Einsteigen - Verstehen - Beherrschen

DM 3,80 öS 30 sfr 3,80

computer kurs

Heft 34

Mobil: PC Compaq Plus

Schreiben für den Schirm

Menüsteuerung per Mauspaket

Maschinencode: Binäre Division

Lenken zweier Motoren

Ein wöchentliches Sammelwerk

computer kurs Heft 34

Inhalt

Computer Welt

Werden Sie überwacht? 925
Der Große Bruder hält seinen Einzug

Schreiben für den Schirm 949
Professionelle Software-Entwicklung

Imagepflege 952

BASIC 34

Völlig aufgelöst 927
Hochauflösende Grafiken des Commodore 64

Computer-Logik

Karnaugh-Tafeln 930
Vereinfachung der Booleschen Algebra

Tips für die Praxis

Doppelte Kraft 932
Steuerung eines zweimotorigen Autos

Software

Software-Symphonie 935
Integrierte Programme von Lotus und Psion

Handel im All 948
Ein abenteuerliches Strategiespiel

Hardware

Kompakt verpackt 937
Ein tragbarer PC: der Compaq Plus

Bits und Bytes

Geteilte Zahlen 940
Binäre Divisionen mit der Maschinensprache

Peripherie

Piktogramme 944
Das AMX-Maus-Paket für den Acorn B

PASCAL

Datentypen 946
Vier verschiedene Variablen

Fachwörter von A—Z

WIE SIE JEDE WOCHE IHR HEFT BEKOMMEN

Computer Kurs ist ein wochentlich 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 konnen 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 gewunschten 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 Bei Bestellungen aus Österreich oder Schweiz senden Sie Ihren Auftrag bitte auch an die Hamburger Adresse Berechnung und Zahlung erfolgen in Landeswährung zum Ladenpreis

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 Zeitschriftenhandler 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 fullen 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

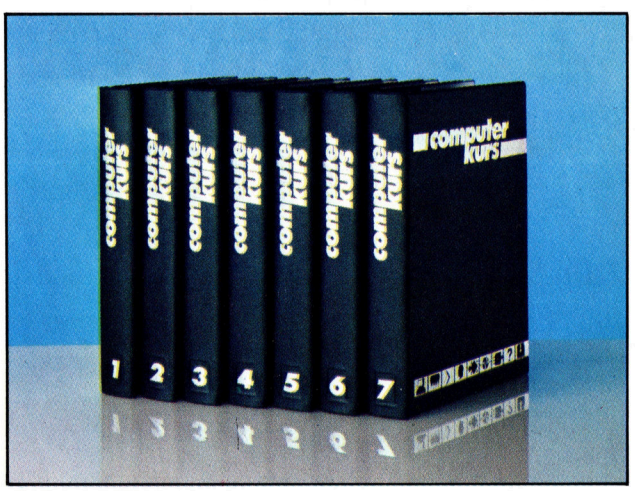
Schweiz: Der Sammelordner kostet sfr 15 Bitte wenden Sie sich an Ihren Kiosk; dort werden Sie jederzeit die gewunschten Exemplare erhalten

INHALTSVERZEICHNIS

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

Redaktion: Winfried Schmidt (verantw f d Inhalt), Elke Leibinger, Susanne Brandt, Uta Brandl (Layout), Sammelwerk RedaktionsService 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

Werden Sie überwacht?

Einerseits stürzen sich Polizisten weltweit auf jugendliche „Hacker", weil diese Computercodes knacken, andererseits sind dieselben Sicherheitskräfte in ähnlich fragwürdige Aktivitäten verstrickt.

Ganz offiziell hält die Computerüberwachung überall auf der Welt in immer mehr Bereiche des täglichen Lebens Einzug. Alles deutet darauf hin, daß wir alle irgendwann in irgendein System einbezogen sein werden.

Einige dieser Überwachungseinrichtungen sind harmlos und nur für die Verwaltungsstellen wichtig, so beispielsweise die Registrierung der Fahrzeuge und Führerscheine. In Deutschland speichert das Kraftfahrzeug-Bundesamt (Flensburg) die Daten der Autofahrer. Diese Daten sind in England in den Computern der DVLC gespeichert und die Daten zur Sozialversicherung im DHSS-System. Man denkt mittlerweile daran, diese unterschiedlichen Systeme so miteinander zu verbinden, daß die Datenbanken des DVLC-Systems und des DHSS-Systems mit dem Police National Computer in Hendon zusammenarbeiten, das heißt, Daten untereinander austauschen können. Damit hätten die Behörden eine größere Möglichkeit, die gesamte Bevölkerung zu überwachen.

Das Datenschutzgesetz (1978 in der Bundesrepublik in Kraft getreten) wurde verabschiedet, um den damit verbundenen möglichen Machtmißbrauch zu verhindern. Angesichts der schnellen Weiterentwicklung der Computertechnik seit Verabschiedung des Gesetzes gibt es viele Stimmen, die sagen, das Gesetz sei bereits überholt.

Viele der bei der Computerüberwachung und bei Sicherungssystemen verwendeten Techniken bedienen sich der Mustererkennung („Rasterfahndung"). Dies ist ein Verfahren, bei dem der Computer das, was er „sieht", mit bereits gespeicherten Mustern vergleicht.

Das beste Beispiel dafür ist das neue Fingerabdrucksystem, das von Logica für die Metropolitan Police bei Londons New Scotland Yard installiert wurde. Fünfzehn Jahre waren zur Entwicklung des Systems erforderlich, das 650 000 Fingerabdrücke und 100 000 „Spuren" – Teilabdrücke – speichern kann, die an Tat-

Es gibt viele Stellen, an denen Abhöreinrichtungen versteckt werden können. In diesem Büro könnten an folgenden Stellen Wanzen sein:
1) Im Telefon. In der Sprechmuschel, im Telefongehäuse oder in der Schreibtischlampe könnte sich die Wanze befinden.
2) Topfpflanze. Hier könnte die Wanze in der Erde oder unter dem Topf verborgen oder gar als echte Wanze getarnt sein.
3) Wand. Ein Nagel könnte eine Wanze beherbergen. Beliebt sind auch Korkwandverkleidungen, hinter denen man Wanzen verstekken kann.
4) Wandbilder. Wanzen werden gern als „Haken" getarnt oder hinter dem Bild versteckt.
5) Schreibtisch. Unter der Schreibtischauflage, in einer Schublade oder unter der Tischplatte könnte die Wanze kleben.

orten gefunden wurden. Das System vergleicht einfach die gefundenen mit den gespeicherten Abdrücken, um zu sehen, ob es irgendwelche Übereinstimmungen gibt. Dafür sind Prime-Minicomputer erforderlich, ergänzt um hochleistungsfähige Prozessoren, Monitore und Fernsehkameras.

Ausstattung für Agenten

Computerisierter Schutz
Manche Menschen, die mit wichtigen Informationen umgehen, schützen sich vor elektronischer Spionage mit Geräten wie diesem Telefon-Zerhacker auf Computerbasis. Statt ins Telefon zu sprechen, wird die Nachricht über die Tastatur eingegeben. Der Zerhakker sendet den Text „stumm" über die normale Telefonleitung. Der Empfänger verfügt über ein entsprechendes Gerät und sieht die Nachricht auf dem eingebauten Bildschirm. Ein integrierter Stimm-Synthesizer kann die eintreffende Nachricht sogar auf Knopfdruck in eine gesprochene Sprache umwandeln.

Codierte Nachricht
Ähnlich dem computerisierten Zerhacker wird bei diesem System eine handgeschriebene Notiz, die verschlüsselt wurde, gesendet. Da auch diese Übertragung „stumm" erfolgt, kann sie nicht abgehört werden. Auf diese Weise können sogar Unterschriften übermittelt werden.

Streßanalysator
Der Stimmen-Streßanalysator mißt den Streßfaktor einer Stimme und gibt die ermittelten Werte in einfacher numerischer Form aus. Es handelt sich dabei um einen verfeinerten Lügendetektor, der auch feststellen kann, ob eine Person ängstlich oder sehr angespannt ist.

Ein ähnliches Mustererkennungs- und Vergleichssystem ist auf einer Brücke über einer Autobahn Englands installiert. Mehrere Kameras sind auf die Spuren ausgerichtet. Das System zeichnet Bilder der Nummernschilder sich nähernder Fahrzeuge auf, die mittels Computer analysiert werden. Danach erfolgt ein Vergleich dieser Nummern mit der Datei der gesuchten Fahrzeuge. Die Information, daß eines dieser Fahrzeuge gesehen wurde, kann dann durch Funk an die Autobahnpolizei weitergegeben werden.

