE, DRAW und PLOT einsetzt.

Sehen wir uns die beiden BASIC-Befehle einmal an, mit denen Sie von BASIC aus OS-Routinen aufrufen können: USR und CALL.

USR führt eine Maschinencoderoutine aus und kehrt dann mit einem Wert ins BASIC zurück, der Aufschluß über die Register A, X und Y, den Status des Übertragsflag und des Prozessor Status Registers (PRS) gibt. Der Befehl hat folgendes Format:

```
Ergebnis%=USR(Adresse%)
```

Adresse% ist die Speicheradresse der auszuführenden Routine. Bevor der Befehl eingesetzt wird, kann man Werte an A%, X% und Y% übergeben, die von der USR-Routine in die Register A, X und Y gestellt werden. Mit dem niederwertigsten Bit von C% läßt sich außerdem das Übertragsflag beeinflussen – ein gerader Wert von C% setzt es auf Null.

Das folgende Programmbeispiel zeigt den Einsatz von USR. Dabei wird eine Routine von OSBYTE aufgerufen, die im X-Register die X-Koordinate des Cursors ablegt und im Y-Register die Y-Koordinate. Der Wert von Ergebnis% zeigt diese Registerinhalte in einer codierten Form, auf die wir später noch genauer bei unserer ausführlichen Betrachtung des Betriebssystems des Acorn B eingehen.

```
10 A%=134
20 X%=0
30 Y%=0
40 LET Ergebnis%=USR(&FFF4)
50 PRINT Ergebnis%: REM zeigt eine Hexzahl
```

Die ermittelten Werte lassen sich leichter decodieren, wenn das Ergebnis im Hexadezimalformat angezeigt wird.

USR in Aktion

Mit USR läßt sich von BASIC aus ein Maschinencodeprogramm aufrufen, das einen Wert an BASIC zurückgibt. Da USR eine Funktion ist (im Gegensatz zu dem „Befehl" CALL), muß der zurückgegebene Ergebniswert entweder einer Variablen zugeordnet werden

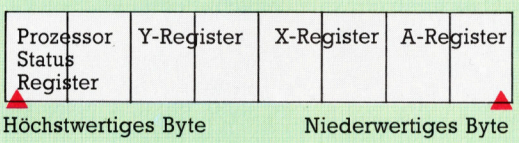
```
Ergebnis=USR(&FFF4)
```

oder Teil eines PRINT-Befehls sein:

```
PRINT USR(&FFF4)
```

Der von USR an BASIC gelieferte Ergebniswert ist eine Ganzzahl mit der Länge von vier Bytes, die den Inhalt der Register P, X, Y und A folgendermaßen speichert:

Ergebnis%=



Höchstwertiges Byte — Niederwertiges Byte

Sie erhalten den Wert eines bestimmten Registers, indem Sie mit AND Teile des Ergebnisses maskieren.

```
Ergebnis%AND&000000FF
```

maskiert alle Werte außer Register A.

Maschinencodeprogramme lassen sich auch mit CALL aufrufen. Wir haben diesen Befehl schon mehrere Male verwandt – speziell für den Aufruf von OSWORD und anderen OS-Routinen. Die gewünschte Routine wird über ihre Adresse aufgerufen. Hier ein Beispiel:

```
CALL &FFF1
```

Das BASIC des Acorn B enthält eine erweiterte Version dieses Befehls, die dem Maschinencodeprogramm Werte von BASIC-Variablen übergeben kann. Dabei stellt sich die Frage, wie dem Maschinencodeprogramm mitgeteilt werden soll, an welcher Stelle des Speichers sich die Variablen befinden. Der BASIC-Interpreter löst das Problem, indem er bei einem CALL mit Parametern wie:

```
CALL Adresse%,A%,C%
```

alle Informationen über A% und C% in einem Bereich der Speicherseite 6 ablegt, der „Parameterblock" genannt wird. Die Tabelle zeigt den Aufbau des Blocks:

Stelle	Inhaltsbeschreibung
&0600	Anzahl der Parameter
&0601	Adresse des ersten Parameters
&0602	
&0603	Typ des ersten Parameters
&0604	Adresse des zweiten Parameters
&0605	
&0606	Typ des zweiten Parameters

Selbstverständlich kann der Block beliebig viele Einträge für Adressen und Typen enthalten. Der Adreßparameter ist einfach beschrieben: An dieser Speicherstelle befindet sich das erste Byte des Parameters. Der Typparameter zeigt an, welche Art Variable der Parameter ist. Hier alle verfügbaren Typen:

Type	Beschreibung	Beispiel
0	8-Bit-Byte	CALL Adresse, ?&70
4	Ganzzahlvariable	CALL Adresse, F%
8	Reelle-Variable	CALL Adresse, G
128	String bei Adresse	CALL Adresse, $(&A00)
129	String-Variable	CALL Adresse, F$

Der letzte Eintrag – Typ 129 – fällt aus dem Rahmen heraus. Hier zeigt der Adreßeintrag des Parameterblocks nicht auf die Position des Strings im Speicher, sondern auf einen Bereich, der weitere Informationen über den String liefert – den „String Information Block". Er hat folgendes Format:

Stelle	Inhaltsbeschreibung
Im Parameterblock gespeicherte Adresse	Adresse des String
	zugeteilte Bytes
	aktuelle Stringlänge

Die ersten zwei Bytes des Information Blocks zeigen auf das erste Zeichen des Strings, und der Eintrag für „zugeteilte Bytes" gibt die maximale Länge an, die dem String zugeteilt wurde. Byte 4 enthält seine aktuelle Länge.

Mit CALL kann vom BASIC aus ein Maschinencodemodul ausgeführt werden, das im Gegensatz zu USR allerdings keinen Wert zurückgibt. Sie können jedoch eine Subroutine schreiben, die die Ergebniswerte des Maschinencodeprogramms in BASIC-Variablen speichert, und die sich nach dem Rücksprung weiterverarbeiten lassen.

Das folgende Programm zeigt nicht nur die Parameterübergabe eines CALL-Befehls, sondern stellt die Ergebniswerte auch noch in die Ganzzahlenvariable A%. Die Routine demonstriert weiterhin, wie vom Maschinencodeprogramm erzeugte Fehler die Fehlerbehandlungsroutinen von BASIC einsetzen. Das Programm wurde in BASIC II geschrieben.

```
 10 DIM C 400
 20 zero=&70
 30 FOR I%=0 TO 2 STEP 2
 40 P%=C
 50 [OPT 1%
 60 .code LDA&600
 62 CMP#1
 64 BNE error
 70 LDA&603
 80 CMP#4
 90 BNE error2
100 LDA&601:STA&70
110 LDA&602:STA&71
120 LDY#0
130 LDA&02:STA(zero),Y:INY
140 LDA&03:STA(zero),Y
160 RTS
170 .error BRK
180 EQUB 254
190 EQUS "Parameterzahl nicht korrekt"
200 EQUB 00
210 .error2 BRK
220 EQUB 253
230 EQUS "Falscher Typ"
240 EQUB 00
250 ]:NEXT
260 A%=0:CALL code,A%
270 PRINT "Ende der BASIC-Variablen ist";A%
```

Das Programm gibt die Endadresse der BASIC-Variablen in der Ganzzahlenvariablen zurück, die auch als Parameter eingesetzt wurde (und zeigt damit, wie CALL Werte an BASIC weitergibt). Die Zeilen 60 bis 64 prüfen, ob nur ein Parameter übergeben wurde. Ist diese Bedingung nicht erfüllt, erscheint die entsprechende Fehlermeldung. Zeile 70 bis 90 testen, ob der als Parameter übergebene Wert eine Ganzzahl ist. Falls nicht, wird auch hier eine Fehlermeldung angezeigt. Die Zeilen 100 bis 160 übertragen die Daten der Adressen 2 und 3 in die niederwertigsten Bytes der als Parameter übergebenen Ganzzahlvariablen, während 170 bis 240 auf weitere Fehler eingehen.

Zeile 260 schließlich enthält den Befehl CALL. A% wird zuvor auf Null gesetzt, da das Maschinencodeprogramm die höherwertigen Bytes der Ganzzahlvariablen nicht anspricht. Nach Ablauf der Routine gibt HIMEM-A% die Zahl der Bytes an, die bei Abzug des BASIC-Programms und der Variablen noch frei sind und für Programme zur Verfügung stehen.

Sinclairs neue Kraft

In den letzten zwei Abschnitten des Selbstbau-Kurses wurden die Acorn- und Commodore-Programme für den Robot-Arm behandelt. Die hier gezeigte Sinclair-Version arbeitet in Verbindung mit dem selbstgebauten Spectrum-Interface.

Die Software für den Robot-Arm muß in Maschinensprache geschrieben sein, weil die Servomotoren in kurzen Zeitabständen mit Steuerimpulsen versorgt werden müssen. Eine Ansteuerung im 50stel-Sekundentakt läßt sich mit BASIC nicht erreichen. Wie in den Programmen für den Commodore und Acorn B werden die erforderlichen Maschinenbefehle mit Interrupts erzeugt.

Die beste Interrupt-Betriebsart für den Z80 sind IM2-Modus-Interrupts. Wenn dieser Interrupt-Modus angewählt wird, ermittelt der Prozessor bei jeder Auffrischung des Bildschirminhalts aus zwei verschiedenen Quellen die Startadresse des Interrupt-Programms: Das höherwertige Byte der Adresse steht im Register I, der Datenbus liefert das niederwertige Byte.

Dieses System soll beim Z80 der Kommunikation mit Peripheriegeräten dienen, die dem Prozessor über den Datenbus das niederwertige Byte angeben können. In unserem Fall kennen wir den Zustand des Datenbusses jedoch nicht und müssen alle 256 Möglichkeiten berücksichtigen. Dafür belegt das Programm eine Memory Page (Speicherseite) für die Startadresse des Programms, die mit #FBs vollgeschrieben wird. Ist das Register I gesetzt, beginnt die Programmausführung nach einem IM2-Interrupt bei der Startadresse #FBFB.

Der Rest des Maschinenprogramms arbeitet mit den gleichen Methoden wie die bereits beschriebenen Programmversionen. Es wird eine Impulskette zum E/A-Port 31 geschickt, der mit dem Spectrum-Interface gekoppelt ist. Auch das übergeordnete BASIC-Steuerprogramm ähnelt der Acorn- bzw. Commodore-Version.

Robot-Arm Steuerprogramm für den Spectrum

BASIC-Loader

```
 10 REM ********************
 15 REM ********************
 17 REM **                  **
 20 REM **  spectrum arm    **
 25 REM **   controller     **
 30 REM **                  **
 40 REM ********************
 50 REM ********************
 60:
 70 CLEAR 30000
 80 LET sa=64017: REM mc start wedge
 82 LET np=64008: REM newpos start address
 84 LET dl=64016: REM delay factor address
 86 LET en=64148: REM wedge off
 88 LET sm=64151: REM smooth move start addr
 90 LOAD "ARM.HEX"CODE sa
100 GO SUB 1000: REM set up
110 CLS
120 PRINT AT 6,2;"would you like to:"
130 PRINT AT 9,2;"1..program new sequence"
140 PRINT AT 10,2;"2..add moves to a program"
150 PRINT AT 11,2;"3..load a file"
160 PRINT AT 12,2;"4..leave program"
170 LET g$=INKEY$: IF g$="" THEN  GO TO 170
180 IF g$="1" THEN  LET c=1: LET lm=0
190 IF g$="1" OR g$="2" THEN  GO SUB 2000: GO SUB
3000: GO SUB 4000: GO SUB 5000
200 IF g$="3" THEN  GO SUB 6000
210 IF g$="4" THEN  CLS : RANDOMIZE USR en: STOP
220 GO TO 110
230:
1000 REM **** set up ****
1010 LET c=1: LET ns=4: REM no of servos
1020 LET lm=0: LET oc=0: LET df=10: REM delay fact
or
1030  LET mc=100: REM max count
1040 DIM r(mc,ns): REM key positions array
1050 DIM l(1): REM lastmax data array
1060 FOR i=0 TO 3: POKE np+i,0: NEXT i
1070 RANDOMIZE USR sa
1090 RETURN
1099:
2000 REM **** inform ****
2010 CLS
2020 PRINT AT 3,2;"please use"
2030 PRINT AT 4,2;"n/m...for 1/r"
2040 PRINT AT 5,2;"p/l...for 1st arm u/d"
2050 PRINT AT 6,2;"a/z...for 2nd arm u/d"
2060 PRINT AT 7,2;"x/c...for grab open/close"
2070 PRINT AT 8,2;"s.....save a position"
2080 PRINT AT 9,2;"q.....return to menu"
2090 PRINT AT 10,2;"r.....move to saved position"
2100 PRINT AT 11,2;"v/b...next and back count"
2110 PRINT AT 12,2;"e.....set new count"
2120 PRINT AT 13,2;"i/d...to inc/dec speed"
2130 PRINT AT 1,2;"count=";c;" ";
2135 FOR i=1 TO 4: PRINT r(c,i);" ";: NEXT i: PRIN
T
2140 RETURN
2190:
3000 REM **** program arm ****
3010 POKE dl,1
3020 LET dx=5
3030 LET a$=INKEY$: IF a$="" THEN  GO TO 3030
3035 IF a$="n" THEN  LET p=PEEK (np)+dx: IF p<256
THEN  POKE np,p
3040 IF a$="m" THEN  LET p=PEEK (np)-dx: IF p>0 TH
EN  POKE np,p
3050 IF a$="p" THEN  LET p=PEEK (np+1)+dx: IF p<25
6 THEN  POKE np+1,p
3060 IF a$="l" THEN  LET p=PEEK (np+1)-dx: IF p>0
THEN  POKE np+1,p
3065 IF a$="a" THEN  LET p=PEEK (np+2)+dx: IF p<25
6 THEN  POKE np+2,p
3070 IF a$="z" THEN  LET p=PEEK (np+2)-dx: IF p>0
THEN  POKE np+2,p
3080 IF a$="x" THEN  LET p=PEEK (np+3)+3*dx: IF p<
256 THEN  POKE np+3,p
3090 IF a$="c" THEN  LET p=PEEK (np+3)-3*dx: IF p>
0 THEN  POKE np+3,p
3100 IF a$="r" THEN  FOR i=1 TO 4: POKE np+i-1,r(c
,i): NEXT i
3110 IF a$="v" THEN  LET c=c+1: IF c>mc THEN  LET
c=1
3120 IF a$="b" THEN  LET c=c-1: IF c<1 THEN  LET c
=mc
3130 IF a$="e" THEN  PRINT AT 2,20;"count value":
INPUT c
3140 IF a$="s" THEN  FOR i=1 TO 4: LET r(c,i)=PEEK
 (np+i-1): NEXT i: LET c=c+1
3145 IF a$="i" THEN  LET dx=dx+1
3147 IF a$="d" THEN  LET dx=dx-1: IF dx<1 THEN  LE
T dx=1
3150 IF c>lm THEN  LET lm=c
3160 IF oc<>c AND c>1 THEN  PRINT AT 1,2;"count=";
c-1;" ";: FOR i=1 TO 4: PRINT r(c-1,i);" ";: NEXT
i: PRINT
3170 LET oc=c
3190 IF a$<>"q" THEN  GO TO 3030
3200 RETURN
3900:
4000 REM **** replay sequence ****
4010 CLS
4020 PRINT AT 12,2;"replay sequence y/n,r repeats"
```

```
4030 LET a$=INKEY$: IF a$<>"y" AND a$<>"n" AND a$<>"r" THEN  GO TO 4030
4040 IF a$="n" THEN  RETURN
4050 IF a$="r" THEN  PRINT AT 14,1;"delay factor 1-255";: INPUT df
4060 IF df<1 OR df>255 THEN  GO TO 4050
4070 POKE dl,df: REM set delay register
4080 FOR i=1 TO lm
4090 PRINT AT 1,2;"no in sequence=";i;" ";
4100 FOR s=1 TO 4
4110 POKE np+s-1,r(i,s)
4120 NEXT s: NEXT i
4140 GO TO 4010
4990:
5000 REM **** save a file ****
5010 CLS
5020 PRINT AT 12,2;"save sequence (y/n)?"
5030 LET g$=INKEY$: IF g$<>"y" AND g$<>"n" THEN  GO TO 5030
5040 IF g$="n" THEN  RETURN
5050 LET l(1)=lm
5060 INPUT "filename";f$
5065 PRINT "press play and record"
5067 RANDOMIZE USR en: REM wedge off
5070 SAVE f$+".lm" DATA l()
5080 SAVE f$+".r" DATA r()
5085 RANDOMIZE USR sa: REM wedge back on
5090 RETURN
5900:
6000 REM **** load a file ****
6010 CLS
6020 PRINT AT 12,2;"load a file (y/n)"
6030 LET g$=INKEY$: IF g$<>"y" AND g$<>"n" THEN  GO TO 6030
6040 IF g$="n" THEN  RETURN
6045 INPUT "filename";f$
6050 FOR i=1 TO mc
6060 FOR j=1 TO ns
6070 LET r(i,j)=0
6080 NEXT j: NEXT i
6090 RANDOMIZE USR en: REM wedge off
6100 LOAD f$+".lm" DATA l()
6110 LET lm=l(1): LET c=lm: LET oc=0
6120 LOAD f$+".r" DATA r()
6125 RANDOMIZE USR sa: REM wedge back on
6130 RETURN
```