Wanzen auf der Lauer

Ein Bereich, auf den die Entwicklungen der Microcomputertechnik besondere Auswirkungen hatten, war die Produktion immer kleinerer Überwachungsgeräte – also der „Wanzen". Die Chiptechnik ermöglicht es, Radiosender von der Größe eines Reiskorns herzustellen, die perfekte elektronische Steuerelemente enthalten. Einer dieser Gerätetypen „zieht" beispielsweise den nötigen Strom aus dem Telefonnetz der Post, und er schaltet sich nur ein, wenn wirklich jemand spricht. Es gibt auch batteriebetriebene, ebenso kleine Wanzen, die in der Ecke eines Raumes deponiert werden, sämtliche Gespräche in diesem Raum aufnehmen und an einen entfernt aufgebauten Empfänger weiterleiten.

Die „Tele-Wanze" entspricht mehr dem James-Bond-Stil. Hierbei wird ein Laserstrahl auf ein Fenster gerichtet. Die durch die Unterhaltung verursachten Schwingungen des Glases werden als Interferenzmuster im reflektierten Laserstrahl aufgefangen, und die Sprachinformation wird mit Computerhilfe aus dem Interferenzmuster herausgelesen.

Derzeit sind die bei der Überwachung und Sicherung eingesetzten Computer noch sehr groß, etwa in der Art der aufwendigen Rechner, die zur Decodierung beim CIA oder der National Security Agency in den USA verwendet werden. Die Weiterentwicklung der Hardware bedeutet jedoch, daß Fingerabdruck-Erkennung oder gar Gesichtererkennung sehr bald automatisiert und billig sein werden.

Es ist vorstellbar, daß künftig Polizeiautos mit Bordcomputern ausgestattet sind, denen der Zugriff zu allen Datenbanken wie Verbrecherdatei, Krankendatei und Sozialversicherungsdatei möglich ist. All das wäre durch einfaches Einstecken einer Plastikkarte in einen Schlitz am Computer möglich. Der Rechner würde die codierten Informationen lesen, mit den Zentraldateien vergleichen und die Identität des Verdächtigen bestätigen.

Das mutet wie eine schreckliche Vision an, die Technik aber ist schon fast perfekt, um das zu verwirklichen. Der „fälschungssichere" neue Ausweis (so groß wie eine Scheckkarte), der bald in Deutschland eingeführt wird, ist ein erster Schritt in diese Richtung.

Völlig aufgelöst

In diesem Teil des BASIC-Kurses wird erklärt, wie man die hochauflösende Grafik des Commodore 64 ansprechen kann.

Bei normaler Auflösung ist der Bildschirm des Commodore 64 in 25 Reihen mit je 40 Zeichen unterteilt, was insgesamt 1000 Positionen ergibt. Jedes Zeichen besteht wiederum aus 64 Pixeln. Bei hoher Auflösung müssen wir jedes einzelne Pixel durch ein Steuerbit ein- bzw. ausschalten. Diese Methode ist unter dem Begriff „bit mapping" bereits bekannt. Da jede Speicherstelle acht Bits enthält und der Bildschirm aus 64000 Pixeln besteht, werden 8000 Speicherstellen benötigt, um die hochauflösenden Bildschirminformationen zu speichern.

Der Commodore 64 wird in den hochauflösenden Modus umgeschaltet, indem Bit 5 der Speicherstelle 53265 auf 1 gesetzt wird. Um dieses Bit zu setzen, ohne andere Biteinstellungen zu verändern, sollten Sie die folgende Anweisung verwenden:

```
POKE53265,PEEK(53265)OR32
```

Der Bildschirm erhält nun seine Informationen aus einem 8000 Bytes großen Speicherblock. Der Beginn dieses Speicherblocks liegt bei Adresse 53272.

Der Bereich des Speichers, der normalerweise die Informationen für den Bildschirm enthält, wird für die Farbinformationen jeder Acht-mal-acht-Zelle auf dem Bildschirm verwendet. Die 16 verfügbaren Farben des Commodore 64 können durch nur vier Bits repräsentiert werden. Somit werden die vier oberen Bits jeder Position im Bildschirmspeicher verwendet, um die Farbe der Pixel, die in einer bestimmten Zeichen-Zelle „an" sind, zu speichern, und die unteren vier Bits repräsentieren die Farbe der Pixel, die „aus" sind. Dadurch ist es möglich, verschiedene Farbpaare für jede Zelle zu generieren. Wenn man einen violetten Hintergrund für den gesamten Bildschirm und eine hochauflösende schwarze Grafik als Vordergrund wünscht, benötigt man die folgenden Codes:

Farbcode für schwarz ist 0 = 0000 binär
Farbcode für violett ist 4 = 0100 binär

Fügt man diese beiden Teile zusammen, erhält man 00000100 oder dezimal 4. POKEt man den Wert 4 in jede Bildschirmspeicher-Adresse (1024 bis 2023), erhält man die gewünschte schwarze Grafik auf einem violetten Hintergrund.

Bevor wir beginnen können, auf dem hochauflösenden Bildschirm zu zeichnen, muß der 8000 Bytes große Speicherblock, der die Ausgabe kontrolliert, gelöscht werden. Dies geschieht, indem man eine 0 in jede Adresse POKEt. Löscht man den Speicher nicht, entsteht auf dem Bildschirm ein wirres Durcheinander von Punkten. Das kommt daher, weil bestimmten Speicherbereichen beim Einschalten des Computers zufällige Zahlenwerte zugeordnet werden.

Ein Programm für hochauflösende Grafik muß individuelle Pixel auf dem Bildschirm ansprechen können. Wenn jeder Punkt eine X- und eine Y-Koordinate erhält, kann das Programm identifizieren, welches Bit in der 8000 Byte umfassenden Speicherkarte auf 0 oder 1 gesetzt werden muß.

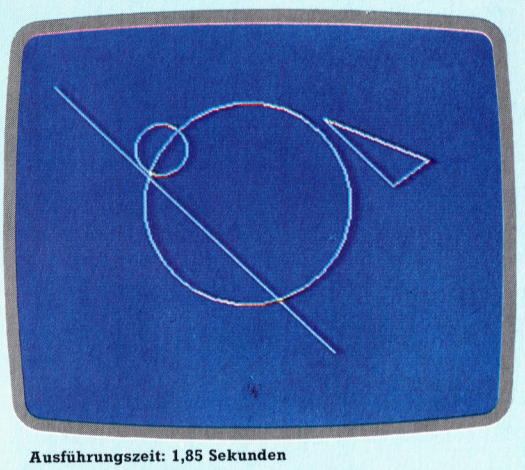
Die horizontale Position eines Bytes kann durch folgenden Befehl über die X-Koordinate ermittelt werden:

```
HB = INT(X/8)
```

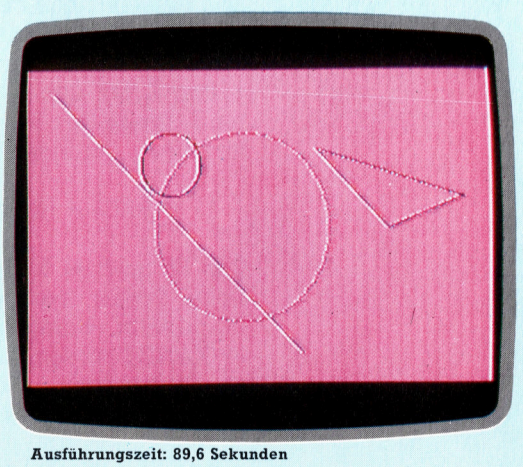
Relative Geschwindigkeit

Um diese Darstellung zu erzeugen, brauchte der Commodore 64 annähernd 90 Sekunden und über 50 Zeilen Programmcode. Die Erstellung desselben Bildes dauerte auf dem Spectrum nur zwei Sekunden und erforderte das folgende kurze und einfache Programm:

```
4000 REM * HiRes Demo *
4050 LET N=18: DIM U(N).
     INK 6: PAPER 1: BORDER 1
     RESTORE 4100
4100 DATA 20,160,160,-160
4120 DATA 80, 123, 15
4140 DATA 130,90,60
4160 DATA 175,140
4180 DATA 40, -40
4200 DATA 20,15
4220 DATA -60,25
4250 FOR K=1 TO N: READ
     U(K): NEXT K
4300 PLOT (U(1),U(2):
     DRAW U(3), U(4)
4350 CIRCLE U(5),U(6),U(7)
4400 CIRCLE U(8),U(9),U(10)
4450 PLOT U(11),U(12):
     DRAW U(13),U(14)
4500 DRAW U(15),U(16):
     DRAW U(17),U(18)
4600 PAUSE 0
```

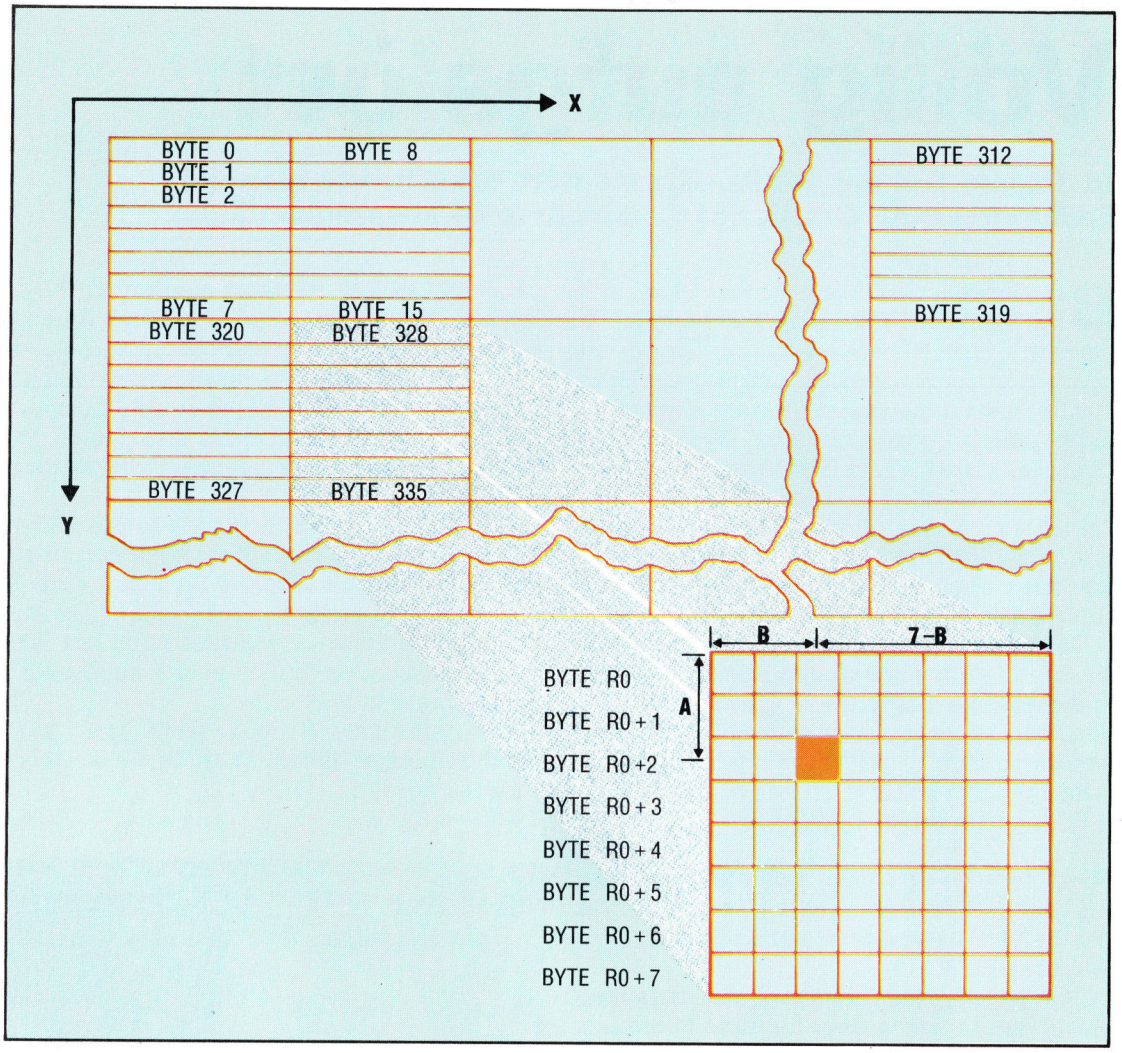


Ausführungszeit: 1,85 Sekunden



Ausführungszeit: 89,6 Sekunden

Auf die einzelnen Pixel, die die hochauflösende Grafik des Commodore 64 produzieren, kann nicht direkt zugegriffen werden: Der 40x25-Text-Modus wird in 8000 Bytes dargestellt, wobei jede Textposition durch acht Bytes repräsentiert wird. Die Entfernung eines Pixels vom linken Rand des Bildschirms wird im hochauflösenden Modus durch X beschrieben und mit Y seine Distanz vom oberen Rand des Bildschirms. Die Zahlen müssen in die Adresse des Bytes umgewandelt werden, das die Pixel beinhaltet, sowie in die Nummer des relevanten Bits innerhalb des Bytes.



Ähnlich kann das vertikale Byte mit Hilfe der Y-Koordinate gefunden werden:

```
VB = INT(Y/8)
```

Das erste Byte der Zeichenposition, das das benötigte Bit beinhaltet (RO), kann aus HB und VB errechnet werden:

```
RO = VB*320+HB*8
```

Das Byte, das das benötigte Bit enthält, ist RO, zuzüglich dem Rest, der sich aus Y/8 ergibt. Wenn A=YAND7 und BASE die Adresse des ersten Bytes des 8000-Bytes-Blocks ist, kann die Adresse des Bytes (BY), das das gesuchte Bit enthält, gefunden werden:

```
BY = BASE + RO + A
```

Das Bit im Byte BY kann durch den Rest, der beim Teilen der X-Koordinate durch acht übrigbleibt, herausgefunden werden. Wenn B=XAND7 ist, setzt der folgende POKE-Befehl das Bit auf eins, das dem Pixel mit den Koordinaten X und Y entspricht:

```
POKE BY,PEEK(BY)OR(2↑ (7-B))
```

Das folgende Programm zeigt, wie gerade Linien von einem Punkt (X1,Y1) zu einem anderen Punkt (X2, Y2) gezeichnet werden können. So könnte man auch einen Kreis zeichnen, indem man die Koordinaten seines Zentrums (CX,CY) und seines Radius RA bestimmt. In dem Programm gibt es auch eine Unterroutine, die ein Dreieck zeichnet, indem die Koordinaten seiner drei Ecken (XA,YA), (XB,YB) und XC,YC) vorgegeben werden. Sie können nun mit dem Programm experimentieren, indem Sie die vorgegebenen Koordinaten ändern.

Die Struktur dieses Programms besteht aus einer Anzahl aneinandergereihter Unterroutinen. Die einfachste zeichnet einen einzigen Punkt auf den Bildschirm. Eine der komplexesten Routinen ist die PLOT-TRIANGLE-Routine, die die PLOT-LINE-Routine dreimal verwendet, um die drei Seiten zu konstruieren.