Assembler Listing

```
                1000 ;SPECTRUM ARM CONTROLLER
                1010
001F            1020 PORT:  EQU  31
F900            1030        ORG  #F900
F900            1040 MOTTAB DEFS 256
FA00            1050 ANGTAB DEFS 8
FA08            1060 NEWPOS DEFS 8
FA10            1070 DELAY: DEFS 1
                1080
                1090 ; SET  UP VECTOR TABLE
FA11 2100FC     1100 INIT:  LD   HL,#FC00
FA14 01FB00     1110        LD   BC,#00FB
FA17 71         1120 LOOP1: LD   (HL),C
FA18 23         1130        INC  HL
FA19 10FC       1140        DJNZ LOOP1
FA1B 71         1150        LD   (HL),C
FA1C 3EFC       1170        LD   A,#FC
FA1E ED47       1180        LD   I,A
                1190 ;INIT TABLES TO FF
                1200 ;
FA20 2100F9     1210        LD   HL,MOTTAB
FA23 3EFF       1220        LD   A,#FF
FA25 06FF       1230        LD   B,#FF
FA27 77         1240 LOOP2: LD   (HL),A
FA28 2C         1250        INC  L
FA29 10FC       1260        DJNZ LOOP2
FA2B 2100FA     1270        LD   HL,ANGTAB
FA2E 0610       1280        LD   B,16
FA30 77         1290 LOOP2A LD   (HL),A
FA31 2C         1300        INC  L
FA32 10FC       1305        DJNZ LOOP2A
FA34 ED5E       1310        IM   2
FA36 C9         1320        RET
                1380 ; INTERRUPT HANDLER
                1390
FA37 F3         1410 HANDLE DI
FA38 F5         1420        PUSH AF
FA39 C5         1430        PUSH BC
FA3A D5         1440        PUSH DE
FA3B E5         1450        PUSH HL
FA3C FF         1460        RST  #38 ;NML ROUTINE
FA3D F3         1470        DI
FA3E CD47FA     1480        CALL EVENT ;OUR ROUTINE
FA41 E1         1490        POP  HL
FA42 D1         1500        POP  DE
FA43 C1         1510        POP  BC
FA44 F1         1520        POP  AF
FA45 FB         1530        EI
FA46 C9         1540        RET
                1550
                1560 ;
                1570 ;++++ EVENT ROUTINE ++++
                1580 ;
FA47 0608       1590 EVENT: LD   B,8
FA49 3E7F       1600        LD   A,#7F
FA4B 16F0       1610        LD   D,#F0
FA4D 2158FA     1620        LD   HL,FIX+2
FA50 DD2100FA   1630        LD   IX,ANGTAB
FA54 3607       1635        LD   (HL),7
FA56 DD5E07     1640 FIX:   LD   E,(IX+7)
FA59 12         1650        LD   (DE),A
FA5A 0F         1660        RRCA
FA5B 35         1670        DEC  (HL)
FA5C 10F8       1680        DJNZ FIX
                1690 ;
                1740 ;PREP MOTTAB FOR PORTSEND
                1750 ;
FA5E 0600       1760        LD   B,0
FA60 3EFF       1770        LD   A,#FF
FA62 2100F9     1780        LD   HL,MOTTAB
FA65 A6         1790 LOOP3: AND  (HL)
FA66 77         1800        LD   (HL),A
FA67 2C         1810        INC  L
FA68 10FB       1820        DJNZ LOOP3
                1830 ;
                1840 ;++++ START PULSES ++++
                1850 ;
FA6A 3EFF       1860        LD   A,#FF
FA6C D31F       1870        OUT  (PORT),A
                1880 ;
                1885 ;++++ CALL DELAY ++++
FA6E D5         1887        PUSH DE
FA6F 1E04       1889        LD   E,4
FA71 16FF       1890        LD   D,#FF
FA73 CD8BFA     1892        CALL OLOOP
FA76 D1         1894        POP  DE
                1920 ;
                1930 ;++++ SEND MOTTAB TO PORT ++++
                1940 ;
FA77 0E1F       1950        LD   C,PORT
FA79 06FF       1960        LD   B,#FF
FA7B 2E00       1970        LD   L,00
FA7D EDB3       1980        OTIR
                1990 ;
                2000 ;++++ RESTORE MOTTAB TO FF +++
+
FA7F 3EFF       2010        LD   A,#FF
FA81 0600       2020        LD   B,0
FA83 2100F9     2030        LD   HL,MOTTAB
FA86 77         2040 LOOP4: LD   (HL),A
FA87 2C         2050        INC  L
FA88 10FC       2060        DJNZ LOOP4
FA8A C9         2070        RET
                2080 ;++++ DELAY LOOP ++++
                2090 ;
FA8B 1D         2100 OLOOP: DEC  E
FA8C C8         2110        RET  Z
FA8D 15         2120 ILOOP: DEC  D
FA8E CA8BFA     2130        JP   Z, OLOOP
FA91 C38DFA     2140        JP   ILOOP
                2200 ;
                2210 ;++++   RESTORE IM 1 ++++
                2220
FA94 ED56       2230 REST:  IM   1
FA96 C9         2240        RET
                2250 ;
                2340 ;++++ SMOOTH MOVER ++++
                2350 ;
FA97 0E04       2360 START: LD   C,4
FA99 0604       2370        LD   B,4   ;SET UP COUN
TERS
FA9B 2103FA     2380        LD   HL,ANGTAB+3
FA9E 110BFA     2390        LD   DE,NEWPOS+3
FAA1 7E         2400 NEXMOT LD   A,(HL)
FAA2 EB         2410        EX   DE,HL
FAA3 BE         2420        CP   (HL)
FAA4 EB         2430        EX   DE,HL
FAA5 280A       2440        JR   Z,MOVED ;SAME
FAA7 3804       2450        JR   C,ADD   ;BORROW
FAA9 35         2460        DEC  (HL)
FAAA C3B2FA     2470        JP   NEXT
FAAD 34         2480 ADD:   INC  (HL)
FAAE C3B2FA     2490        JP   NEXT
FAB1 0D         2500 MOVED: DEC  C
FAB2 2B         2510 NEXT:  DEC  HL
FAB3 1B         2520        DEC  DE
FAB4 10EB       2530        DJNZ NEXMOT
FAB6 F5         2540        PUSH AF
FAB7 D5         2550        PUSH DE
FAB8 ED5B10FA   2560        LD   DE,(DELAY)
FABC 16FF       2570        LD   D,#FF
FABE CD8BFA     2580        CALL OLOOP ;CALL DELAY
FAC1 D1         2590        POP  DE
FAC2 F1         2600        POP  AF
FAC3 20D2       2610        JR   NZ,START
FAC5 C9         2620        RET
                3000 ;++++ SET UP HANDLER ++++
                3010 ;++++ JUMP ADDRESS   ++++
                3020 ;
FBFB            3030        ORG  #FBFB
FBFB F3         3040        DI
FBFC C337FA     3050        JP   HANDLE

Pass 2 errors: 00

ADD     FAAD    ANGTAB FA00
DELAY   FA10    EVENT  FA47
FIX     FA56    HANDLE FA37
ILOOP   FA8D    INIT   FA11
LOOP1   FA17    LOOP2  FA27
LOOP2A  FA30    LOOP3  FA65
LOOP4   FA86    MOTTAB F900
MOVED   FAB1    NEWPOS FA08
NEXMOT  FAA1    NEXT   FAB2
OLOOP   FA8B    PORT   001F
REST    FA94    START  FA97
```

Piraten voraus!

In diesem Teil unseres Programmprojekts befassen wir uns mit den restlichen vier Hauptereignissen. Dabei kann beispielsweise das Schiff von Piraten angegriffen werden, durch einen Sturm vom Kurs abkommen oder das Ruder des Schiffes brechen.

Durch die Unterroutine bei Zeile 6500, die eine Zufallszahl generiert, wird das Programm per Zufallsfaktor zur Hauptereignis-Unterroutine verzweigt. Um auch die neuen Möglichkeiten ansprechen zu können, müssen die ersten Zeilennummern der vier Ereignis-Routinen in dieser Zeile eingetragen werden. Zeile 6510 sollte dann wie folgt aussehen:

```
6510 ON X GOSUB 6530, 6700, 6800, 6900,
7000, 7050
```

Die dritte Nummer ruft eine Unterroutine auf, in der das Schiff von Piraten angegriffen wird. Das Programm überprüft, ob das Ereignis bereits einmal eingetreten ist, indem das Flag M(3) untersucht wird. Ist M(3)=1, erfolgt der Rücksprung zum Hauptprogramm. Ansonsten wird der Programmlauf fortgesetzt und M(3) dann in Zeile 6818 auf 1 gesetzt.

Theoretisch ist es möglich, daß die gesamte Mannschaft tot ist, und somit wäre ein Piratenangriff irrelevant. Daher wird eine Schleife durchlaufen, mit der das Stärke-/Typ-Array TS(T,2) untersucht wird. Dabei werden alle Mitglieder mit den Werten 0 oder -999 gezählt. Diese Werte werden in X addiert. Ist X=16 (alle sind tot), erfolgt Rücksprung zum Hauptprogramm.

Ist das Ergebnis der Bedingungen positiv, greifen die Piraten an, wobei einige Männer getötet werden. Die Anzahl der Opfer ist abhängig davon, ob die Mannschaft Waffen zur Verteidigung hat. Die Anzahl der Getöteten wird in K gezählt, wobei die Minimalanzahl 2 beträgt. In Zeile 6825 wird K auf diesen Wert gesetzt sowie das entsprechende Vorrats-Array OA(2) darauf untersucht, ob Waffen an Bord sind. Ist der Wert von OA(2) 0 oder -999, gibt es keine Waffen, und K wird auf 4 gesetzt. Stehen Waffen zur Verfügung, wird in S$ „IN THE SPITE OF YOUR GUNS“ abgelegt. Wurde K auf 4 gesetzt, wird in S$ „YOU HAVE NO GUNS“ abgelegt.

Abschließend wird zwischen den Zeilen 6836 und 6845 eine weitere Schleife gestartet, mit der die Anzahl an Männern gelöscht wird. In Zeile 6835 wird X auf 0 gesetzt, um den Vorgang mitzuzählen. Ist der Stärkewert eines Elementes 0 oder -999, wird das nächste Element untersucht. Ansonsten wird durch Zeile 6840 der Wert auf -999 gesetzt und der Wert von X um 1 erhöht. Ist der Wert von X=K, wird durch Zeile 6842 der Schleifenzähler T auf 16 gesetzt und die Schleife beendet. Sind nur noch weniger Männer vorhanden als durch K angegeben, wird die Schleife nach 16 Durchläufen verlassen. Die Zeilen 6828 oder 6830 geben die Anzahl der getöteten Männer an.

Das Ruder bricht

Durch das nächste Ereignis wird das Ruder beschädigt. Haben Sie einen Mechaniker angeheuert (und ist er noch am Leben), ist der Schaden ohne große Verzögerung schnell behoben. Ist dies nicht der Fall, verlängert sich die Reisedauer. Die entsprechende Unterroutine beginnt bei Zeile 6900 (die vierte Zeilennummer hinter ON X GOSUB in Zeile 6510). Auch hier wird überprüft, ob das Ereignis bereits eingetreten ist oder noch nicht.

Der Wert von X wird auf 4 gesetzt, was der Anzahl an Wochen entspricht, um die sich die Fahrt verlängert, wenn kein Mechaniker an Bord ist. Danach wird TS(,) nach einem lebenden Mechaniker durchsucht. Zwischen den Zeilen 6930 bis 6938 wird eine Schleife initialisiert, die das Typ-Array TS(,) nach dem Wert 3 durchsucht, wobei der Stärkewert nicht 0 oder -999 sein darf. Sind alle Bedingungen positiv, wird X von 4 auf 1 gesetzt, und die Fahrtzeit verlängert sich nur um eine Woche.

Auf dem Bildschirm erscheint entweder die Meldung: „ALTHOUGH YOU HAVE A MECHANIC“ oder, wenn X=4, „YOU HAVE NO MECHANIC AND“. Danach wird die zusätzliche Reisedauer dargestellt und in Zeile 6965 zu der Gesamtfahrzeit EW addiert. Ein Tastendruck setzt das Spiel fort. Die Variable X hat zwei Aufgaben: Sie beinhaltet die zusätzliche Wochenzahl, und sie dient als Flag, das anzeigt, ob ein Mechaniker an Bord ist oder nicht.

Das fünfte Hauptereignis in der Unterroutine ab Zeile 7000 ist ein Sturm. Durch ihn kommt das Schiff vom Kurs ab, wodurch sich die Reisezeit verlängert. Ist ein Navigator an Bord, wird das Schiff schnell wieder auf den richtigen Kurs gebracht. Ohne Navigator dauert die Fahrt länger.

Diese Unterroutine ähnelt strukturell der Ruder-Routine und nutzt auch einzelne Programmteile dieser Routine.

Wieder wird überprüft, ob das Ereignis schon einmal eingetreten ist, und bei negativem Ergebnis der Wert von M(5) auf 1 gesetzt. Die zusätzliche Reisezeit beträgt zwei Wochen und wird in X gespeichert, wenn kein Navigator an Bord ist. Eine Schleife zwischen den Zeilen 7030 und 7038 durchsucht das Mannschafts-Array nach einem Navigator (=4 im Typ-Array) und einem Stärkewert größer als 0 (im Stärke-Array). Ist die Suche erfolgreich, werden X auf 1 und T auf 16 gesetzt, so daß das Programm die Schleife verläßt.

S$ wird, entsprechend dem Wert von X, die Meldung „ALTHOUGH YOU HAVE A NAVIGATOR" oder „YOU HAVE NO NAVIGATOR AND" zugeteilt. Anschließend verzweigt das Programm zu Zeile 6950 (Unterroutine für das Ruder), um die Programmteile zu verwenden, die für die zusätzliche Reisedauer, die Erhöhung von EW, die Reisezeit mit X und den Rücksprung zum Hauptprogramm zuständig sind.

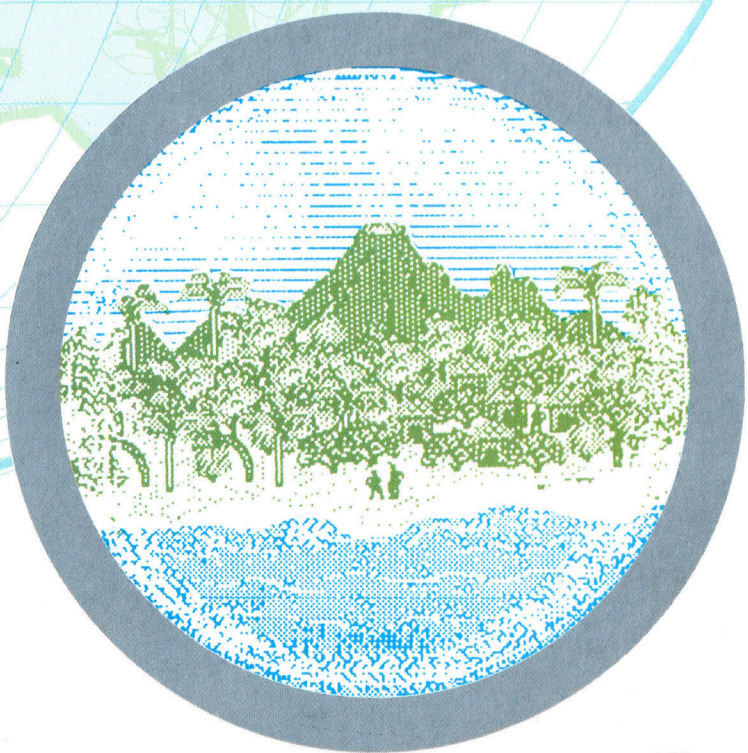
Das letzte Hauptereignis ist das Sichten einer Insel (Unterroutine ab Zeile 7050). Dies ist die letzte Zeilennummer in der ON X GOSUB-Anweisung in Zeile 6510. Da die Insel nicht auf dem Kurs liegt, kostet ein Besuch zusätzliche Zeit. Durch den Besuch besteht jedoch die Chance, die Vorräte aufzufrischen. Der Spieler muß sich also entscheiden, ob er den Kurs ändern will oder nicht. Besuchen Sie die Insel, ist der Erfolg dieser Expedition davon abhängig, ob und wie man dem Albatros begegnet ist. Hat der Spieler den Albatros geschossen, werden Sie keinen Erfolg haben!

Land in Sicht

Es erfolgt die übliche Kontrolle, ob das Ereignis schon eingetreten ist. Danach wird der Spieler informiert, daß eine Insel in Sicht ist, und daß die Möglichkeit zum Auffrischen der Vorräte besteht. Nach der Aufforderung zur Entscheidung wartet das Programm (Zeile 7082) auf eine Antwort. Antwortet der Spieler mit N, erfolgt durch Zeile 7086 Rücksprung zum Hauptprogramm. Bei positiver Antwort erreicht der Spieler die Insel bei Zeile 7100.

Jetzt überprüft das Programm in Zeile 7106, ob der Albatros geschossen wurde. Trifft das nicht zu, bleibt der Inhalt von B$ (in Zeile 48 auf N gesetzt) unverändert, und der Programmlauf wird mit Zeile 7110 fortgesetzt. Hat der Spieler den Vogel getötet, wird B$ mit Zeile 6162 der Albatros-Unterroutine auf Y geändert. In diesem Fall ist die Insel unfruchtbar, das Wasser vergiftet, und der Spieler wird an den „Mord" erinnert. Anschließend verzweigt das Programm zu Zeile 7130, wo die Reisedauer erhöht wird.

Wurde der Vogel nicht geschossen, initialisiert das Programm in Zeile 7110 eine Schleife, die zum Aufnehmen der Vorräte von 1 bis 4 (für die vier Vorratsarten) dient. Die jeweiligen Mengen werden zufällig bestimmt. In Zeile 7112 wird ein Zufallswert generiert. Ist er kleiner 0.25, erfolgt der nächste Schleifendurchlauf. In Zeile 7115 wird eine Zufallszahl zwischen 5 und 14 generiert und in X abgelegt. Diese Menge wird durch Zeile 7120, gefolgt von der Einheit des jeweiligen Vorrates (Kilo oder Barrel), ausgegeben. Wurde ein Vorrat in der aktuellen Woche über Bord gespült, steht der entsprechende Wert im Vorrats-Array auf -999. In diesem Fall werden diese Werte durch Zeile 7122 auf 0 gesetzt, damit die neuen Vorräte addiert werden können (Zeile 7125). Anschließend wird die Reisedauer um eine oder zwei Wochen verlängert. Zeile 7135 entscheidet per Zufall, welche Erhöhung vorgenommen werden soll, der Spieler wird informiert und die zusätzliche Zeit in Zeile 7140 zu EW addiert.



Modul neun: weitere Hauptereignisse

Initialisiere Flags

```
48 A$="N":B$="N"
```

Piraten-Unterroutine

```
6800 REM PIRATES
6805 IFM(3)=1THENRETURN
6810 X=0
6812 FORT=1TO16
6814 IFTS(T,2)=0ORTS(T,2)=-999THENX=X+1
6815 NEXT
6816 IFX=16 THEN RETURN
6818 M(3)=1
6820 PRINTCHR$(147)
6822 S$="  PIRATES ATTACK THE SHIP!*":GOSUB9100
6824 PRINT:GOSUB9200
6825 K=2
6826 IFOA(2)=0OROA(2)=-999THENK=4
6828 S$="IN SPITE OF YOUR GUNS*"
6830 IFK=4THENS$="YOU HAVE NO GUNS*"
6832 GOSUB9100
6835 X=0
6836 FORT=1TO16
6838 IFTS(T,2)=0ORTS(T,2)=-999THEN6845
6840 X=X+1:TS(T,2)=-999
6842 IFX=KTHENT=16
6845 NEXT
6850 PRINTX;
6855 S$="OF THE CREW IS KILLED*"
6856 IFX>1THENS$="OF THE CREW ARE KILLED*"
6860 GOSUB9100
6865 PRINT:GOSUB9200
6890 S$=K$:GOSUB9100
6895 GETI$:IFI$=""THEN6895
6899 RETURN
```

Ruder-Unterroutine

```
6900 REM RUDDER
6905 IFM(4)=1THENRETURN
6910 PRINTCHR$(147)
6915 M(4)=1
6920 S$="TROUBLE WITH THE RUDDER!*":GOSUB9100
6925 PRINT:GOSUB9200
6928 X=4
6930 FORT=1TO16
6935 IFTS(T,1)=3ANDTS(T,2)<>0ANDTS(T,2)<>-999
THENX =1 :T=16
6938 NEXT
6940 S$="ALTHOUGH YOU HAVE A MECHANIC*"
6945 IFX=4THENS$="YOU HAVE NO MECHANIC AND*"
6950 GOSUB9100
6955 S$="YOUR JOURNEY WILL TAKE*":GOSUB9100
6960 PRINTX;"WEEKS LONGER"
6965 EW=EW+X
6967 PRINT:GOSUB9200
6969 S$=K$:GOSUB9100
6970 GETI$:IFI$=""THEN6970
6975 RETURN
```

Sturm-Unterroutine

```
7000 REM STORM
7005 IFM(5)=1THENRETURN
7010 PRINTCHR$(147)
7015 M(5)=1
7020 S$="YOU ARE BLOWN OFF COURSE*":GOSUB9100
7022 S$="IN A STORM!*":GOSUB9100
7025 PRINT:GOSUB9200
7028 X=2
7030 FORT=1TO16
7035 IFTS(T,1)=4ANDTS(T,2)<>0ANDTS(T,2)<>-999THENX
=1:T=16
7038 NEXT
7040 S$="ALTHOUGH YOU HAVE A NAVIGATOR*"
7045 IFX=2THENS$="YOU HAVE NO NAVIGATOR AND*"
7049 GOTO6950
```

Insel-Unterroutine

```
7050 REM ISLAND
7055 IFM(6)=1THENRETURN
7060 PRINTCHR$(147)
7065 M(6)=1
7070 S$="YOUR CHARTS SHOW AN ISLAND*":GOSUB9100
7071 S$="WHERE YOU MAY BE ABLE TO*":GOSUB9100
7072 S$="RE-STOCK YOUR PROVISIONS*":GOSUB9100
7073 S$="BUT IF YOU GO THERE*":GOSUB9100
7074 S$="IT WILL ADD TIME TO YOUR JOURNEY*":GOSUB9
100
7075 PRINT:GOSUB9200
7080 S$="DO YOU WANT TO GO THERE*":GOSUB9100
7082 INPUTI$:I$=LEFT$(I$,1)
7084 IFI$<>"Y"ANDI$<>"N"THEN7082
7086 PRINT:GOSUB9200
7090 IFI$="N"THEN7145
7100 S$="YOU REACH THE ISLAND*":GOSUB9100
7105 S$="AND OBTAIN:*":GOSUB9100
7106 IFB$="N"THEN7110
7107 PRINT:GOSUB9200
7108 PRINT"NOTHING!":GOSUB9200
7109 S$="(REMEMBER THE ALBATROSS!)":GOSUB9100:GOTO
7130
7110 FORT=1TO4
7112 IFRND(1)<.25THEN7129
7115 X=INT(RND(1)*10)+5
7120 PRINTX;U$(T);"S OF";P$(T)
7122 IFPA(T)=-999THENPA(T)=0
7125 PA(T)=PA(T)+X
7129 NEXT
7130 S$="BUT THE JOURNEY WILL NOW TAKE*":GOSUB9100

7135 X=INT(RND(1)*2)+1
7139 PRINTX;:S$="WEEKS LONGER*":GOSUB9100
7140 EW=EW+X
7145 PRINT:GOSUB9200
7150 S$=K$:GOSUB9100
7155 GETI$:IFI$=""THEN7155
7159 RETURN
```

BASIC-Dialekte

Spectrum
Ändern Sie das Programm wie folgt:

```
6512 IF X=3 THEN GOSUB 6800
6513 IF X=4 THEN GOSUB 6900
6514 IF X=5 THEN GOSUB 7000
6515 IF X=6 THEN GOSUB 7050

6820   CLS
6895   LET I$=INKEY$: IF I$=""THEN GO TO 6895
6910   CLS
6970   LET I$=INKEY$: IF I$=""THEN GO TO 6970
7010   CLS
7060   CLS
7082   INPUT I$: LET I$=I$(1 TO 1)
7155   LET I$=INKEY$: IF I$=""THEN GO TO 7155
```

Acorn B:
Führen Sie folgende Änderungen durch:

```
6820   CLS
6895   I$=GET$
6910   CLS
6970   I$=GET$
7010   CLS
7060   CLS
7155   I$=GET$
```

Reden ist Silber

Sprachsynthesizer gibt es heute für fast jeden Heimcomputer. Wir stellen zwei neue Sprachsynthesizer für den Acorn B und den Schneider vor. Was haben sie uns zu sagen?

In den letzten Jahren hat die Sprachsynthese große Fortschritte gemacht. Allmählich dringen Synthesizer auch in den Bereich der kleineren Computersysteme vor. Frühere Synthesizer hatten eine Liste von Wörtern in ihrem Speicher. Wenn Text eingegeben wurde, verglichen sie die empfangenen Wörter mit dem Speicherinhalt. Bei Übereinstimmungen wurde ein Unterprogramm aufgerufen, um das gefundene Wort aus einer Tonfolge zusammenzusetzen. Moderne Sprachsynthesizer sind leistungsfähiger – der Anwender kann einen beliebigen Satz eingeben, der von den Geräten in Wörter umgesetzt wird.

Es ist ziemlich schwierig, einen geschriebenen Text in verständliche Sprache umzusetzen. Die Komplexität von Sprache übertrifft auch noch das ausgefeilteste Computerprogramm. Ein Beispiel: Die Wörter „naiv" und „Mai" beinhalten die gleichen Vokale, die jedoch völlig unterschiedlich ausgesprochen werden. Ein Computer kann also höchstens versuchen, die richtige Aussprache zu treffen. Moderne Sprachsynthesizer sind zwar nicht perfekt, erzielen aber bereits recht bemerkenswerte Ergebnisse.

Ausnahmen der Regel

Diese Systeme lesen eine Kette von ASCII-Zeichen in einen Bufferspeicher. Jedes Wort wird geprüft und mit einer Reihe grammatikalischer Regeln verglichen, die vom Programmierer definiert worden sind. Teile eines Wortes werden nach bestimmten Aussprachеregeln interpretiert. Eine solche Regel könnte etwa lauten: Ein „ch", das auf ein „e" oder „i" folgt, wird anders als nach anderen Vokalen artikuliert. Durch diese Regel würde sich etwa das „ch" in „dich" von „doch" unterscheiden. Das Problem von Sprache, speziell der deutschen Sprache ist aber, daß sie zwar festen Regeln folgt, aber eine Vielzahl von Ausnahmen enthält: Nach der oben erwähnten Regel würde das „ch" in „Fuchs" nicht definiert. Also wird bei vielen Sprachsynthesizern zusätzlich eine Liste der häufigsten Ausnahmen vorgesehen. Der begrenzte Speicherplatz eines Computers macht es aber unmöglich, alle Ausnahmen zu berücksichtigen. Selbst wenn genügend Speicherkapazität vorhanden wäre, würde sich beim Überprüfen aller Möglichkeiten eine unannehmbar lange Verarbeitungszeit ergeben.

Bei den hier vorgestellten Geräten handelt es sich um den Namal Type & Talk für den Acorn B und den Amsoft-Speech-Synthesizer für den Schneider, die beide mit einem System von Regeln für die Erzeugung allerdings von englischer Sprache arbeiten.

Der Type & Talk gehört zu den Peripheriegeräten, die für die Nutzung des Acorn B an englischen Schulen entwickelt wurden. Das Gerät befindet sich in einem beigefarbigen Metallgehäuse. Angeschlossen wird der Synthesizer entweder über die Centronics- oder die RS423-Schnittstelle. Die Buchsen dafür liegen an der Rückwand des Gerätes, wo auch der Anschluß für einen externen Lautsprecher ist.

Vorn am Gerät sitzen rechts und links des Lautsprechers der Ein/Ausschalter und ein Lautstärkeregler. Im Inneren finden sich zwei Leiterplatten, eine für das Netzteil und die Synthesizerlogik, die zweite zur Verstärkung des Tonsignals und die eigentliche Lauterzeugung.

Namal Type & Talk

SPRACHCHIP
Votrax SC01A

BETRIEBSARTEN
Direkte Umsetzung Text-Sprache und Phonem-Modus

SPRACHAUSGABE
Verstärker und Lautsprecher sind eingebaut

SCHNITTSTELLEN
Centronics und RS423-Schnittstelle. Anschluß für Kopfhörer oder externen Verstärker.

VORZÜGE
Kann eine Vielzahl von Wörtern exakt aussprechen. Durch die ROM-Software des Gerätes bleibt der Computer frei für andere Aufgaben.

NACHTEILE
Der Synthesizer neigt zur Überhitzung, dadurch wird die Sprachausgabe gestört. Der Preis des Gerätes ist verhältnismäßig hoch.

Der Namal Type & Talk ist der Sprachsynthesizer für den Acorn B. Das Gerät verfügt über einen eigenen Microprozessor, die Software befindet sich im ROM. Der Synthesizer wurde speziell für Schulen und Fortbildungseinrichtungen konzipiert.

Auf der ersten Platine gibt es neben den Logikelementen ein 8K-EPROM, das den Wörterbuch- und Regelspeicher enthält. Die Umsetzung der Informationen vom Computer für das Einspeisen in den Synthesizer-Chip besorgt ein NEC D780C-Prozessor, der fast völlig dem Z80 gleicht. Auf der Platine finden sich außerdem zwei KByte RAM, die als Buffer für eingehende Signale dienen. Mit acht DIP-Schaltern kann das Gerät auf die Baudrate der ankommenden

Amsoft Speech Synthesiser

SPRACHCHIP
SP0256

BETRIEBSARTEN
Umsetzung Text-Sprache und Allophon-Modus

SPRACHAUSGABE
Lautsprecherpaar

SCHNITTSTELLEN
Erweiterungsanschluß für den Diskettenport, HIFI-Ausgang.

VORZÜGE
Echte Text-Sprach-umwandlung.

NACHTEILE
Das Verzeichnis der grammatikalischen Regeln ist nicht so umfangreich wie bei Type & Talk.

Signale und andere Bedingungen des „Handshaking"-Betriebs eingestellt werden.

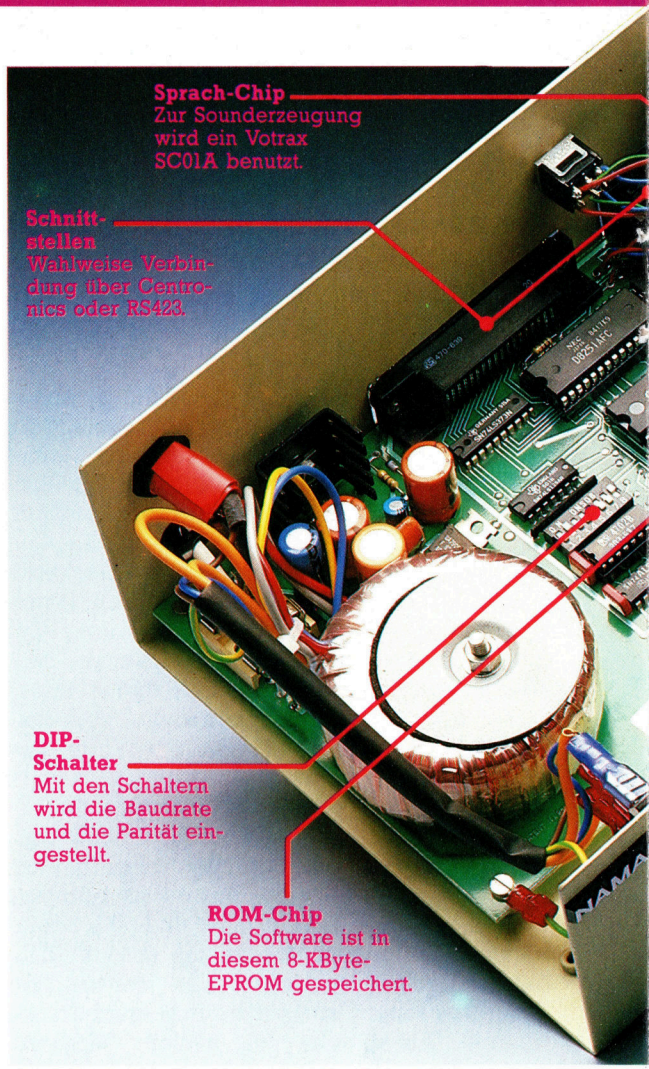
Fast ein Drittel der Platine belegt das eingebaute Netzteil. Es ist recht angenehm, die Stromversorgung in einem Gerät integriert vorzufinden – es gibt nicht so viel Kabelsalat. In diesem Fall scheint man aber nicht alle Probleme dieser Lösung bedacht zu haben: Nach mehrstündigem Betrieb zeigt das Gerät Anzeichen von Überhitzung, die sich in verringerter Sprachqualität ausdrücken.

Auf der zweiten Platine befinden sich Verstärker und ein Regler, mit dem die Geschwindigkeit der über den Lautsprecher ausgegebenen Sprache eingestellt werden kann. Die eigentliche Spracherzeugung besorgt der Synthesizer-Chip Votrax SC01A, den man auch in vielen anderen Sprachsynthesizern findet.

Nach dem Anschluß des Gerätes an den Rechner ist die Spracherzeugung erstaunlich einfach. Der Type & Talk verkündet die Meldung „Ready Master" und zeigt so seine Arbeitsbereitschaft an. Da der Synthesizer seine Signale (im ASCII-Format) über die Drucker-Schnittstellen erhält, muß der Computer so eingestellt werden, daß er alle Bildschirmeingaben auch an den Druckertreiber ausgibt. Beim Acorn B geht das entweder durch Eintippen von VDU 2 oder durch Übermittlung eines CTRL B-Zeichens zum Type & Talk. Danach werden sofort alle eingetippten Sätze oder Wörter vom Synthesizer gesprochen.

Die Funktionsweise des Type & Talk-Systems und auch seine durch die Hardware bedingten Grenzen kennen wir. Um trotz der Schwierigkeiten eine korrekte Aussprache zu erlangen, ist es häufig notwendig, die Schreibweise eines Wortes zu verändern. Es braucht manchmal eine Menge Einfallsreichtum, bis Worte richtig klingen. Das englische Wort „bough" (Zweig) wird vom System genau so interpretiert wie das Wort „cough" (Husten) und wie „boff" ausgesprochen. Auch wenn man „bow" als alternative Schreibweise eingibt, er-

Im Verhältnis zum Type & Talk hat der Amsoft Speech Synthesiser einen weitaus günstigeren Preis, obwohl auch dieses Gerät nach dem Verfahren grammatikalischer Regeln arbeitet, das man sonst nur bei teuren Modellen findet. Die Software wird über Cassette geladen, die eigentliche Umsetzung von Text in Sprache führt der Prozessor des Computers aus. Voraussetzungen für einen Einstieg in die Sprachsynthese mit kleineren Computern bringt aber auch dieses eher schlichte Gerät mit.