Es ist ratsam, das Programm zu speichern, bevor Sie es starten. Überprüfen Sie die POKEs sorgfältig, da eine falsch eingegebene Adresse zum „Absturz" des Rechners führen kann. Wenn Sie nach Beendigung des Programmlaufs wieder den normalen Grafik-Modus aktivieren wollen, müssen Sie die Tasten RUN/STOP und RESTORE genau gleichzeitig drücken.

```
65 REM  **** HI-RES DEMO ****
70 PRINT CHR$(147): REM CLEAR SCREEN
80 POKE 53280,0 : REM COLOUR BORDER BLACK
90 :
100 REM **** COLOUR SCREEN MEMORY AREA ****
110 FOR I=1024 TO 2023: POKE I,4: NEXT I
120 :
130 REM **** SET BIT MAP POINTER ****
140 BASE =8192: POKE 53272, PEEK(53272) OR 8
150 :
160 REM **** CLEAR BIT MAP MODE ****
170 FOR I=BASE TO BASE + 7999:POKE I,0:NEXT I
180 :
190 REM **** SET BIT MAP MODE ****
200 POKE 53265, PEEK(53265) OR 32
210 :
220 REM **** DRAW STRAIGHT LINE ****
230 X1=20: X2=190: Y1=15: Y2=180
240 GOSUB 800: REM PLOT LINE
250 :
300 REM **** DRAW CIRCLE ****
310 CX=150: CY=100: RA=60
320 GOSUB 900: REM PLOT CIRCLE
330 :
370 **** ANOTHER CIRCLE ****
380 CX=100: CY=60: RA=20
390 GOSUB 900: PLOT CIRCLE
400 :
410 REM **** DRAW TRIANGLE ****
420 XA=200:XB=250:XC=300:YA=50:YB=100:YC=80
430 GOSUB 600: REM PLOT TRIANGLE
440 :
450 GOTO 450: REM END OF MAIN PROGRAM
460 :
470 :
600 REM **** PLOT TRIANGLE SUBROUTINE ****
610 :
620 X1=XA: X2=XB: Y1=YA: Y2=YB
630 GOSUB 800: REM PLOT LINE
640 X1=XB: X2=XC: Y1=YB: Y2=YC
650 GOSUB 800: REM PLOT LINE
660 X1=XC: X2=XA: Y1=YC: Y2=YA
670 GOSUB 800: REM PLOT LINE
680 RETURN
690 :
800 REM ***** PLOT LINE SUBROUTINE *****
810 S=1
820 IF X2<X1 THEN S=-1
830 FOR X=X1 TO X2 STEP S
840 Y=(Y2-Y1)*(X-X1)/(X2-X1)+Y1
850 GOSUB 1000: REM PLOT POINT
860 NEXT X
870 RETURN
880 :
900 REM **** PLOT CIRCLE SUBROUTINE ****
910 :
920 FOR ANGLE = 0 TO 2*PI STEP .04
930 X=INT (RA*COS(ANGLE)+CX)
940 Y=INT (CY-RA*SIN(ANGLE))
950 GOSUB 1000: REM PLOT POINT
960 NEXT ANGLE
970 RETURN
980 :
1000 REM **** PLOT POINT SUBROUTINE ****
1010 :
1020 IF X>319 OR X<0 OR Y>199 OR Y<0 THEN
GOTO 1070
1030 HB=INT(X/8): VB=INT(Y/8)
1040 RO=VB*320+HB*8: A= Y AND 7: B=X AND 7
1050 BY=BASE+RO+A
1060 POKE BY, PEEK(BY)OR(2^(7-B))
1070 RETURN
```

U-BOOT-JAGD

Ein wichtiger Teil unseres Spieles ist die Routine, die die Punktzahl des Spielers aktualisiert. Unser Punkte-System basiert auf den folgenden Regeln:

1) Die Tiefe und Geschwindigkeit des U-Bootes sind wichtige Faktoren. Ein in großer Tiefe mit hoher Geschwindigkeit fahrendes U-Boot ist schwerer zu treffen als ein langsames U-Boot dicht an der Wasseroberfläche. Die Punktzahlen, die jedem U-Boot zugeordnet werden, berücksichtigen diesen Umstand.

2) Wenn das U-Boot getroffen wird, wird sein Wert zu der Punktzahl des Spielers addiert. Erreicht ein U-Boot jedoch unbeschädigt den Rand des Bildschirmes, wird sein Wert von der Punktzahl des Spielers subtrahiert.

Zu einem späteren Zeitpunkt werden wir uns mit der Routine befassen, die die Geschwindigkeit und Tiefe eines U-Bootes zufällig auswählt. Die Tiefe eines U-Bootes wird in der Variablen Y3 und die Geschwindigkeit in der Variablen DX gespeichert. Auf dieser Basis kann der Wert eines U-Bootes berechnet werden. Um zu gewährleisten, daß diese beiden Werte immer ganze Zahlen sind, wird die INT-Funktion verwendet:

U-Boot-Wert = INT(Y3+DX*30)

Den jeweils aktuellen Punktestand eines Spielers speichern wir in der Variablen SC. Alles was jetzt noch zu machen ist, besteht darin, den Wert eines U-Bootes mit SC zu addieren oder zu subtrahieren, je nachdem, ob das U-Boot getroffen wurde oder nicht. Die UPDATE-SCORE-Unterroutine wird von zwei Programmteilen verwendet:

1) Wenn die Position des U-Bootes überprüft wird, um festzustellen, ob es den Rand des Bildschirmes erreicht hat, und

2) in der HIT-Routine.

Das Flag DS kann während der Ausführung beider Teile gesetzt werden, um so anzuzeigen, welcher Teil die UPDATE-SCORE-Unterroutine verwendet. Wird DS in der HIT-Routine auf 1 gesetzt und auf -1 in der EDGE-OF-SCREEN-Routine, kann die Punktzahl um den Wert des U-Bootes wie folgt erhöht oder verringert werden:

SC = SC + INT (Y3+DX*30)*DS

Nach einer Feststellung, daß die Punktzahl nicht unter 0 liegt, kann die neue Punktzahl in die erste Zeile des Bildschirmes gePRINTet werden. Fügen Sie die folgenden Zeilen in Ihr Programm ein:

```
5500 REM **** UPDATE SCORE ****
5510 SC=SC+INT(Y3+DX*30)*DS
5520 IF SC < 0 THEN SC=0
5530 PRINT CHR$(19);CHR$(144);"SCORE";
     SC;CHR$(157);" "
5540 RETURN
```

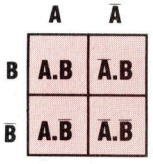
Im nächsten Teil des Kurses werden wir uns mit der Entwicklung der Sprites beschäftigen, die für das Schiff, das U-Boot, die Wasserbomben und die Explosionen benötigt werden.

Karnaugh-Tafeln

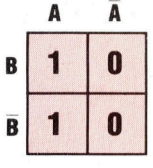
Karnaugh-Tafeln sind bei der Vereinfachung von Logikschaltungen unentbehrlich. Dabei ersetzen sie nicht die bereits behandelten rechnerischen Methoden, können aber eine große Entlastung sein, wenn schwierige Ausdrücke der Booleschen Algebra mit mehreren Variablen vereinfacht werden sollen.

Karnaugh-Tafeln (auch als k-Tafeln bezeichnet) sind eine Erweiterung der Venn-Diagramme, die schon aus früheren Kursabschnitten bekannt sind. Sie dienen zur bildlichen und vereinfachten Darstellung logischer Ausdrücke. Die Form einer solchen Tafel wird durch die Menge der Buchstaben oder Variablen der Funktion bestimmt. Besonders gut läßt sich mit k-Tafeln für zwei, drei oder vier Variablen arbeiten.

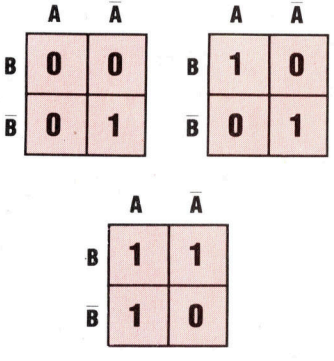
Zwei Variablen: Jede Zelle einer k-Tafel für zwei Variablen entspricht einer AND-Funktion, wie sie hier im Bild gezeigt wird:



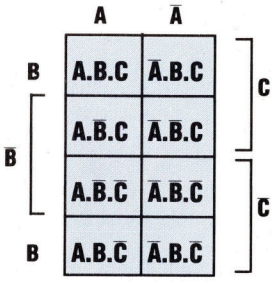
Um den Ausdruck $AB + A\overline{B}$ als k-Tafel darzustellen, werden Einsen an den entsprechenden Stellen eingetragen:



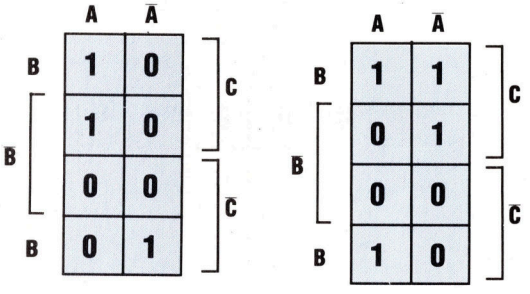
Hier drei weitere Beispiele, die entsprechend die Funktionen $\overline{A}\overline{B}$, $AB+\overline{A}\overline{B}$ und $AB + \overline{A}B + A\overline{B}$ darstellen:



Drei Variablen: In diesem Fall nimmt die Anzahl der Zellen um den Faktor Zwei zu ($2^3 = 8$ Zellen). Die Basis-k-Tafel für 3 Variablen:



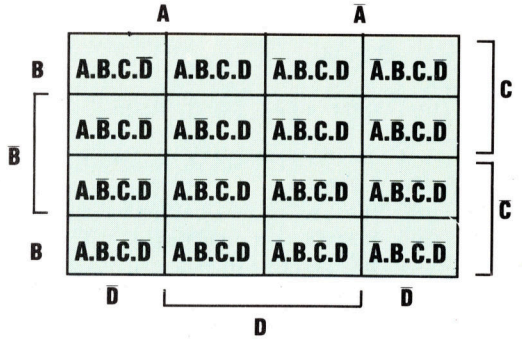
Diese k-Tafeln verkörpern die Ausdrücke $AC + \overline{A}B\overline{C}$ und $AB + \overline{A}C$:



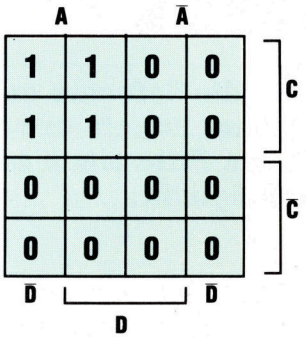
$ABC + A\overline{B}C + \overline{A}B\overline{C}$
$= AC(B + \overline{B}) + \overline{A}B\overline{C}$
$= AC + \overline{A}B\overline{C}$
$ABC + AB\overline{C} + \overline{A}BC + \overline{A}\overline{B}C$
$= AB(C + \overline{C}) + \overline{A}C(B + \overline{B})$
$= AB + \overline{A}C$

Beachten Sie, daß beide Ausdrücke mit dem Booleschen Gesetz der Universalität oder Identität vereinfacht wurden – ein A, das durch OR mit seiner Negation ($\overline{A}$) verknüpft wird, ergibt die 1.