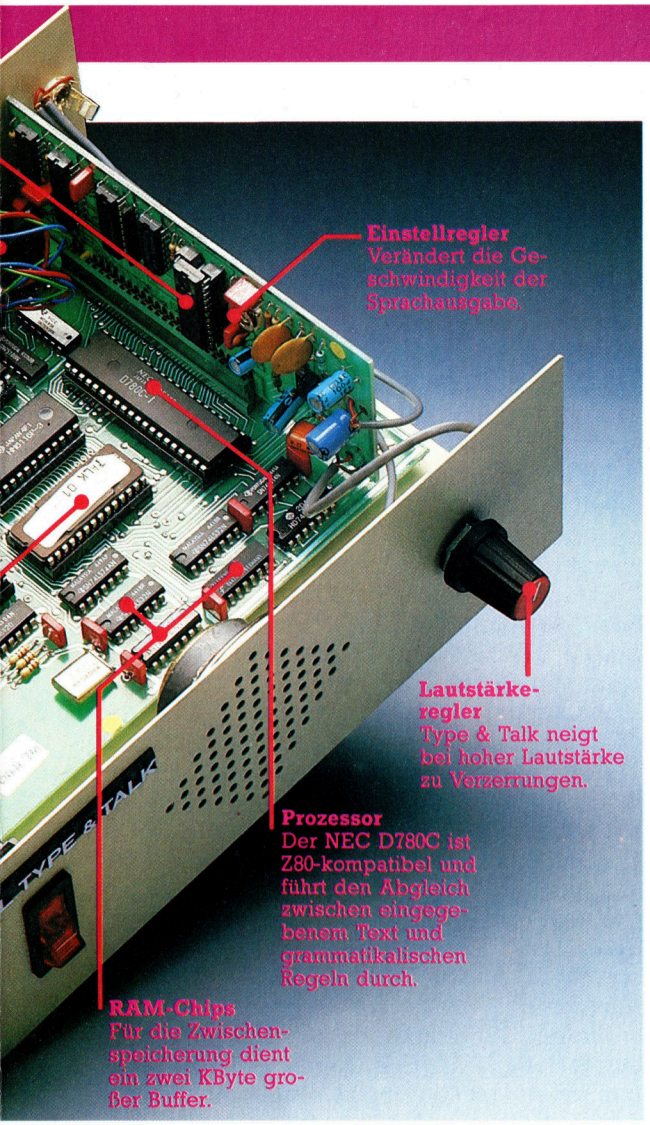


Lautschrift

Im Gegensatz zum Lesen, wo ein Wort an seiner Schreibweise wiedererkannt wird, erkennen wir das gesprochene Wort am Zusammenklang der „Phoneme", der einzelnen Lauteinheiten. Die Zusammenfassung mehrerer Phoneme bildet den „Körper" des Wortes, und die kleinen Abweichungen (je nach dem Kontext des Wortes und seiner Stellung im Satz) werden durch „Allophone" erzeugt. Das englische „ch" kommt im Wort „chip" ebenso vor wie bei „batch", wird dabei aber völlig verschieden ausgesprochen. Die Abweichungen in der Aussprache werden durch die Allophone erzeugt.

Die folgenden Beispiele zeigen die großen Unterschiede zwischen der phonetischen und der üblichen Schreibweise englischer Texte:

THV ER PA0	K W I K PA0	B R AW2 AW2 N PA0
THE	QUICK	BROWN
F O2 O2 K S PA0	D J UH1 M P T PA0	O V E R PA0
FOX	JUMPED	OVER
THV ER PA0	L A Z I PA0	D O G STOP
THE	LAZY	DOG



klingt das Wort so, wie es bei „low" richtig wäre. Die richtige Aussprache erreicht man erst mit der Schreibweise „bou".

Die Betonung auf einzelnen Silben läßt sich durch Eingabe von Steuerzeichen zwischen den einzelnen Buchstaben erreichen. Zur Unterscheidung von normalen ASCII-Zeichen wird ihnen ein ! vorangestellt. Beispiel: HEL!InLO, wobei I für Tonfall steht und n eine Zahl zwischen 0 und 3 ist.

Noch weit größere Wandlungsfähigkeit ermöglicht der mit !P aufrufbare Phonem-Modus. Phoneme sind Zeichenketten, aus denen sich Worte aus einzelnen Lauten kombinieren lassen. (Phonetische Schrift oder Lautschrift findet man sonst häufig in Wörterbüchern, wo sie in Klammern hinter dem Haupteintrag die Aussprache eines Wortes erklärt.) Aus den Lautschriftzeichen kann theoretisch jede beliebige Lautfolge zusammengesetzt werden.

Ganz anders arbeitet der Amsoft Speech Synthesiser. Das Gerät kann direkt am Diskettenport des Schneider-Computers angeschlossen werden. Die Umsetzung von Text in Sprache wird vom Prozessor des Computers selbst erledigt, wobei das eigentliche Wörter- und Regelverzeichnis von der Cassette geladen werden muß. Die Steuerbefehle gelangen auf etwas andere Weise zum Synthesizer als beim Acorn B: Es gibt neun Befehle, mit denen sich das Gerät von Basic aus steuern läßt. Diese Befehle sind, wie das Disketten-Betriebssystem des Schneider, fest integrierte Systemerweiterungen (RSX). Das Problem dabei ist, daß Parameter erst in Strings umgewandelt werden müssen, bevor sie an die RSXs übergeben werden können. So erfordert etwa der Befehl |SAY (zusammen mit |ECHO die Anweisung für direkte Text/Sprachumwandlung), daß der einzugebende Satz erst als String definiert werden muß. Das |ECHO-Kommando zur Wiedergabe des Bildschirminhalts verfügt glücklicherweise über die Fähigkeit der Parameterübergabe.

Leistungsunterschiede

Der Amsoft Speech Synthesiser erreicht weder in der Wiedergabequalität noch beim Umsetzen selbst die Leistungen des Type & Talk. Ganz überraschend ist das nicht – das Amsoft-Gerät ist erheblich preiswerter als der Type & Talk. Beim Amsoft Speech Synthesiser gibt es sehr viel weniger Regeln, die auch nicht so streng beachtet werden wie beim Type & Talk. Daher werden im Amsoft-Produkt viele Wörter falsch ausgesprochen, die beim Konkurrenten auf Anhieb richtig klingen. So wird etwa aus „able" „abble" und „cough" klingt wie „cowf". Einfallsreiche Schreibweise hilft aber auch in diesen Fällen oft weiter, ebenso wie bei deutschen Wörtern.

Ein zusätzliches Kommandowort für den Amsoft ist |APHONE, das ungefähr dem !P-Befehl des Acorn-Synthesizers entspricht und den Aufbau von Wörtern aus ihren Grundkomponenten ermöglicht. Allerdings wird in diesem Fall mit sogenannten „Allophonen" gearbeitet – das sind die Bausteine, aus denen sich die oben erwähnten „Phoneme" zusammensetzen, also noch kleinere Bruchstücke von Lauten. Auf den Befehl |APHONE folgt eine Reihe von Zahlen, die jeweils einem bestimmten Laut-Bruchteil entsprechen. Das geht zwar nicht ganz so schnell und direkt wie beim Type & Talk, weil man sich die Zahlen heraussuchen muß, allerdings gewöhnt man sich relativ schnell an dieses Verfahren.

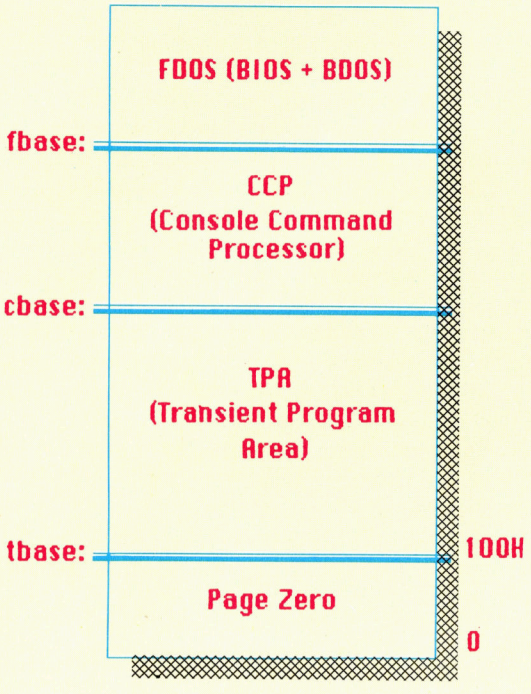
Die Sprachsynthese ist zwar aus den Kinderschuhen heraus, ein Randproblem ist aber immer noch nicht gelöst: Die Synthesizer sprechen nach wie vor im „Roboter-Sound". Auch die Beachtung der Sprachregeln macht oft noch Schwierigkeiten und läßt Raum für Weiterentwicklung. Die Hochleistungstechnologie hat zwar bereits erhebliche Fortschritte gemacht, doch sind diese Techniken gegenwärtig noch zu aufwendig und teuer, um sie auf kleinere Computer umzusetzen. Allerdings kann man darauf hoffen, daß die CD-ROMs (ROM auf Compact-Disks) hier neue Wege eröffnen. Die beiden vorgestellten Geräte zeigen im Ansatz bereits, zu welchen Leistungen zukünftige Sprachsynthesizer fähig sein werden.

Vielversprechend

In der letzten Folge unserer CP/M-Serie sehen wir uns an, wie der begrenzte Platz im RAM von den CP/M-Modulen optimal eingesetzt wird. Wir untersuchen weitere Funktionen der drei CP/M-Hauptmodule und werfen einen Blick auf die Zukunft dieses weitverbreiteten Betriebssystems.

Eng gepackt

Das Bild zeigt die Speicherbereiche, in denen die verschiedenen Teile von CP/M untergebracht sind.
An der Speicheruntergrenze, auf der Zero Page, befinden sich die Systemvariablen, die Einsprungspunkte und der Urlader. Darüber liegt der Programmspeicher (TPA), dessen Obergrenze sich verschieben läßt, so daß Dateien, die normalerweise nicht in den Speicher passen, den „Console Command Processor" überschreiben können. Oben im Speicher liegen die Routinen des BIOS und BDOS. Sie dürfen nicht überschrieben werden, da sie bei der Befehlsausführung ständig eingesetzt werden.



Die drei Hauptmodule von CP/M befinden sich an der Obergrenze des Arbeitsspeichers. Ihre Startadressen sind von Version zu Version verschieden. Das BDOS (Dateiverwaltungssystem) und das BIOS (Ein/Ausgabesystem zur Steuerung der Peripheriegeräte) liegen direkt unter der Obergrenze des Speichers. An diese beiden Module schließt sich der Befehlsinterpreter CCP an.

Die ersten 256 Bytes (Zero Page) am anderen Ende des RAM enthalten die Systemvariablen und Informationen, die CP/M zum Funktionieren braucht, etwa die Einsprungspunkte zu den BDOS- und BIOS-Bereichen und zum Urlader. Da während des Arbeitsvorganges CP/M-Routinen nachgeladen werden, muß der Urlader im RAM bleiben.

Auf der Zero Page liegt auch der „Transient File Control Buffer" (TFCB – Buffer für Dateiinformatinen). In der vorigen Folge wurde bereits erwähnt, daß der CCP beim Laden einer Diskettendatei einen vorläufigen File Control Block (FCB) anlegt. Das BDOS vergleicht die Dateinamen des Directories mit diesen Bufferdaten und übergibt dann Informationen der Directoryspur an das CCP. Diese Informationen werden im TFCB gespeichert.

Zwischen der Zero Page und dem CCP liegt der Programmspeicher (Transient Program Area – TPA). Er ist der eigentliche Arbeitsbereich von CP/M. Wenn ein Befehl an das Betriebssystem übergeben wird, durchsucht der CCP zunächst die Liste der eingebauten CP/M-Befehle. Findet er dort keine Übereinstimmung, nimmt er an, daß es sich um einen Diskettenbefehl handelt und veranlaßt das BDOS, die Diskette zu durchsuchen. Das BDOS sieht nun auf der Systemdiskette nach, kopiert das (gefundene) Maschinencodeprogramm von $100 an (das erste Byte der Seite 1 des Arbeitsspeichers) in den TPA und veranlaßt dessen Ausführung.

Um mit CP/M arbeiten zu können, müssen mindestens 16 KByte RAM frei verfügbar sein. Das scheint recht wenig zu sein, wenn man bedenkt, daß dort außer den normalen Programmen auch CP/M Platz finden soll. Da das Betriebssystem jedoch nur einen kleinen Teil des Speichers belegt, stehen für den TPA sieben KByte zur Verfügung (bei minimalem Speicherausbau endet der TPA bei $2900, die Zero Page wird für Systemvariablen eingesetzt und der Rest für CP/M).

Da die meisten Diskettenbefehle nur zwei bis drei KByte belegen, steht der übrige TPA für Dateien zur Verfügung, die von diesen Befehlen bearbeitet werden. Der TPA wird also nicht nur für Programme, sondern auch für Dateien genutzt.

TPA-Daten überlagern CCP

Hier entsteht bei minimalem Systemausbau ein Problem. Wenn von den sieben KByte TPA drei KByte für Diskettenbefehle eingesetzt werden, bleiben nur vier KByte für weitere Daten übrig. Selbst für kleine Programme reicht das kaum, längere Textdateien haben darin überhaupt keinen Platz. Nun ließe sich der Speicher – und damit auch der TPA – durch den Einbau zusätzlicher RAM-Chips bis auf 64K erweitern (der Speicherbereich, der von CP/M direkt adressiert werden kann), doch schafft es CP/M auch bei minimaler Auslegung, dort mehr als vier KByte unterzubringen.

Die Lösung ist einfach: TPA-Daten, die über vier KByte hinausgehen, „überlagern" den CCP. Wie alle Betriebssysteme akzeptiert CP/M während der Ausführung eines Befehls keine weiteren Befehle und kann daher das CCP überschreiben, das zu diesem Zeitpunkt nicht gebraucht wird.

Für die Aufnahme eines neuen Befehls muß CP/M nach Beendigung eines Ablaufs des CCP neu laden. Der letzte Vorgang eines Diskettenbefehls ist daher immer der Aufruf der „Location Boot"-Routine (Adresse $0005 auf der Zero Page). Über diesen Einsprungspunkt lädt das Betriebssystem das CCP-Modul wieder in den Speicher und ist dann bereit, den nächsten Befehl entgegenzunehmen.

Da die Hardware zum Zeitpunkt der Entwicklung von CP/M noch sehr teuer war (besonders die RAM-Chips), besaßen nur wenig Maschinen mehr als 16 KByte RAM. Die Software mußte sich nach den vorhandenen Grenzen der Maschinen richten.

Bereiche überlappen

In CP/M überlappen sich daher einige Bereiche, um so wenig Speicher wie möglich zu belegen. Auf BDOS und BIOS konnte nicht verzichtet werden, da viele Befehle sie ständig für den Dateizugriff oder Ein- und Ausgabevorgänge brauchen (beispielsweise zur Zwischenspeicherung einer Druckdatei oder für die Bildschirmanzeige von Daten). CCP und TPA sind jedoch nie gleichzeitig aktiv, und so entschieden sich die Designer von CP/M, beide in den gleichen Bereich zu legen.

Gary Kildall und sein Team verdienen große Anerkennung für die Entwicklung eines Systems, das die damaligen Hardwaregrenzen berücksichtigt, aber auch heute noch seine Gültigkeit hat.

Da inzwischen ein Großteil der kommerziellen Micros mit 16-Bit-Prozessoren arbeitet, liegt die Vermutung nahe, daß die Zeit für Acht-Bit-Betriebssysteme abgelaufen ist. Doch trotz der Vorherrschaft von 16-Bit-Betriebssystemen (wie MS-DOS) scheint der Z80-Prozessor unverwüstlich zu sein.

Kleine Geschäftscomputer und Heimgeräte brauchen nur selten die großen Kapazitäten der 16-Bit-Prozessoren, wohl aber ein Betriebssystem, das die Diskettenlaufwerke steuert. Hierfür ist CP/M ideal geeignet, da es lange Zeit der Standard für die Acht-Bit-Geräte war und über eine immense Softwarebasis verfügt. Einige Computerhersteller haben diesen Vorteil erkannt und ihre Produktion umgestellt.

Trotz der Softwarebasis von etwa 10 000 CP/M-Programmen gibt es Kompatibilitätsprobleme. Bei speziellen Diskettenformaten – wie beispielsweise bei der Schneider-Floppy – sind zusätzliche Schwierigkeiten zu überwinden. CP/M wurde ursprünglich für 8" und 5 1/4" Disketten entwickelt, Schneider verwendet jedoch das 3" Format von Hitachi. Es kann daher eine Weile dauern, bis genügend CP/M-Pakete auch für diese Laufwerke zur Verfügung stehen.

Schneider 664 Computer haben jedoch noch ein weiteres Problem: Die Bildschirmsteuerung läßt nur 38 KByte für Programme übrig, weit weniger, als die meisten neuen CP/M-Programme benötigen. Bis die Software auf den verfügbaren Speicherplatz zurechtgeschnitten ist, können CPC 664-Anwender nur mit älteren bzw. gekürzten Programmversionen arbeiten, die noch mit weniger RAM auskommen. Doch abgesehen davon scheint CP/M immer noch eine Anwenderbasis zu haben.

Digital Research hat den 16-Bit-Markt nicht aufgegeben, auch wenn es heute mehr Konkurrenten gibt, als zur Zeit, als Gary Kildall die erste CP/M-Version entwickelte. CP/M-86 war der erste Versuch, in den Markt um den Intel 8088/86 Chip einzubrechen. Die Anwender entschieden sich jedoch für MS-DOS und die damit verbundene IBM-Kompatibilität.

Vor kurzem stellte Digital Research nun Concurrent CP/M vor – eine Mehrplatzversion von CP/M. Dieses Betriebssystem läßt sich als Einplatzsystem einsetzen, unterstützt aber auch die Arbeit mehrerer Computer, die in einem Netzwerk zusammengeschlossen sind. Es läßt sich zwar noch nicht sagen, ob Concurrent CP/M den Erfolg seines Acht-Bit-Vorgängers erreichen wird, mit Sicherheit ist jedoch damit zu rechnen, daß CP/M in der einen oder anderen Form noch lange im Markt bleiben wird.

MP/M und CP/NET

In dieser Serie haben wir uns fast ausschließlich mit der CP/M Version (2.2) beschäftigt, die am weitesten verbreitet ist. CP/M (2.2) wurde als Einplatzsystem für Einplatzanwendungen entwickelt. Digital Research bietet jedoch auch Systeme für „Multitasking", die sich also für den vernetzten Einsatz von mehreren Maschinen eignen.

In einem Multitaskingsystem benutzen mehrere Anwender oder Peripheriegeräte die gleiche Zentraleinheit. Das entsprechende Betriebssystem heißt MP/M (Multi-Processing Monitor Control Program). Unter MP/M können an einen Computer bis zu 16 Terminals angeschlossen werden, die unabhängig voneinander arbeiten. Zur Bewältigung dieser Aufgabe erhielt MP/M eine Reihe Befehle, die den gleichzeitigen Zugriff auf Dateien oder Systemkomponenten (zum Beispiel Drucker) steuern.

CP/NET ist eine weitere Variante von CP/M. Bei dieser Netzwerkversion von CP/M können mehrere Anwender mit unterschiedlichen Maschinen Daten austauschen. Im Gegensatz zu MP/M braucht CP/NET kein Mastersystem mit untergeordneten Terminals, doch gibt es eine „Master-Schaltstelle", die das Netzwerk steuert.

Aus alt mach neu

Da FORTRAN aus den fünfziger Jahren stammt, fehlen dieser Sprache viele Strukturen und Möglichkeiten, die in modernen Programmiersprachen als selbstverständlich angesehen werden. Wir sehen uns einige Techniken an, die diesen Mangel beheben.

FORTRAN fehlen zwei wichtige Eigenschaften: Es hat keine brauchbaren Steuerstrukturen, die die Sprache leicht verständlich machen und verfügt außerdem nur über eingeschränkte Möglichkeiten der Zeichenverarbeitung. Im Laufe der Zeit gab es eine Reihe von Versuchen, diese Einschränkungen gänzlich zu beseitigen.

Eine Möglichkeit dafür bietet der Einsatz eines „Vorprozessors" (eine Art Compiler auf einer höheren Ebene). Er bearbeitet Programme, die mit einer erweiterten FORTRAN-Version erstellt wurden und setzt für alle Zusatzeigenschaften die entsprechenden Abläufe des Standard-FORTRAN ein. Damit lassen sich strukturierte Programme schreiben, die in kompilierter Form dem Standard entsprechen und sich allgemein verwenden lassen. WATFOR und RATFOR sind die am weitest verbreiteten Versionen.

Die Eigenschaften von FORTRAN 77 entsprechen denen der bekannteren Vorprozessoren. Aus diesem Grund blieb die Essenz der Sprache erhalten, obwohl viele Compiler für Microcomputer auf dem erweiterten FORTRAN IV aufbauen und nicht auf FORTRAN 77.

FORTRAN besitzt keine Blockstruktur wie ALGOL, PASCAL oder C, was sich speziell bei der Einrichtung von Steuerstrukturen negativ bemerkbar macht. Außerdem können nicht mehrere Befehle zu einer Anweisung zusammengefaßt werden (beispielsweise durch BEGIN. . .END). Blöcke lassen sich nur isolieren, wenn sie als separate Subroutinen angelegt werden, oder von GOTO-Befehlen umgeben sind. Es gibt jedoch eine neue Steuerstruktur, die allerdings nicht so recht zu der Sprache paßt: IF. . .THEN. . .ELSE. . .ENDIF:

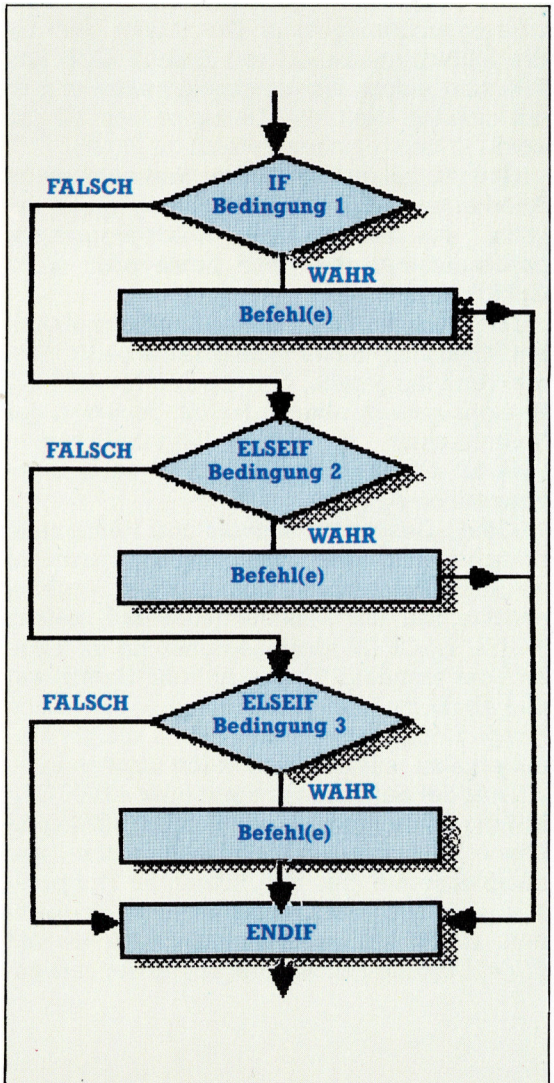
```
IF (logischer Ausdruck) THEN
    . . . . . .
    Anweisungen
    . . . . . .
ELSE
    . . . . . .
    Anweisungen
    . . . . . .
ENDIF
```

Der logische Ausdruck hat das gleiche Format wie Standard-FORTRAN. Normalerweise wird der ELSE-Teil weggelassen.

Eine sehr praktische Form dieser Anweisung arbeitet mit verschachtelten IFs:

```
IF (logischer Ausdruck) THEN
    . . . . . .
    Anweisungen
    . . . . . .
ELSEIF
    . . . . . .
    Anweisungen
    . . . . . .
ELSEIF
    . . . . . .
ELSE
    . . . . . .
ENDIF
```

Es lassen sich beliebig viele ELSEIFs für tiefere Strukturierung verwenden.



In den meisten BASIC-Versionen sind verschachtelte IFs möglich. FORTRAN hat jedoch zusätzlich die Struktur ELSEIF, mit der sich IF-Anweisungen über mehrere Programmzeilen bis zu einer beliebigen Tiefe verschachteln lassen. ENDIF beendet die Struktur.

Die zweite wesentliche Verbesserung ist die Einführung eines Datentyps für Zeichen. Zeichenstrings werden – wie die anderen Deklarationen – am Programmanfang in einem der folgenden Formate festgelegt:

```
CHARACTER*9          CH1,CH2
CHARACTER            CH3*7,CH4*16
```

Die maximale Länge eines Strings wird hier mit *n angegeben, wobei n eine Ganzzahl ist. Dieser Vorgang kann entweder auf das Schlüsselwort CHARACTER angewandt werden (wenn alle Strings die gleiche Länge haben), oder auf einzelne Strings mit individuellen Bezeichnungen. Die beiden aufgeführten Deklarationen ergeben zwei Strings der Länge 9, einen mit Länge 7 und einen mit Länge 16.

Stringarrays werden wie üblich definiert:

```
CHARACTER*4 CHARS(50)
```

deklariert ein Array von 50 Strings mit jeweils vier Zeichen. Stringkonstanten werden Stringvariablen folgendermaßen zugeordnet:

```
CH1'ABCDEFGHI'
```

Ist der zugeordnete String zu kurz, werden Leerzeichen angehängt; ist er zu lang, werden die Zeichen am Stringende ignoriert.

Strings können mit normalen relationalen

CODEKNACKER

Dieses FORTRAN-Programm liest Daten aus einer Datei namens GEHEIM und dechiffriert über einen Decodierschlüssel den chiffrierten Text.

```
C     EIN PROGRAMM, DAS GEHEIMNACH-
C     RICHTEN DECODIERT. BEACHTEN SIE DIE
C     ANWEISUNG ‚PROGRAM' IN FORTRAN 77
C
      INTEGER COUNT,PTRIN,PTROUT
      LOGICAL FILE
      CHARACTER *10 NUM, CHAR
      CHARACTER *30 IN, OUT
      DATA COUNT /0/
C
C     ‚DATA' ANWEISUNG ZUR INITIALISIERUNG
C     EINER VARIABLE
C
C     WENN KEINE GEHEIMDATEI EXISTIERT,
C     FEHLERMELDUNG AUSGEBEN
C
      INQUIRE (FILE='GEHEIM',EXIST=FILE)
      IF (.NOT.FILE) THEN
        WRITE(1,10)
        STOP
      ENDIF
C
C     DATEIEN ERÖFFNEN
C
      OPEN (UNIT=12,FILE='MELDUNG',STATUS
        ='NEW')
C
C     DECODIERSCHLÜSSEL ANGEBEN
C
      NUM='0123456789'
      CHAR='MW3 OD$%^T'
C
C     GEHEIME MELDUNG LESEN UND
C     DECODIEREN
C
100   READ (11,20,END=1000)IN
        OUT=' '
        PTROUT=1
        DO 200I=1,30
          PTRIN=INDEX (CHAR,IN(I:I))
C
C     STANDARDFUNKTION 'INDEX'
C     BESTIMMT DIE POSITION
C     DES TEILSTRING IN(I:I) IM STRING CHAR
        IF (PTRIN.NE.O) THEN
          OUT (PTROUT:PTROUT)=NUM
          (PTRIN:PTRIN)
          PTROUT=PTROUT+1
        ENDIF
200   CONTINUE
      WRITE (12,20) OUT
C
C     BEACHTEN SIE, DASS EIN- UND AUSGABE
C     DAS GLEICHE FORMAT HABEN
C
        COUNT=COUNT+1
      GOTO 100
C
C     ENDVORGANG
C     MIT 'ENDFILE' WIRD DIE MARKIERUNG
C     DES DATEIENDES GESCHRIEBEN
C
1000  ENDFILE (UNIT=12)
      STOP
C
C     FORMATS
10    FORMAT (1H,'ES EXISTIERT
      KEINE GEHEIMDATEI')
20    FORMAT (A)
C
C     BEACHTEN SIE, DASS NICHT
C     ANGEGEBEN WERDEN MUSS,
C     WIEVIEL ZEICHEN
C     EINZUGEBEN SIND
C
      END
```

Operatoren wie GT.,.LT etc. verglichen werden. Es gibt jedoch zwei neue Operatoren nur für Stringfunktionen. Der Teilstringoperator liefert eine Zeichenfolge des Strings.

```
CH1(3:5)
```

holt den Teilstring von CH1, der beim dritten Zeichen beginnt und beim fünften endet.

```
CH1='ABCDEFGHI'
CH2=CH1(3:5)
```

legt in CH2 den Inhalt ‚CDE', oder genauer 'CDE ' ab. Beachten Sie, daß die zweite Zahl die Position des letzten Zeichens angibt und nicht die Länge des Teilstrings.

Der Verkettungs-Operator (//) verbindet zwei Strings miteinander.

```
CHARACTER CH1*4,CH2*4,CH3*8
CH1='ABCD'
CH2='WXYZ'
CH3=CH1//CH2
```

ergibt CH3 mit dem Inhalt ‚ABCDWXYZ'.

Um FORTRAN richtig nutzen zu können, ist ein System mit Diskettenlaufwerken nötig. Die meisten FORTRAN-Versionen können sequentiell und direkt auf Dateien zugreifen. Die Quelle oder die Bestimmung eines E/A-Vorgangs wird von der Gerätebezeichnung des READ- oder WRITE-Befehls bestimmt. Einige Parameterwerte – normalerweise die niedrigeren – sind für E/A-Geräte wie Tastatur, Bildschirm und Drucker reserviert, die anderen Nummern stehen dem Programmierer zur Verfügung. Der Befehl OPEN ordnet einer Diskettendatei eine Namen zu:

```
OPEN (UNIT=Ganzzahlausdruck,
FILE=Dateiname, STATUS=Status)
```

Der Ganzzahlenwert für UNIT bezeichnet in den folgenden READ- oder WRITE-Befehlen die Dateinummer. Status kann mehrere Werte annehmen – beispielsweise OLD für eine Eingabedatei, die es bereits gibt, oder NEW für eine Ausgabedatei, die angelegt werden soll.

Falls die angesprochene Datei vom einfachen sequentiellen Format abweicht, kann OPEN um eine Reihe von Zusatzangaben ergänzt werden: ACCESS= bestimmt den direkten oder sequentiellen Zugriff; FORM= gibt an, ob die Datei formatiert oder unformatiert ist; IOSTAT bietet die Möglichkeit, Fehler abzufangen, falls keine Dateizuordnung zustande kommt; RECL= legt die Dateilänge fest.

CLOSE hat einen ähnlichen Aufbau:

```
CLOSE (UNIT=Ganzzahlausdruck,
STATUS=Status)
```

Der Befehl schließt eröffnete Dateien, wird aber nur selten benötigt, da bei Beendigung des Programms alle Dateien automatisch geschlossen werden.

Von den anderen Dateibefehlen ist INQUIRE möglicherweise am interessantesten, da sich damit der aktuelle Status jeder beliebigen Datei feststellen läßt.

Auch zu READ und WRITE sind Zusätze möglich, die die E/A-Vorgänge bestimmen.

```
READ(7,20,REC=I)A,B,C
```

liest beispielsweise Record I einer Datei, die für den direkten Zugriff eröffnet wurde.

```
READ(7,11,END=1000)A,B,C
```

übergibt die Steuerung an Anweisung Nummer 1000, wenn in einer sequentiellen Datei das Dateiende gefunden wird.

Zum Schluß noch ein Blick auf die FORTRAN-Compiler. Sie arbeiten schnell und effektiv und erzeugen schnellen, kompakten und leistungsfähigen Code, der besonders bei Echtzeitanwendungen große Vorteile bringt.

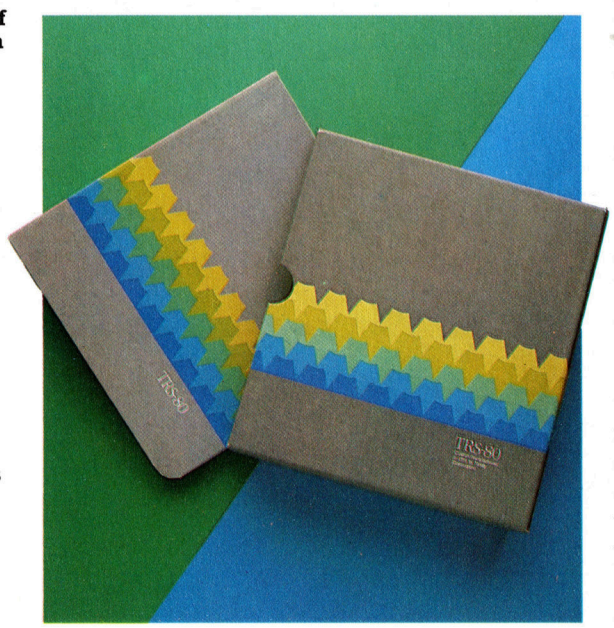
Fehler lassen sich mit den FORTRAN-Compilern jedoch nicht gut abfangen. So akzeptieren die Compiler Fehler, die beispielsweise vor einem PASCAL-Compiler nie bestehen könnten. So haben Leerzeichen in einer FORTRAN-Anweisung etwa keine Bedeutung. Ein Compiler nimmt daher oft als erstes alle Leerzeichen aus einer Befehlszeile. Nun legt der Befehl

```
DO 100 I=1,100
```

eine Schleife an, die 100 mal ausgeführt wird. Ein Punkt anstelle des Kommas ergibt nach Beseitigung aller Leerzeichen jedoch

```
DO100I=1.100
```

und damit eine völlig legitime Wertzuweisung für die Variable DO100I. Ein Gerücht besagt, daß dieser kleine Irrtum im Weltraumprogramm der Vereinigten Staaten zu einem sehr teueren Fehler geführt haben soll. FORTRAN war lange Zeit die Standard Programmiersprache der Streitkräfte. Fehler dieser Art waren Anlaß für die Entwicklung von ADA.



Da FORTRAN-Compiler recht teuer sind, gibt es sie normalerweise nur für kommerzielle Maschinen. Das im Bild gezeigte Paket läuft auf dem Tandy 2000. Durch die begrenzten Speicherkapazitäten sind FORTRAN-Anwendungen auf Micros problematisch, wenn nicht unmöglich.

Mit der wachsenden Verfügbarkeit von CP/M auf Heimcomputern wie dem Commodore 128, Schneider CPC und anderen, stehen vielen Anwendern nun auch FORTRAN-Versionen zur Verfügung. Für Besitzer von Schneider Geräten gibt es eine besonders preiswerte Version, das „Nevada FORTRAN". Sie kann bei New Star Software Ltd., 45 Plovers Mead, Wyatts Green, Essex, CM15 OPS bestellt werden.

Form der Zukunft

Umfangreiche Entwicklungen in der Hardware und eine Revolution der Datenspeicherungstechniken verändern das herkömmliche Klassenzimmer. Wir beenden unsere Serie über „Computer im Unterricht“ mit einer Prognose über die Ausbildung in der Zukunft.