Vier Variablen: Bei vier Variablen wird die Tafel bereits recht kompliziert (sie hat $2^4 = 16$ Zellen). Durch das unveränderte Prinzip ist sie aber leicht zu interpretieren:

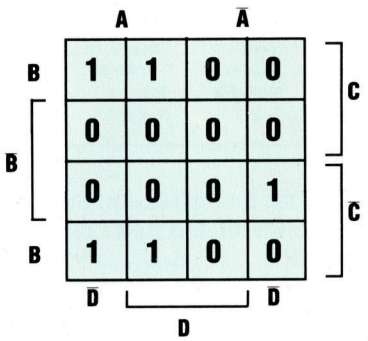


Hier sehen Sie eine k-Tafel mit den dazugehörigen Vereinfachungen:



$$\begin{aligned}
&AB\bar{C}\bar{D} + ABCD + A\bar{B}C\bar{D} + A\bar{B}CD \\
&= ABC(\bar{D} + D) + A\bar{B}C(\bar{D} + D) \\
&= ABC + A\bar{B}C \\
&= AC(B + \bar{B}) \\
&= AC
\end{aligned}$$

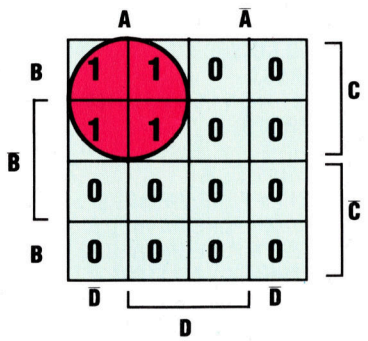
Ein weiteres Beispiel:

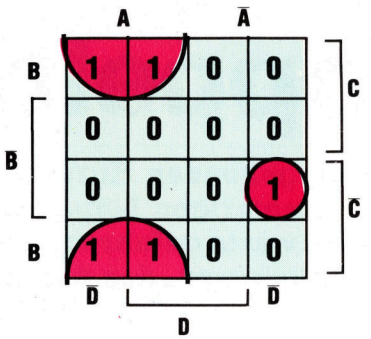


$$\begin{aligned}
&AB\bar{C}\bar{D} + ABCD + \bar{A}\bar{B}C\bar{D} + AB\bar{C}\bar{D} + AB\bar{C}D \\
&= ABC(\bar{D} + D) + \bar{A}\bar{B}C\bar{D} + AB\bar{C}(\bar{D} + D) \\
&= ABC + \bar{A}\bar{B}C\bar{D} + AB\bar{C} \\
&= AB(C + \bar{C}) + \bar{A}\bar{B}C\bar{D} \\
&= AB + \bar{A}\bar{B}C\bar{D}
\end{aligned}$$

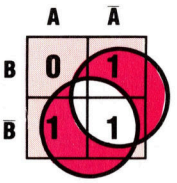
Stellung der Einsen ist wichtig

Wenn Sie die Stellung der Einsen in einer Tafel betrachten, werden Sie ein Muster erkennen. So haben im ersten Beispiel alle Ausdrücke mit AC eine Eins in der entsprechenden Zelle, im zweiten Beispiel alle Ausdrücke mit AB. Das führt zu einem Verfahren, mit dem sich Boolesche Ausdrücke leichter vereinfachen lassen. Hier ein Beispiel:





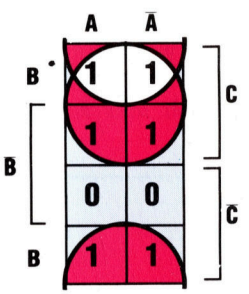
Mit ein wenig Übung können Sie Gruppen von Einsen herausgreifen und damit einfachere Ausdrücke formulieren. Hier etwa $\bar{A}B + A\bar{B} + \bar{A}\bar{B}$.



Aus einer k-Tafel mit zwei Variablen können zwei Gruppen von Einsen herausgelöst werden. Die eine Gruppe umfaßt alle NOT(B)-Fälle, die andere alle NOT(A)-Fälle, der Ausdruck läßt sich also zu $\bar{A} + \bar{B}$ vereinfachen. Mit dem Gesetz von de Morgan wird daraus $\overline{A.B}$. Ist dieses Ergebnis durch eine k-Tafel schneller zu erzielen?

Hier noch ein etwas schwierigeres Beispiel mit drei Variablen:

$ABC + A\bar{B}C + \bar{A}BC + AB\bar{C} \cdot \bar{A}B\bar{C} + \bar{A}\bar{B}C$



Die Gruppe der vier Einsen oben in der Tafel stellt alle Möglichkeiten dar, in denen C wahr ist. Die unteren und oberen Zeilen stehen für die Fälle, in denen B wahr ist. Der vereinfachte Ausdruck lautet also: B + C.

Im nächsten Kursabschnitt werden wir diesen Einstieg in die Anwendung von Karnaugh-Tafeln weiter vertiefen und dabei Ausdrücke mit vier Variablen untersuchen. Auch die Aufgabe der k-Tafeln beim Schaltungsentwurf soll gezeigt werden – dazu sind alle in diesem Kurs behandelten Theorien nötig.

Übung 4
Zeichnen Sie eine k-Tafel für drei Variablen, um die folgenden Ausdrücke zu vereinfachen:

a) $\bar{A}.B.C + \bar{A}.\bar{B}.C + \bar{B}.\bar{C} + \bar{A}.B.\bar{C}$
b) $A.\bar{B}.\bar{C} + \bar{A}.B.\bar{C} + A.B.\bar{C}$

Doppelte Kraft

Im vorhergehenden Kursabschnitt haben wir die Software zum Steuern eines einmotorigen Legoautos entworfen. Bisher konnte nur vorwärts oder rückwärts gefahren werden. Jetzt soll das Auto einen zweiten Motor bekommen, der das Kurvenfahren ermöglicht.

Durch den Einbau zweier gleichstarker Motoren in unser Auto kann durch das Zusammenspiel eines vor- und eines rückwärtslaufenden Antriebs in jede beliebige Richtung gefahren werden. Um ein zweimotoriges Fahrzeug zu drehen, lassen sich unterschiedliche Steuerungsprinzipien anwenden: So kann ein Motor gestoppt werden, während der andere weiterläuft – das Fahrzeug beschreibt einen Bogen um die stillstehenden Räder herum. Eine zweite Möglichkeit ist, die beiden Motoren in gegensätzlicher Richtung zu betreiben – dabei dreht das Auto auf der Stelle um die Hauptachse und wird so viel wendiger.