Der Computer-Unterricht wird sich in Zukunft stark verändern. Preiswertere Hardware, verbesserte Software und technologische Entwicklungen werden eine Wandlung des Unterrichts herbeiführen. Ausbildungs-Software wird nicht länger ein unbedeutender Zweig des herkömmlichen Unterhaltungsmarktes sein, Computer-Lernsoftware wird sich zum eigenen, bedeutenden Marktbereich entwickeln. Der Grund für diese Wende ist nicht einmal überraschend. Zu Beginn der Technologie-Revolution übersahen Hard- und Software-Anbieter den Ausbildungsbereich, weil die anderen Sparten zunächst gewinnträchtiger waren. Nachdem Software-Hersteller den Wert erkannt hatten, ihre Produkte an Schulen zu verkaufen, um somit Markenbewußtsein bei den Schülern, ihren möglichen Kunden zu schaffen, änderte sich die Situation.

Spürbar wurde das zunächst mit Preisreduzierungen bei Hard- und Software. 1985 begann Apple mit dem gezielten Verkauf des Macintosh an Schulen und gab 30 Prozent Rabatt. Atari folgte dem Beispiel mit seiner ST-32-

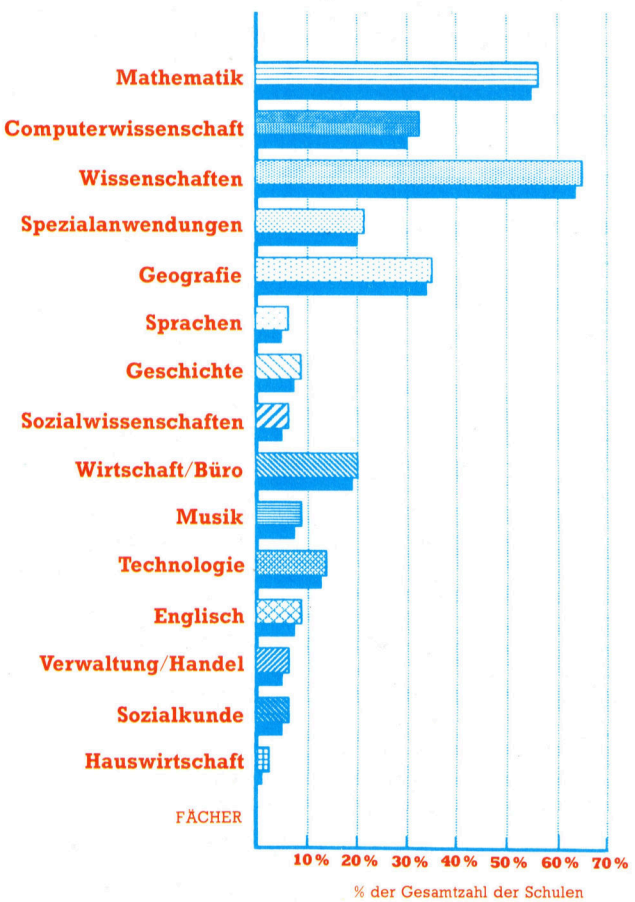
Computer können unser Ausbildungssystem völlig revolutionieren, indem das Kind durch sie in den Mittelpunkt des Lernprozesses gestellt wird und Zugang zu allen technologischen Quellen hat. Satellitentechnik und Netzwerke erlauben Studenten, mit Datenbanken anderer Länder und Kulturen in aller Welt zu kommunizieren. Schulen werden vielleicht „Lern-Überwachungs“-Zentren weichen, in denen Kinder sich treffen, um ihre zumeist daheim erledigten Arbeiten zu diskutieren. Interaktive Videosysteme können Basis für die Entwicklung hochkomplexer Software sein. Roboter finden Anwendung bei der physikalischen Darstellung abstrakter Probleme. Schließlich könnten Schüler und Studenten Zugang zu lokalen Großrechnern haben, wobei der Microcomputer als intelligentes Terminal verwendet wird. Solche Systeme würden den Datenfluß, die Kommunikation zwischen Schülern überwachen und nötigenfalls die Verarbeitungsleistung entsprechend erhöhen.

Die Anwendungen von Micros in Schulen zeigen einen deutlichen Überhang bei wissenschaftlichen Fächern. Die Einführung interaktiver Videosysteme und die Entwicklung von IV-Software kann das Gleichgewicht wiederherstellen. Der Geschichtsunterricht etwa läßt sich durch komplexe Simulationen radikal verändern, die aufgrund des enormen Datenspeicherungspotentials der VideoDisc-Technologie möglich sind.

Bit-Computer-Serie und bot dem Anwender den Vorteil, statt BASIC ein LOGO in ROM-Form zu erhalten. In England wollen Amstrad (Schneider), Sinclair und RML in den Schulen Fuß fassen. Der Computerhersteller Amstrad versucht täglich bis zu 100 Rechner in der Ausbildung unterzubringen.

Neben kommerziellen Absichten, die hinter der Erschließung dieses Marktes stehen, gilt: Es gibt eine zweite Generation von Computeranwendern, die nach ernsthafteren Einsatzmöglichkeiten für ihre Rechner suchen. Software-Entwicklungen, die bisher lediglich für Schulen gedacht waren, werden nun als „ernsthafte" Ausbildungs-Software hergestellt und sind somit marktgerechter. Mangel an Hard- und Software werden also in Zukunft kaum ein Hindernis für den Einsatz von Computern in der Schule sein.

Computer im Unterricht können auf zweierlei Art Einsatz finden: Zur Erweiterung und Verbesserung bestehender (Lehr)-Techniken und zur Einführung völlig neuer Methoden. Seymour Papert hat bereits eine Reihe von Möglichkeiten aufgezeigt, mit denen das LOGO-Konzept auch auf dem Macintosh Anwendung finden könnte. Papert stellt sich eine „Microwelt der Physik" vor, die in LOGO geschrieben ist und es Kindern ermöglicht, mit den physikalischen Gesetzen zu experimentieren und zu spielen. Er prognostiziert ebenfalls eine „Datenbank Microwelt", in der Kinder über Informations-Verarbeitung unterrichtet werden. Einige dieser Ideen Paperts wurden bereits bei Mac'LOGO verwirklicht.



Die wohl beeindruckendsten Möglichkeiten des Computers im Unterricht ranken sich um die Einführung völlig neuer Techniken, speziell des CD-ROMs und des interaktiven Videos (IV). Durch die Möglichkeit, große Datenmengen preiswert zu speichern, ist Ausbildungs-Software nicht mehr an „Linear"-Programme gebunden, wie sie für kleine, anwenderunfreundliche Systeme bisher entwickelt wurden. Die Interaktion bei einem 48 KByte-Programm ist extrem begrenzt. Folglich wirkt solche Software eher hemmend auf den Lernprozeß. Das interaktive Video bietet dem Programmierer größere Flexibilität, das Lernziel zu erreichen, und er kann die größere Speicherkapazität dazu nutzen, das Programm umfangreicher zu gestalten.

Anläßlich des „Council for Educational Technology" (eines Lernseminars für Ausbildungstechniken) wurden vier Bereiche definiert, in den IV an Schulen besonders sinnvoll angewendet werden kann. Lehrer könnten es als Lehrhilfe benutzen, womit Instruktionsprogramme und audiovisuelle Mittel zur Verfügung ständen. Kleinere Gruppen von Schülern könnten IV als „Privatlehrer" einsetzen, den

Der 900. Geburtstag

Die ungeheure Speicherkapazität der Video-Disc ermöglicht völlig neue Anwendungsmöglichkeiten. In Verbindung mit Philips, Thorn EMI und dem Projekt „Microcomputer in Education" gedenkt die BBC des 900. Geburtstages des Reichsgrundbuchs: In einem landesweiten Projekt werden Daten über das England von heute gesammelt.

Schulen überall im Land sammeln Informationen aus ihrer unmittelbaren Umgebung, die zu einer „Datenbank der Menschen" werden. Sie enthalten Einzelheiten über Landnutzung, Sehenswürdigkeiten, Besonderheiten usw. Ergänzend werden nationale Datenquellen (z. B. Volkszählung und Verkehrsaufkommen) zu einer „nationalen Datenbank" kombiniert, die dann auf einer einzigen LaserDisc zur Verfügung steht.

Das Reichsgrundbuch-Projekt ist eine bemerkenswerte Darstellung der heutigen Gesellschaft.

Stoff diskutieren und sich Fragen beantworten lassen. Einzelpersonen könnten es für Simulationen einsetzen, ferner für höher entwickelte CAL-Programme. Schließlich bietet sich der Anwendungsbereich der Datenbank – eine Bibliothek aus akustischem und visuellem Referenzmaterial.

Teddys blaue Augen

Für die Zusammenstellung und Sammlung des Materials ist natürlich noch viel Arbeit erforderlich. Und doch hat man das System bereits in verschiedenen Bereichen eingeführt. Die „Volkshochschule" in England benutzt IV in ihren Sommerkursen. Da nur wenig Zeit zur Verfügung steht, haben etwa Ingenieurstudenten Probleme, die Belastung verschiedener Materialien zu untersuchen. Es gibt ein IV-Programm, das Fotos, Bewegung und Videosequenzen nutzt. Dabei geht es um einen Gerichtsstreit zwischen dem Hersteller von Teddybären und seinem Zulieferer für Unterlegscheiben, mit denen die Teddybär-Augen im Kopf befestigt werden. Der Student hat eine Reihe von Experimenten durchzuführen, die als Expertise in der Verhandlung dienen. Das interaktive Programm wird von einem CAL-Programm gesteuert. Es diskutiert die möglichen Ursachen dafür, daß die Teddybär-Augen ständig herausfallen, etwa den Verschleiß der Substanz, aus der sie bestehen. Der Student führt verschiedene Experimente durch und interpretiert die Ergebnisse, die in einem dramatischen Plädoyer präsentiert werden.

Die Konsequenzen der Übertragung bestehender Software-Techniken – Simulation, CAL, Datenbank und dergleichen – auf CD-ROM-Systeme sind an sich schon aufregend. Langfristig wird die neue Technologie sich aber intensiver auswirken. „Ich glaube nicht", sagt Seymour Papert, „daß es in 20 Jahren noch Schulen gibt. Im Zeitalter des Computers wird sich das Lehrkonzept völlig verändern. Wenn man mit offenen Augen durch die Klassenzimmer geht, sieht man, das vieles überflüssig ist, was in der Schule stattfindet. Manch anderes wichtige geschieht jedoch nicht, etwa, daß Kinder miteinander reden und versuchen, einander zu verstehen . . . Es wird etwas geben müssen, wo Kinder einander begegnen, lernen, sich entwickeln können. Ob wir das Schule nennen oder nicht, ist unwichtig. Wichtig ist nur zu wissen, daß es Schulen im heutigen Sinne nicht mehr geben wird."

Seine Betrachtungen fanden Anklang und wurden weitergeführt. So von Professor Tom Stonier: „Der Computer wird die Ausbildung, den Lernprozeß von der Schule nach Hause verlagern. Die Rolle des Lehrers wird sich verändern. Von einer wandelnden Datenbank zu einem kundigen Berater."

Die möglichen Auswirkungen dieser Verlagerung von system-orientierter Ausbildungsstruktur zur Individualausbildung sind gewaltig. Professor Stonier ist Co-Autor des Buches „Kinder, Computer und Kommunikation", in dem er eine Ausbildung durch ein „elektronisches Großmutter"-System vorhersagt.

Stonier zufolge werden aufgrund der steigenden Lebenserwartung zunehmend ältere Menschen die wichtigen Rollen in der Ausbildung übernehmen, während die Verlagerung des Lehrvorganges von der Schule nach Hause stattfindet. In vielen sogenannten „primitiven" Gesellschaftsformen ist Ausbildung längst Aufgabe der Alten. Stonier glaubt, daß sie ideal für eine Teilverantwortung im Ausbildungsbereich geeignet sind.

Wenngleich sich ein derartiges Projekt derzeit auf behinderte Kinder beschränkt, gibt es doch die Möglichkeit, es auf andere Schüler zu übertragen. In England ist es statthaft, sein Kind selbst zu unterrichten, wenn man die örtliche Schulbehörde davon überzeugen kann, daß man dazu in der Lage ist. Viele Familien möchten diese Möglichkeit nutzen, und über 2000 haben ihre Kinder bereits „ausgeschult". Zugriff zu großen öffentlichen Datenbanken, leistungsfähige Microcomputer mit umfangreichen CD-ROM-Datenbanken sowie interaktive Software werden Eltern optimale Möglichkeiten geben, ihre Kinder daheim zu unterrichten.

Mit zunehmender Komplexität der Lern-Software wird die Rolle des Lehrers immer schwerer zu definieren sein. Stonier sagt voraus, daß Schüler bald mehr als ihre Lehrer wissen werden. Solange Lehrer die Funktion „wandelnder Datenbänke" erfüllen, ist das Wissen eines Kindes durch die „Kapazität" des Lehrer beschränkt. Stonier glaubt, daß dieses Muster sich mit der Entwicklung von Datenbanken und Ausbildungs-Software verändert. Der Lehrer wird so zum Lern-Experten.

Rechneranimierte Filmsequenzen und andere hochauflösende Bildinformation via Computer setzen die Verarbeitung und Speicherung großer Datenmengen voraus und übersteigen die Leistungsfähigkeit von Microcomputern bisher. IV-Technologie macht's möglich. Diese Szenenfotos zeigen ein computergesteuertes Videosystem in Aktion. Hier handelt es sich um ein Ausbildungsprogramm für Ingenieurstudenten. Das Programm „spielt" in einer fachbezogenen Gerichtsverhandlung, in der der Student die Rolle des Experten übernimmt, Experimente ausführt und vor Gericht aussagt.

CD-ROM Revolution

Um Daten zuverlässig speichern zu können, ist in CD-ROMs viel Speicherplatz für Fehlerprüfroutinen sowie Formatieranweisungen erforderlich. Dennoch bietet eine CD-ROM Disc dem Benutzer 552 Megabyte an Speicherplatz, womit Tausende von Informationsfeldern zur Verfügung stehen.

Anders als Audio CD-Spieler, für die aufwendige D/A (digital/analog)-Wandler erforderlich sind, arbeiten CD-ROM-Spieler nur mit digitalen Daten. Damit werden die Kosten für das Gerät beträchtlich reduziert.

Entsprechende Mengen vorausgesetzt werden CD-ROM Discs für denselben Preis erhältlich sein, der heute für Audio Compactdiscs verlangt wird. Da auf einer einzigen CD-ROM Disc zahlreiche Programme abgespeichert werden können, wird es nach ihrer Einführung eine gewaltige Veränderung in der Software-Industrie geben.

Kurvenberechnung

Dies ist der letzte Kursabschnitt, der sich mit dem Robot-Arm befaßt. Wir stellen die Technik vor, die die Bewegungen des Arms weicher und eleganter macht. Dabei hilft die mathematische Methode der „kubischen Interpolation".

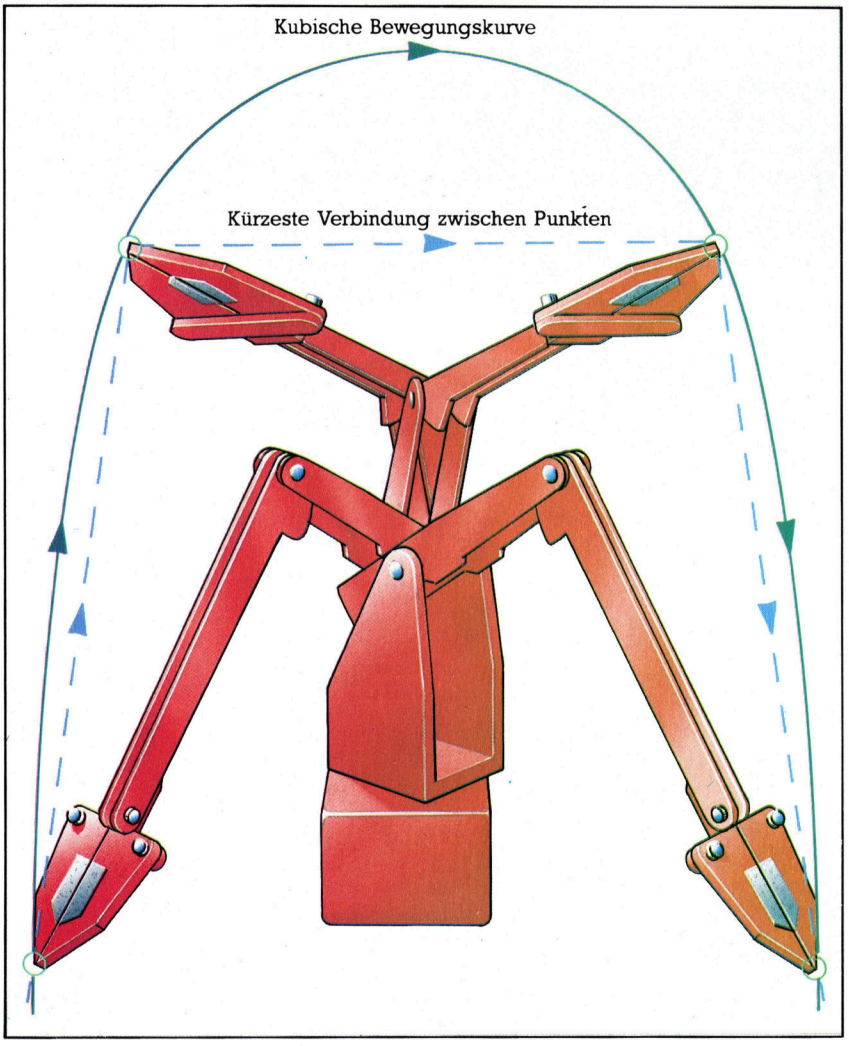
Der Programmteil „moveservo", den wir als Modul der Steuersoftware des Armes eingeführt haben, vermindert sprunghafte Bewegungen beim Wandern von einer Position zur nächsten. Die Motoren bekommen dabei ihre Winkelinformation aus mehreren auf „ANGLE" folgenden Speicherplätzen.