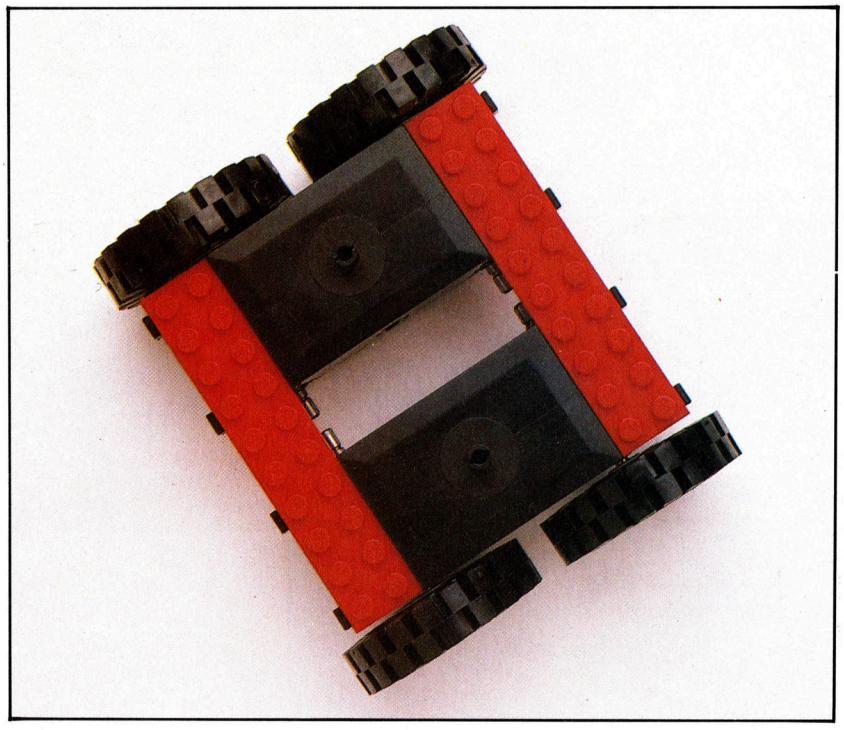
Ein solcher bidirektionaler Antrieb läßt sich über die vier roten Ausgangsbuchsen des Niedervolt-Gerätes steuern. Der rechte Motor wird dazu mit Anschluß 0 und 1 und der linke Motor mit Anschluß 2 und 3 verbunden.

Jeder Motor ist jetzt an zwei eigene Ausgangsbuchsen angeschlossen, wodurch er sich unabhängig vom anderen ansteuern läßt. Mit den richtigen Zahlen im Datenregister kann das Auto vorwärts, rückwärts, nach rechts oder nach links gelenkt werden. Liegt Anschluß 0 auf High und Anschluß 1 auf Low, dreht sich der rechte Motor vorwärts, im umgekehrten Fall läuft er in Gegenrichtung. Ähnlich verhält sich der linke Motor: Er läuft vorwärts, wenn Anschluß 2 High und Anschluß 3 Low ist. Durch Kombination der Drehrichtungen besitzt das Auto jetzt seine volle Manövrierfähigkeit:

Bewegung d. Autos	Linker Motor	Rechter Motor	Bit-Muster	Zahl im Datenregister
Stand	Aus	Aus	0000	0
Vorwärts	Vorwärts	Vorwärts	0101	5
Rückwärts	Rückwärts	Rückwärts	1010	10
Drehen links	Vorwärts	Rückwärts	0110	6
Drehen rechts	Rückwärts	Vorwärts	1001	9
Kurve links	Vorwärts	Aus	0100	4
Kurve rechts	Aus	Vorwärts	0001	1

Mit diesem Programm können Sie das Auto direkt über die Rechnertastatur steuern. „T" bedeutet vorwärts, „B" rückwärts, „F" ist eine Links- und „H" eine Rechtskurve.

Mit einem zweiten Motor kann unser Legoauto mehr als nur vorwärts- und rückwärtsfahren. Die Motoren werden unabhängig voneinander angesteuert – das Auto gewinnt dadurch volle Bewegungsfreiheit und läßt sich in jede beliebige Richtung lenken.

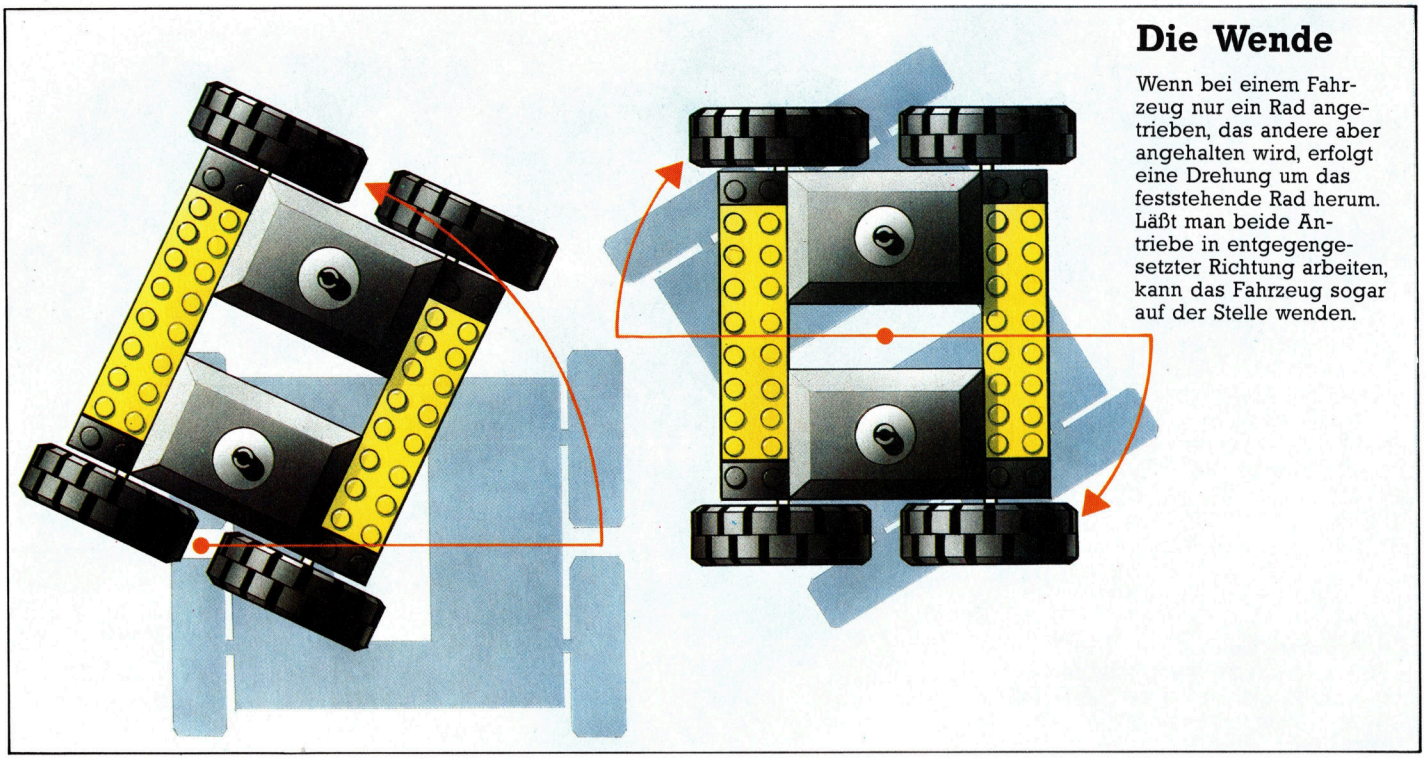


Acorn B

```
 10 REM BBC TWIN MOTORS
 20 DDR=&FE62:DATREG=&FE60
 30 ?DDR=255
 40 REPEAT
 50 A$=INKEY$(10)
 60 PROCtest_keyboard
 70 UNTIL A$="X"
 80 ?DATREG=0
 90 END
1000 DEF PROCtest_keyboard
1010 IF A$=" " THEN ?DATREG=0
1020 IF INKEY(-36) = -1 THEN ?DATREG=5
1030 IF INKEY(-101) = -1 THEN ?DATREG=10
1040 IF INKEY (-68) = -1 THEN ?DATREG=6
1050 IF INKEY (-85) = -1 THEN ?DATREG=9
1060 ENDPROC
```

Commodore 64

```
 10 REM CBM 64 TWIN MOTORS
 20 DDR=56579:DATREG=56577
 25 POKE650,128: REM REPEAT KEY MODE
 30 POKE DDR,255
 40 GETA$
 50 GOSUB1000:GOTO70
 60 POKEDATREG,0
 70 IF A$<>"X" THEN FOR I=1TO100:NEXT:GOTO40
 80 POKE DATREG,0
 90 END
1000 REM TEST INPUT S/R
1005 IFA$=" " THEN POKE DATREG,0
1010 IFA$="T" THEN POKE DATREG,5
1020 IFA$="B" THEN POKE DATREG,10
1030 IFA$="F" THEN POKE DATREG,6
1040 IFA$="H" THEN POKE DATREG,9
1050 RETURN
```



Die Wende

Wenn bei einem Fahrzeug nur ein Rad angetrieben, das andere aber angehalten wird, erfolgt eine Drehung um das feststehende Rad herum. Läßt man beide Antriebe in entgegengesetzter Richtung arbeiten, kann das Fahrzeug sogar auf der Stelle wenden.

Wenn keine Taste gedrückt ist, bleibt das Auto stehen. Das Programm setzt beim Loslassen der Taste automatisch eine Null ins Datenregister – die Motoren halten an. Beide Programme werden mit „X" gestartet.

Die Acorn B-Version des Programms ermöglicht durch die TEST-KEYBOARD-Funktion eine schnellere Tastaturabfrage als das Lesen des Tastaturbuffers mit INKEY. Dadurch wird die Steuerung etwas bequemer. Im Commodore-Programm wird die Auto-Repeat-Funktion eingeschaltet, so daß bei einem anhaltenden Tastendruck ständig Zeichen zum Tastaturbuffer geschickt und dort mit GET wieder gelesen werden. Leider hat der Commodore 64 keine Möglichkeit zur direkten Tastaturabfrage, was die exakte Steuerung des Autos erschwert. Eine Verbesserung läßt sich aber durch Löschen des Tastaturbuffers vor dem Lesen erreichen – fügen Sie dazu diese Programmzeile ein:

35 GET J$:IF J$<>"" THEN35

Außerdem muß noch das GOTO am Ende von Zeile 70 gegen GOTO35 ausgetauscht werden.

Die Geschwindigkeit, mit der das Fahrzeug auf einen Tastendruck reagiert, kann bei beiden Programmversionen Probleme verursachen: Falls die Schleife im Hauptprogramm schneller ausgeführt wird als die Auto-Repeat-Funktion, kann die Keyboard-Abfrage ergeben, daß keine Taste gedrückt ist. Dann schaltet der Motor schnell zwischen der gewählten Richtung und Aus hin und her. Dieses Problem wird in beiden Programmen durch eine künstliche Verzögerung beim Durchlaufen des Hauptprogramms umgangen. Beim Acorn B sorgt INKEY$(10) dafür, daß der Rechner eine Zehntelsekunde auf eine neue Eingabe wartet, bevor er das Programm fortsetzt. Das Commodore-Programm enthält eine Verzögerungsschleife in Zeile 70. Die richtige Verzögerungszeit haben wir durch Ausprobieren gefunden, sie hängt vom Zeitbedarf des einmaligen Programmdurchlaufs ab. Wenn Sie ein eigenes Programm schreiben, brauchen Sie Verzögerungszeiten nur dann vorzusehen, wenn der Programmdurchlauf schneller ist als die Auto-Repeat-Funktion der Tastatur.

Die nächste Aufgabe ist, das Auto zu programmieren, das heißt, bestimmte Bewegungsabläufe sollen eingegeben und automatisch wiederholt werden. Das ist durch Speicherung in einem zweidimensionalen Feld möglich, das sämtliche Fahrzeiten und Richtungswechsel enthält. Der erste Teil des Programms entspricht natürlich dem zuvor gezeigten Steuerprogramm, der zweite Teil dient zum Abruf der gespeicherten Daten. Diese werden im Feld DR() abgelegt. DR(C,1) speichert die Richtung, DR(C,2) die Dauer aller Manöver. Bei jedem Richtungswechsel wird ein neues Feldelement eingesetzt, da sich der Inhalt des Datenregisters ändert. Die Variable C dient dabei als „Zählwerk".

Acorn B

```
1000 REM BBC MOVEMENT MEMORY
1010 DDR=&FE62:DATREG=&FE60
1020 DIM DR(100,2)
1030 ?DDR=255:C=1:REM INIT COUNT
1040 REPEAT
1050 A$=INKEY$(10)
1060 PROCtest_keyboard
1070 UNTILA$="X"
1080 ?DATREG=0
1090 DR(C-1,2)=TIME
1100 REPEAT A$=GET$
1110 UNTIL A$="C"
```

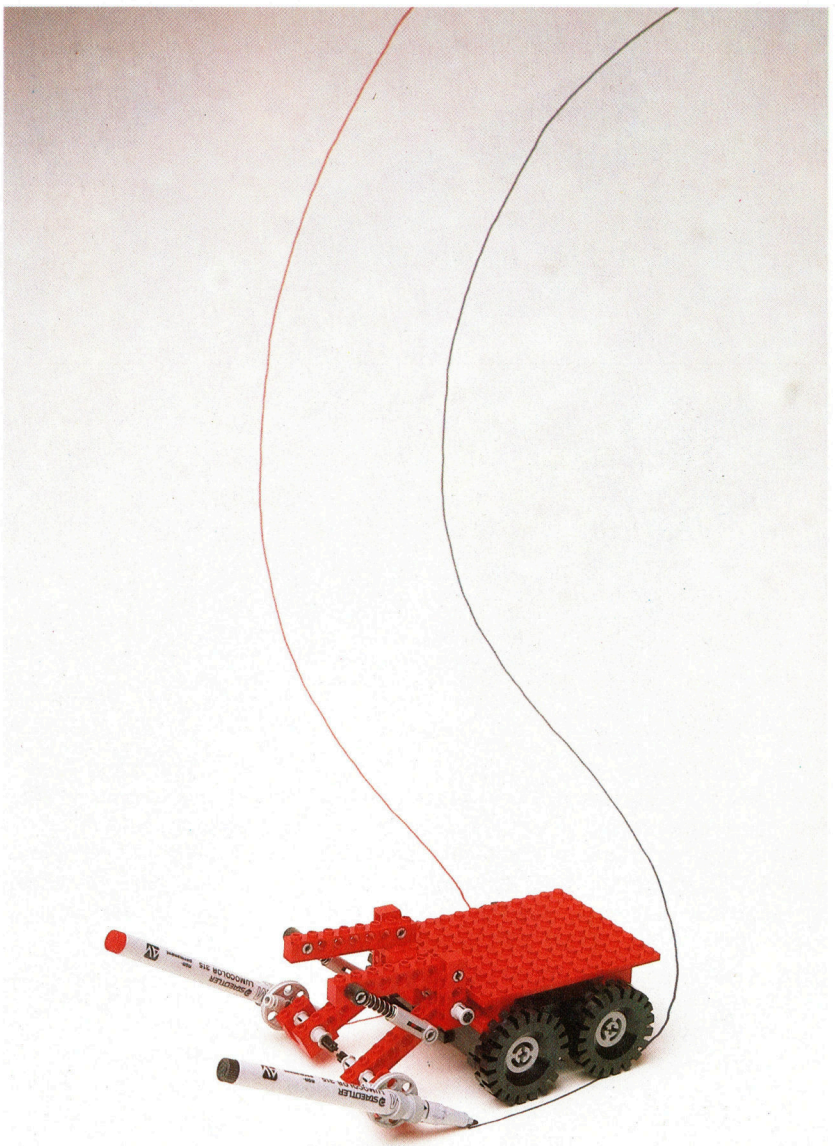
```
1120 REM REPLAY DATA
1130 FOR I=1TOC
1140 ?DATREG=DR(I,1)
1150 TIME=0
1160 REPEAT UNTIL TIME>=DR(I,2)
1170 NEXT I
1180 END
1190 :
1200 DEF PROCtest_keyboard
1210 IFA$=" " THEN ?DATREG=0
1220 IF INKEY(-36) = -1 THEN ?DATREG=5
1230 IF INKEY(-101) = -1 THEN ?DATREG=10
1240 IF INKEY(-68) = -1 THEN ?DATREG=6
1250 IF INKEY(-85) = -1 THEN ?DATREG=9
1260 PT=?DATREG
1270 IF PT<>DR(C-1,1) THEN PROCadd_data
1280 ENDPROC
1290 :
1300 DEF PROCadd_data
1310 DR(C-1,2)=TIME: REM STORE LAST TIME
1320 TIME=0: REM START NEW TIME
1330 DR(C,1)=PT: REM STORE PORT STATUS
1340 C=C+1: REM INCREMENT COUNT
1350 ENDPROC
```

Commodore 64

```
  10 REM CBM 64 MOVEMENT MEMORY
  15 DIMDR(100,2): REM DIRECTION ARRAY
  20 DDR=56579:DATREG=56577
  25 POKE650,128 : REM SET KEY REPEAT MODE
  30 POKEDDR,255: REM ALL OUTPUT
  35 C=1: REM INITIALISE COUNT
  40 GETA$
  50 GOSUB1000: REM TEST INPUT
  70 IF A$<>"X" THEN FOR I=1TO200:NEXT:GOTO40
  80 POKE DATREG,0: REM OFF
  85 DR(C-1,2)=TI-T: REM ENTER LAST TIME
  90 STOP: REM TYPE 'CONT' TO CONTINUE
  95 REM REPLAY DATA
 100 FOR I=1TOC
 110 POKEDATREG,DR(I,1)
 120 T=TI
 130 IF (TI-T)<DR(I,2)THEN130
 140 NEXT
 150 END
 999 :
1000 REM TEST INPUT S/R
1005 IFA$=" " THEN POKEDATREG,0
1010 IFA$="T" THEN POKEDATREG,5
1020 IFA$="B" THEN POKEDATREG,10
1030 IFA$="F" THEN POKEDATREG,6
1040 IFA$="H" THEN POKEDATREG,9
1045 PT=PEEK(DATREG)
1050 IFPT<>DR(C-1,1)THENGOSUB1500
1498 RETURN
1499 :
1500 REM ADD DATA TO ARRAY
1510 DR(C-1,2)=TI-T: REM ADD LAST TIME
1520 T=TI: REM TAKE NEW TIME
1530 DR(C,1)=PT: REM ENTER CURRENT PORT CONTENTS
1540 C=C+1: REM INCREMENT COUNT
1999 RETURN
```

Weil jede Bewegung durch diese Programme in Form von Richtung und Zeit gespeichert wird, wiederholen sich Fehler bei der Zeitfestlegung einzelner Bewegungen auch im nächsten Durchlauf. Damit nähern wir uns bereits den Problemen einer Echtzeit-Computersteuerung, bei der die Programm-Struktur und auch die Dauer der Ausführung eines Befehls zu wichtigen Faktoren werden.