Um den Bewegungsablauf gleichmäßig zu machen, werden die Positionsangaben beim Ansteuern einer neuen Stellung nicht mehr direkt zu den mit den Motoren gekoppelten Speicherstellen gePOKEt, sondern in „NEWPOS"-Speicherstellen abgelegt. Der Aufruf von „moveservo" erhöht bzw. erniedrigt die „ANGLE"-Werte nach einem Vergleich mit „NEWPOS" so lange um je eine Einheit, bis „ANGLE" mit „NEWPOS" übereinstimmt.

Das Einlesen in NEWPOS anstelle von ANGLE schützt die Motoren vor abrupten Positionswechseln – der Robot-Arm gewinnt an Betriebssicherheit und Genauigkeit. Über die Tastatur wird der Arm langsam durch die Zwischenpositionen auf die Endstellung gebracht. Die wichtigsten Punkte des Bewegungsablaufs speichert ein Array. Durch Einbau eines Verzögerungsfaktors in „moveservo" läßt sich der Bewegungsablauf mit unterschiedlichen Geschwindigkeiten wiederholen.

Es sind aber noch weitere Verbesserungen möglich: In erster Näherung sind alle Bewegungen den Veränderungen in NEWPOS proportional. Sie hängen auch noch davon ab, wie lang der Hebel zwischen Armgelenk und Antriebselement ist. Auch der Radius der Antriebscheiben an den Motoren selbst spielt eine Rolle. „moveservo" bewegt jedes Glied des Arms gleichmäßig und mit konstanter Winkelgeschwindigkeit (Grad pro Sekunde) in die neue Position. Am Ziel kann es aber vorkommen, daß der Arm ganz plötzlich seine Bewegungsrichtung verändern muß, um die nächstfolgende Position anzusteuern.

Sind vier Punkte gegeben, die der Robotarm mit einer Bewegung bestreichen soll, dann führt er, geleitet durch unser Programm, nicht den Weg der gestrichelten Linie aus. Vielmehr folgt er der kubischen Bewegungskurve, die einen fließenderen Bewegungsablauf gewährleistet.



Natürliche Bewegung

Dieses Problem ist weniger schwerwiegend als die abrupten Positionswechsel. Durch Speichern einer größeren Zahl von Zwischenpositionen lassen sich die plötzlichen Änderungen der Richtung nach dem Erreichen einer Position vermeiden.

Die Bewegungen des Arms können durch den Einsatz verbesserter Software immer „natürlicher" werden – allerdings in den Grenzen, die vom verfügbaren Speicherplatz gesetzt werden. Beim Transport der Garnrolle durch nur vier Einzelpositionen – Ergreifen des Röllchens, Hochheben, Position über der Tasse einnehmen, Garnrolle wieder herunterlassen – brauchen wir nur wenig Speicherplatz, die Unregelmäßigkeiten der Bewegung sind allerdings erheblich.

Um sie zu vermindern, könnten wir etwa 60 Zwischenpositionen festlegen. Dadurch würde die Bewegung flüssig und elegant. Allerdings wäre dafür sehr viel Speicher nötig, und die zeitraubende Eingabe der Werte über die Tastatur ist ein weiteres Hindernis.

Als Alternative können wir aber eine Reihe von Zwischenpositionen berechnen lassen und sie mit den manuell eingegebenen Werten kombinieren.

Wenn wir uns vier festgelegte Punkte vorstellen, läßt sich ein elastischer Stahldraht stets so biegen, daß er zumindest durch drei dieser Punkte hindurchgeht. Die mathematische Entsprechung zum Biegen eines Federstahls um festgelegte Punkte herum ist die kubische Interpolation (kubisch, weil die mathematische Funktion Werte in der dritten Potenz enthält).

Das abgedruckte Programm zeigt, wie zusätzliche Punkte ermittelt werden können.

Kubische Interpolation

Acorn B

```
1000 REM ************************
1010 REM ** BBC CUBIC SPLINES **
1020 REM ************************
1030:
1040 MODE 0
1050 VDU 23,240,24,126,66,219,219,66,126,24
1060 V=7
1070 DIM X(V+3),Y(V+3),M(V+3),Z(V+3)
1080 X(2)=0 :X(3)=10 :X(4)=40 :X(5)=50
1090 Y(2)=2 :Y(3)=12 :Y(4)=12 :Y(5)=2
1100 REM linearly extend the ends
1110 X(1)=X(2)-(X(3)-X(2))
1120 Y(1)=Y(2)-(Y(3)-Y(2))
1130 X(6)=X(5)+(X(5)-X(4))
1140 X(7)=X(6)+(X(6)-X(5))
1150 Y(6)=Y(5)+(Y(5)-Y(4))
1160 Y(7)=Y(6)+(Y(6)-Y(5))
1170 CLG:VDU 5:MOVE 100,700
1180 PRINT"Demonstration of a routine"
1190 PRINT"which incorporates a cubic spline"
1200 PRINT"joining knots"
1210 FOR I=2 TO 5
1220 MOVE X(I)*20-4,Y(I)*20+15:PRINT CHR$(240);
1230 NEXT
1240 GOSUB 1500:REM SET AKIMA
1250 FOR X=X(2) TO X(5)
1260 GOSUB 1330:REM AKIMA FUNCTION
1270 IF X=X(2) THEN MOVE X*20,Y*20
1280 IF X<>X(2) THEN DRAW X*20,Y*20
1290 NEXT
1300 END
1310 :
1320 :
1330 REM **** AKIMA FUNCTION ****
1340 REM AKIMA RUCKDESCHEL BOOK 2  (BYTE/ MCGRAW-HILL)
1350 N=1
1360 REM CHECK TO SEE IF X IS IN THE TABLE RANGE
1370 IF X<X(1) OR X>X(V-2) THEN RETURN
1380 REM FIND RELEVANT TABLE INTERVAL
1390 I=0
1400 I=I+1:IF X>=X(I) THEN 1400
1410 I=I-1
1420 REM BEGIN INTERPOLATION
1430 B=X(I+1)-X(I)
1440 A=X-X(I)
1450 Y=Y(I)+Z(I)*A+(3*M(I+2)-2*Z(I)-Z(I+1))*A*A/B
1460 Y=Y+(Z(I)+Z(I+1)-2*M(I+2))*A*A*A/(B*B)
1470 RETURN
1480 :
1490 :
1500 REM **** SET AKIMA ****
1510 REM CALCUALATE AKIMA COEFS
1520 FOR I=1 TO V-1
1530 REM SHIFT I TO I+2
1540 M(I+2)=(Y(I+1)-Y(I))/(X(I+1)-X
1550 NEXT I
1560 M(V+2)=2*M(V+1)-M(V)
1570 M(V+3)=2*M(V+2)-M(V+1)
1580 M(2)=2*M(3)-M(4)
1590 M(1)=2*M(2)-M(3)
1600 FOR I=1 TO V
1610 A=ABS(M(I+3)-M(I+2))
1620 B=ABS(M(I+1)-M(I))
1630 IF A+B<>0 THEN Z(I)=(A*M(I+1)+B*M(I+2))/(A+B)
1640 IF A+B=0 THEN Z(I)=(M(I+2)+M(I+1))/2
1650 NEXT I
1660 RETURN
```

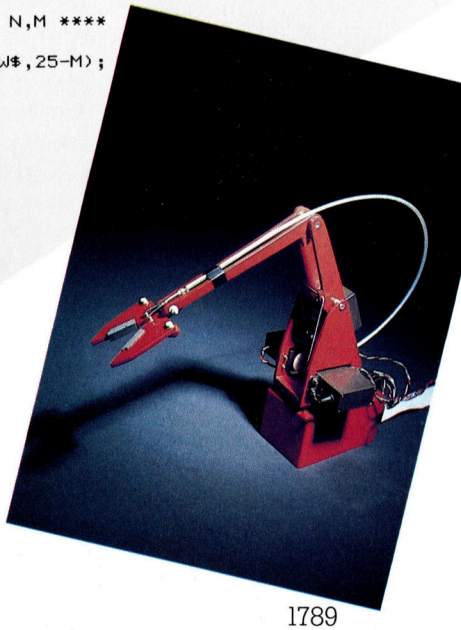
Commodore 64

```
10 REM ************************
20 REM ** CBM CUBIC SPLINES **
30 REM ************************
40 :
50 V=7:FOR I=1 TO 25:DW$=DW$+CHR$(17):NEXT
60 DIM X(V+3),Y(V+3),M(V+3),Z(V+3)
70 X(2)=0:X(3)=5:X(4)=20:X(5)=25
80 Y(2)=2:Y(3)=12:Y(4)=12:Y(5)=2
90 REM ** LINEARLY EXTEND THE ENDS **
100 X(1)=X(2)-(X(3)-X(2))
110 Y(1)=Y(2)-(Y(3)-Y(2))
120 X(6)=X(5)+(X(5)-X(4))
130 X(7)=X(6)+(X(6)-X(5))
140 Y(6)=Y(5)+(Y(5)-Y(4))
150 Y(7)=Y(6)+(Y(6)-Y(5))
160 PRINTCHR$(147)
170 PRINT"DEMONSTRATION OF A ROUTINE WHICH"
180 PRINT"INCORPORATES A CUBIC SPLINE"
190 PRINT"JOINING KNOTS"
200 GOSUB 3000:REM DRAW KNOTS
230 GOSUB 1000:REM SET AKIMA
235 J=2
240 FOR X=X(2) TO X(5)
250 GOSUB2000:REM AKIMA FUNCTION
255 IF X=X(J) THEN J=J+1:GOTO270:REM DON'T DRAW
260 N=X:M=Y:GOSUB5000:PRINT CHR$(46)
270 NEXT
280 GET A$:IF A$="" THEN 280
290 END
1000 REM **** SET AKIMA ****
1010 REM CALC AKIMA COEFS
1020 FOR I=1 TO V-1
1030 M(I+2)=(Y(I+1)-Y(I))/(X(I+1)-X(I))
1040 NEXT I
1050 M(V+2)=2*M(V+1)-M(V)
1060 M(V+3)=2*M(V+2)-M(V+1)
1070 M(2)=2*M(3)-M(4)
1080 M(1)=2*M(2)-M(3)
1090 FOR I=1 TO V
1100 A=ABS(M(I+3)-M(I+2))
1110 B=ABS(M(I+1)-M(I))
1120 IF A+B<>0 THEN Z(I)=(A*M(I+1)+B*M(I+2))/(A+B)
1130 IF A+B=0 THEN Z(I)=(M(I+2)+M(I+1))/2
1140 NEXT I
1150 RETURN
2000 REM **** AKIMA FUNCTION ****
2010 N=1
2020 IF X<X(1) OR X>X(V-2) THEN RETURN:REM CHECK RANGE
2030 I=0
2040 I=I+1:IF X>=X(I) THEN 2040
2050 I=I-1
2060 REM BEGIN INTERPOLATION
2070 B=X(I+1)-X(I)
2080 A=X-X(I)
2090 Y=Y(I)+Z(I)*A+(3*M(I+2)-2*Z(I)-Z(I+1))*A*A/B
2100 Y=Y+(Z(I)+Z(I+1)-2*M(I+2))*A*A*A/(B*B)
2110 RETURN
3000 REM **** PRINT KNOTS ****
3010 FOR I=2 TO 5
3020 N=X(I):M=Y(I):GOSUB 5000:PRINT CHR$(215)
3030 NEXT I
3040 RETURN
5000 REM **** POSITION AT N,M ****
5010 PRINT CHR$(19);
5020 PRINT TAB(N);LEFT$(DW$,25-M);
5030 RETURN
```

Basic-Dialekte

Spectrum:
Zeile 1040 und 1050 des Acorn B-Programms weglassen. Dafür diese Zeilen einfügen:

```
1170 CLS
1220 CIRCLE X(I)*5—4,Y(I)*5+15,10
1270 IF X=X(2) THEN PLOT X*5,Y*5
1280 IF X<>X(2) THEN DRAW X*5,Y*5
```



Wir unterbrechen dieses Programm...

Interrupts geben Betriebssystemen die Möglichkeit, Prozessorkapazitäten des Computers optimal einzusetzen. In dieser Folge über das OS des Acorn B untersuchen wir die Funktionsweise von Interrupts und Ereignissen.

Ein BASIC-Programm, das auf dem Acorn B abläuft, vermittelt leicht den Eindruck, daß die gesamte Maschine sich der Programmausführung widmet. Das ist jedoch nicht der Fall, das Betriebssystem führt auch ständig andere Funktionen aus, etwa die Abfrage der Tastatur. Diese Aufteilung der Computerkapazität wird durch Interrupts (Unterbrechungen) möglich. Interrupts sind im wesentlichen Signale, die die CPU veranlassen, ihre augenblicklichen Aktivitäten anzuhalten und andere Aufgaben auszuführen. Sobald die CPU diese zweite Aufgabe beendet hat, kehrt sie zum ursprünglichen Ablauf zurück und setzt ihn fort. Im Acorn B steuern Interrupts eine ganze Reihe von Aktivitäten: Tastaturabfragen, die Darstellung der Farben, den Ablauf des Befehls ENVELOPE, den Konvertiervorgang des A/D-Wandlers etc. Sehen wir uns diesen Mechanismus einmal an.

Die Interrupts des Acorn B können von mehreren Hardwarekomponenten erzeugt werden, darunter der serielle Schnittstellenchip, die VIAs (Versatile Interface Adaptors – Schnittstellenadaptoren) für Anwender und System sowie die Schnittstellenhardware für Econet und Diskettenlaufwerk. Diese Bauelemente geben zwei verschiedene Interrupttypen an den 6502-Prozessor weiter. Beide unterbrechen den augenblicklichen Ablauf, doch kann ein Interrupttyp von der CPU ignoriert werden. Unterbrechungen dieser Art werden „maskierbare Interruptanforderungen" („Maskable Interrupt Requests" – IRQs) genannt. Der andere Interrupttyp heißt „nicht-maskierbarer Interrupt" („Non-Maskable Interrupt") oder kurz NMI. Er hat eine hohe Priorität und kann von der CPU nicht ignoriert werden.

Die NMIs des Acorn B sind für Systemkomponenten reserviert, die die unmittelbare Aufmerksamkeit der CPU verlangen, wogegen Geräte, die IRQs ausgeben, ruhig etwas warten können. Auf dem Acorn B lösen nur die Econet-Hardware und das Diskettensystem NMIs aus. Da das OS bei Erhalt eines NMI eine Maschinencoderoutine auf Seite &0D anspricht, sollten Sie nach Installation einer Diskettenschnittstelle nichts mehr auf diese Seite schreiben. Wir werden nicht näher auf NMIs eingehen, da der Acorn B sie nur hierfür verwendet.

Wenden wir uns nun den IRQs zu. Auf dem 6502 werden maskierbare Interrupts mit dem Maschinencodebefehl SEI abgeschaltet und mit CLI wieder aktiviert. Wenn der 6502 ein Interrupt erhält, fragt er zunächst alle Interruptquellen ab, ob die Unterbrechung dort ausgelöst wurde. Nach der Identifizierung der Quelle arbeitet der Prozessor die dafür vorgesehene Interruptroutine ab und informiert die Quelle, daß sie die Interruptanforderung nun „vergessen" kann („Clearing the Interrupt"). Maschinencoderoutinen zur Bearbeitung von Interrupts heißen „Interrupt Service Routinen" oder kurz ISR.

ISR sichert die Werte

Da das unterbrochene Programm nach Behandlung des Interrupts mit den gleichen Registerinhalten weiterarbeiten soll, sichert die ISR als erstes diese Werte und schiebt sie auf den Stapel. Nach Behandlung des Interrupts werden sie wieder in ihre Register zurückgeladen. Inzwischen führt die ISR die entsprechende Interruptroutine aus und löscht vor dem Rücksprung in das unterbrochene Programm noch die Interruptbedingung. Der Löschvorgang ist notwendig, da der 6502 sonst unmittelbar nach dem Rücksprung wieder den gleichen IRQ erhalten würde: Ohne Löschung entsteht eine Endlosschleife.

Sehen wir uns diesen allgemeinen Ablauf einmal im konkreten Fall an. Der Acorn setzt Interrupts für viele Vorgänge ein. Aus diesem Grund dürfen Interrupts auch nicht allzu lange mit SEI abgeschaltet werden. Ein paar Millisekunden bringen noch keine Probleme, doch bei längeren Zeiten arbeitet das OS nicht mehr korrekt. Das folgende Programm zeigt, was geschieht, wenn Interrupts für längere Zeit abgeschaltet werden:

```
10 MODE 2
20 DIM C(100):REM Platz für Maschinen-
   code reservieren
30 FOR I%=0 TO 2 STEP 2
40 P%=C
50 [OPT I%
60 .code SEI
70 RTS
```

```
 80 ]:NEXT
 90 COLOUR 14
100 PRINT "Hallo!"
110 ENVELOPE 1,1,4,-4,3,10,20,20,127,0,
    0,-5,126,126
120 SOUND 1,1,160,200
130 TIME=0
140 REPEAT
150 PRINT TIME,I%
160 I%=I%+1
170 IF TIME>100 THEN CALL code
180 UNTIL FALSE:REM Endlosschleife
```

Beim Ablauf dieses Programms werden Sie bemerken, daß sich schon kurz nach Aufruf der Maschinenroutine „code" die Variable TIME nicht mehr verändert, die Farben aufhören zu blinken und die Bearbeitung von ENVELOPE nicht weitergeht. Die Routine gibt aber immer noch das richtige I% aus und zeigt damit, daß die Maschine nicht abgestürzt ist. Nach Beendigung des Programms sollten Sie aber in jedem Fall CTRL BREAK drücken.

Sobald der 6502 ein IRQ erhält, übergibt er die Steuerung an eine ISR, deren Adresse sich im Vektor IRQV1 befindet. IRQV1 liegt bei &204 und &205. Die über diesen Vektor angesprochene Routine prüft zunächst, ob Hardwareinterrupts vorliegen, die das OS routinemäßig benötigt. Die ISR dieses Vektors bearbeitet die Interruptanforderungen dreier Hardwarekomponenten in dieser Reihenfolge: die des seriellen Schnittstellenchips, der System-VIA und einem Teil der Anwender-VIA.

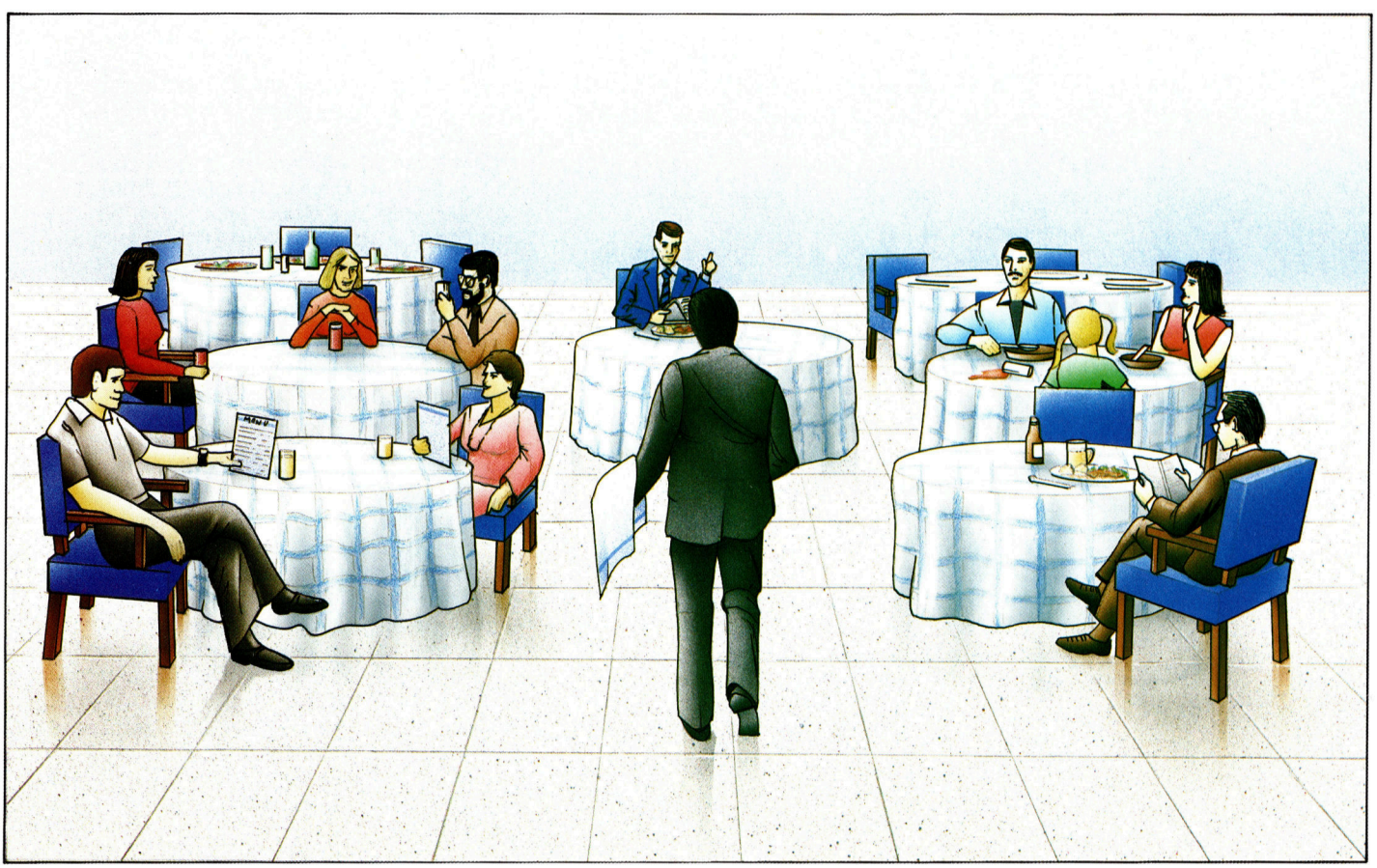
Der serielle Schnittstellenchip ist ein Teil der Hardware und für die Schnittstelle RS423 und das Cassetteninterface zuständig. Interrupts dieser Quellen betreffen natürlich auch diese Systemteile. Der Chip hat bei diesem Vektor höchste Priorität.

Die Interrupts der System-VIA stehen an zweiter Stelle der „Hackordnung". Die außerordentlich flexible VIA verbindet die CPU mit einer Reihe anderer Systemteile und bietet außerdem Möglichkeiten der Zeitbestimmung. Jeder Tastendruck erzeugt ein Interrupt der System-VIA und informiert damit das OS, daß eine Eingabe bearbeitet werden muß. Weiterhin veranlaßt der Chip eine Unterbrechung, wenn ein voller Bildschirminhalt zum Monitor gesandt werden soll. Dieser Interrupt heißt „Vertical Sync Interrupt" und wird jede Fünfzigstelsekunde ausgelöst.

Auch der A/D-Wandler steht mit der System-VIA in Verbindung. Nach jeder Signalumwandlung löst er ein Interrupt aus und veranlaßt damit das OS, die Werte zu aktualisieren, die für die Rückgabe einer Zahl an die Funktion ADVAL nötig sind. Ein weiterer wichtiger Interrupt der System-VIAs findet jede Hundertstelsekunde statt. Er inkrementiert TIME, aktualisiert den „Interval Timer" (der für die Erzeugung von „Ereignissen" eingesetzt wird), dekrementiert den INKEY-Zeitzähler (der die Verzögerung eines INKEY-Befehls mißt) und bearbeitet einen Teil von ENVELOPE. Der Interrupt veranlaßt das OS weiterhin, alle inzwischen betätig-

Vergleichen Sie einmal die CPU des Acorn B mit dem Kellner eines gutbesuchten Restaurants. Wie der Kellner zunächst die wichtigen Gäste (die Priorität vor anderen verlangen) bedient, dann Tische abräumt und gelegentlich auftretende Schwierigkeiten beseitigt, so muß auch der 6502 seine Aufmerksamkeit auf die „Bedienung" der Peripherie, die Steuerung von Datenübertragung und die Programmausführung aufteilen.

Ein geschickter Kellner richtet seine Aufmerksamkeit auf jede dieser Aktivitäten, vergißt dabei aber nicht, seine hungrigen Gäste mit Speisen zu versorgen. Der 6502 arbeitet auf ähnliche Weise. Er antwortet auf Unterbrechungen („Interrupts") oder Ereignisse („Events"), die entweder von der Hardware oder per Software ausgelöst werden. Nach der Behandlung eines Interrupts setzt der Prozessor seine unterbrochene Arbeit fort.



ten Tasten zu bearbeiten. Andere Interrupts der System-VIA betreffen das Sprachsynthesesystem und (falls vorhanden) den Lichtgriffel. Zwar gibt es OSBYTE-Routinen, mit denen Sie die Reaktion des OS auf Interrupts dieser beiden Quellen ändern können, doch werden Sie gleich sehen, welche schwerwiegende Folgen Veränderungen von IRQV1 haben können.

Die Interrupts mit der niedrigsten Priorität kommen von der Anwender-VIA, also der Systemkomponente, die die Kommunikation mit dem User Port und dem Druckerinterface selbständig steuert.

Jedes Interrupt, das von diesem Vektor nicht bearbeitet werden kann, wird an eine ISR von Vektor IRQV2 weitergegeben. Die meisten Interrupts, die diesen Vektor ansprechen, kommen von der Anwender-VIA, die sich mit einer Reihe von Interrupts an die CPU programmieren läßt. Diese Unterbrechungen würden jedoch von der CPU im Normalfall nicht erkannt und ausgeführt werden.

„Intercepting"

IRQV2 liegt bei den Adressen &206 und &207 und läßt sich ändern, um auf die ISR des Anwenders zu zeigen. Obwohl auch IRQV1 geändert werden kann, sollten Sie für Ihre ISR nur IRQV2 einsetzen. Die Umstellung eines Vektors auf eine andere Routine heißt „Intercepting" (abfangen) eines Vektors. Um den IRQV2-Vektor abfangen zu können, müssen beim Einsprung in eine selbsterstellte Routine folgende Bedingungen erfüllt sein:

- Die CPU muß den Inhalt des A-Registers bereits in &FC gespeichert und das Prozessor Status Register (PSR) auf den Stapel geschoben haben.
- Die Register X und Y sollen in dem Zustand sein, in dem sie sich vor Auftreten des Interrupts befanden.

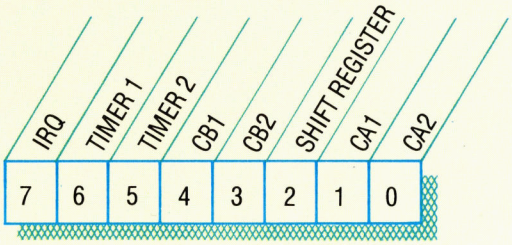
Punkt zwei besagt, daß Sie die X- und Y-Register beim Eintritt in Ihre ISR mit folgenden Befehlen sichern müssen:

```
TXA:PHA
TYA:PHA
```

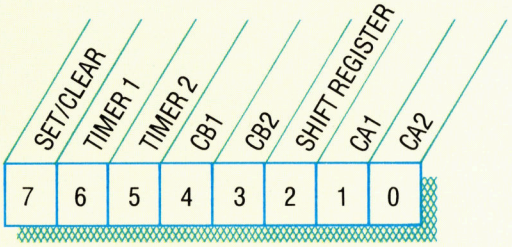
Danach sollte Ihre Routine die Interruptbedingung löschen. Dieser Vorgang hängt jedoch von der Quelle der Unterbrechung ab. Bei der Anwender-VIA müssen dafür gewöhnlich bestimmte Register der VIA gelesen oder geschrieben werden. Denken Sie auch daran, vor der Rückkehr in das unterbrochene Programm die Register wieder in ihren ursprünglichen Zustand zu versetzen und die ISR mit dem Befehl RTI zu verlassen. Noch ein letzter Hinweis: Obwohl das A-Register beim Sprung auf IRQV1 in &FC gespeichert wurde, kann ein weiterer Interrupt durchaus veranlaßt haben, daß dieser Wert vor Beendigung Ihres eigenen Interrupts überschrieben wurde.

Interrupts des VIA-Chips

Flag-Register der Interrupts



Register zur Freigabe von Interrupts



Obwohl der 6502-Prozessor des Acorn B nur zwei Interruptleitungen – IRQ und NMI – besitzt, läßt sich der Schnittstellenchip zur Peripherie-VIA so programmieren, daß sich mehrere Systemkomponenten die gleiche IRQ-Interruptleitung des Prozessors teilen können. Jeder VIA-Chip hat zwei E/A-Ausgänge im Acht-Bit-Format, von denen jeder wiederum über acht Datenleitungen und je ein Paar Signalleitungen verfügt. Die Leitungen von Ausgang A heißen CA1 und CA2 und die von Ausgang B CB1 und CB2. Sie werden oft für das „Handshaking" (die Synchronisation der Datenübertragung zwischen Micro und einem Peripheriegerät) eingesetzt. Die Leitungen können aber auch als Interruptleitungen programmiert werden. Zusätzlich zu den vier möglichen Interruptleitungen jeder VIA gibt es ein internes Interrupt, das sich über das Shift-Register des Chips generieren läßt. Dieses Register steuert die serielle Datenübertragung, die von externen, über CB1 hereinkommenden Impulsen ausgelöst wird und über CB2 abläuft. Schließlich gibt es noch zwei Intervalltimer, die ebenfalls Interrupts erzeugen können. All diese Interruptquellen sind mit der einzigen IRQ-Leitung des Prozessors verbunden.

Zur Bearbeitung eines Interrupts muß der Prozessor die Quelle kennen. Die CPU fragt dazu in der VIA das Flag-Register für Interrupts ab. Jedes Bit dieses Registers entspricht einer der potentiellen Interruptquellen und wird auch von der Interruptanforderung dieser Systemkomponente gesetzt. Eine Ausnahme ist Bit 7, das nur gesetzt wird, wenn ein Peripheriegerät ein Interrupt anfordert.

In manchen Situationen muß verhindert werden, daß Interrupts ausgegeben werden, während der Prozessor gerade einen anderen Interrupt bearbeitet. Diese Information kann in das Register zur Freigabe von Interrupts („Interrupt Enable Register" – IER) programmiert werden.

Fachwörter von A bis Z

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

Macro = Makrobefehl
Ein Makrobefehl oder kurz Makro ist ein einzelner Befehl innerhalb eines Programms, der für eine ganze Anweisungsfolge steht. Sie wird an anderer Stelle definiert und ausgeführt, wenn das Programm auf den „Namen" des Makros trifft. Makros werden vor allem bei der Assemblerprogrammierung verwendet.

Ein „Makro-Assembler" erlaubt die Definition eines Makro, für die dann im Assemblerprogramm nur ein einziger Befehl erforderlich ist. Insofern sind Makros mit den Prozeduren beim strukturierten BASIC oder bei LOGO verwandt. Die Prozedur wird jedoch als Unterprogramm ausgeführt, während ein Makro bei der Assemblierung jedesmal durch die Folge der zuvor definierten Anweisungen ersetzt wird. Wie ein Prozedurname kann auch ein Makroname, wenn er einmal vereinbart ist, im Programm beliebig oft Verwendung finden.

Magnetic Card = Magnetkarte
Magnetkarten wurden Ende der sechziger Jahre als Speichermedium für die Daten- und Textverarbeitung verwendet, insbesondere bei IBM in Verbindung mit den „Selectric"-Schreibmaschinen. Die Karten aus Kunststoff oder starkem Papier hatten etwa das Format einer Lochkarte und waren ähnlich wie Disketten und Bänder mit einem ferromagnetischen Material beschichtet. Die gespeicherte Information konnte durch einen speziellen Kartenleser wieder entziffert und mit der Schreibmaschine ausgedruckt werden.

Heute sind Magnetkarten vorwiegend als Scheck- und Kreditkarten in Gebrauch. Ein Magnetbandstreifen auf der Kartenrückseite enthält in codierter Form den Namen des Inhabers und seine Bankverbindung mit Kontonummer. Anhand dieser Daten können Bankgeschäfte über Automaten abgewickelt werden. Kleine Kartenleser gibt es auch für Einzelhändler, um Kreditkarten rechnergestützt mit der Liste gestohlener oder verlorgengegangener Karten zu vergleichen und um Zahlungsleistungen sofort über ein Telefonmodem durch einen Zentralcomputer der Bank bestätigen zu lassen.

Verbreitet ist die Verwendung von Magnetkarten bei Bargeldautomaten. Der Magnetstreifen auf der Karte enthält eine Codenummer, die elektronisch gelesen und mit den Ziffern verglichen wird, die der Benutzer eintastet. Stimmen die Zahlen überein, wird der Zugang zur Geldausgabe freigegeben.

Magnetic Disk = Magnetplatte, Diskette
Magnetplatte und Diskette sind Datenträger in Gestalt einer rotierenden Scheibe (Disk), deren Oberfläche mit einer dünnen Aufzeichnungsschicht aus magnetisierbarem Material überzogen ist. Bei der flexiblen Diskette oder ‚Flopy Disk' ist die Platte selbst aus elastischem Kunststoff, bei der starren Magnetplatte oder ‚Hard Disk' im allgemeinen aus einer Aluminiumlegierung. Für die Beschichtung wird bei Disketten meist Eisenoxid verwendet, bei Starrplatten oft auch ein Kobalt-Nickel- oder ein Kobalt-Chrom-Film. Der Scheibendurchmesser variiert von 7,6 cm bei der 3-Zoll-Diskette bis zu 91,4 cm bei Festplatten für Großrechner.

Bildnachweise
1765: Darrell Huff
1778, 1769: Caroline Clayton
1774, 1775, 1786: Ian McKinnell
1777–1779: Crispin Thomas
1783: Mike Clowes
1784: MS-Fortran, Computer World
1785: Alan Adler
1787: BBC Open University Production Centre
1788, 1789, 1791: Kevin Jones
1792: Liz Dixon

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

computer kurs

Heft 65

Der Texter
Hält das preisgünstige Textverarbeitungssystem Joyce von Schneider, was sein Name verspricht? Zum Lieferumfang gehören immerhin Diskettenlaufwerk, Drucker und Software.

Höret und staunet!
Den Computer als Musikinstrument einzusetzen ist für Profis und Amateure gleichermaßen interessant.

COBOL für Micros
Selten auf Micros eingesetzt. Wir zeigen die vier Hauptabschnitte von COBOL.

Zeit für Go
Das orientalische Brettspiel Go hat das Interesse der KI-Wissenschaftler geweckt. Wir erklären die Regeln und beginnen mit der Programmierung des Spiels.