Im nächsten Kursabschnitt wird die Steuerung per Joystick erläutert.



Übungen

Die volle Manövrierfähigkeit unseres Fahrzeuges macht eine Vielzahl interessanter Programmierübungen möglich. Lassen Sie sich von dieser kleinen Auswahl anregen:

1) Kalibrieren Sie Ihr Auto. Wie lange muß die entsprechende Zahl im Datenregister stehen, damit das Fahrzeug einen Meter vor- oder zurückfährt? Wie lange für eine Kurve von 90 Grad?

2) Entwerfen Sie eine kurvenreiche Fahrstrecke, die das Auto unter Tastatursteuerung „erlernen" soll. Schreiben Sie ein Programm, mit dem das Auto nach einmaligem Durchfahren programmgesteuert wieder zum Ausgangspunkt zurückfindet. (Verwenden Sie dazu Teile der abgedruckten BASIC-Programme.)

3) Verbinden Sie vier Schalter so mit dem Ausgangsbuffer, daß Sie Ihr Legoauto damit über den User Port steuern können.

Es ist verhältnismäßig einfach, ein Tastatur-Steuerprogramm zur Fahrzeuglenkung zu entwerfen. Und es ist auch nicht viel schwieriger, dieses Programm so auszubauen, daß die Eingaben des „Fahrers" gespeichert werden und das Fahrzeug einen bekannten Weg wiederholt, jedenfalls in der Theorie. Ein Vergleich der Original-Route mit dem vom Fahrzeug zurückgelegten Weg macht aber deutlich, wie schwer sich ein Programm auf die Wirklichkeit abstimmen läßt: Der Computer arbeitet exakt nach dem vereinfachten Modell einer „perfekten" Außenwelt, in der Abweichungen nicht berücksichtigt werden.

Software-Symphonie

Wir untersuchen die Programmpakete „1-2-3“ und „Symphony“ der Firma Lotus und „Xchange“ von Psion. Diese drei Systeme sind zwar auf große kommerzielle Geräte ausgerichtet, doch auch für preisgünstigere Maschinen gibt es bereits ähnliche Programme.

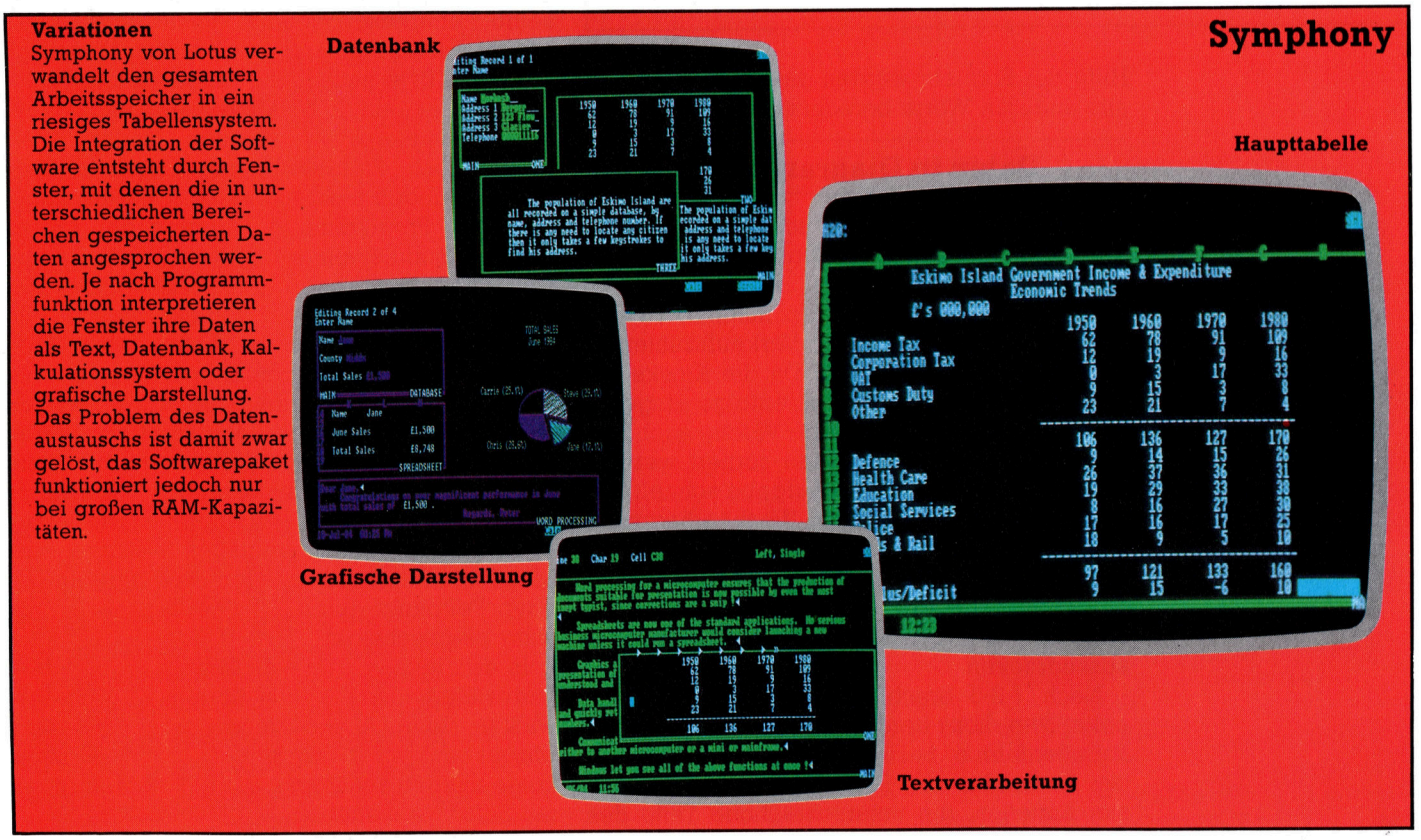
Integrierte Software sollte dem Anwender die Möglichkeit geben, alle Funktionen der verschiedenen Programmteile jederzeit einsetzen zu können. Weiterhin sollte eine einheitliche Befehlsstruktur existieren, und die Daten der einzelnen Anwendungen sollten sich frei austauschen lassen. Es gibt verschiedene Wege, diese Ziele zu erreichen.

Lotus 1-2-3 besitzt das vertraute Format eines Kalkulationssystems, dessen tabellenartig angeordnete Felder Zahlen und Formeln aufnehmen können, die automatisch berechnet werden. Lotus 1-2-3 hat vielseitige Fähigkeiten, die über finanzielle Prognosen und Analysen hinausgehen. So können die Felder außer numerischen Daten auch Namen und Adressen enthalten. Die Daten – beispielsweise eine Kundenliste mit Kontonummern – werden dabei in einem bestimmten Bereich des Rasters gespeichert. Da 1-2-3 Funktionen enthält, mit denen sich diese Daten durchsuchen und reorganisieren lassen, eignet es sich auch als Datenbank. Numerische Daten können in Diagrammform dargestellt werden, so daß auf ein zusätzliches separates Grafikprogramm verzichtet werden kann. Schließlich besitzt 1-2-3 auch Möglichkeiten der Textverarbeitung, die sich für einfache Schriftstücke problemlos einsetzen lassen. Wegen der geringen Speicherkapazität eignet es sich jedoch nicht als echtes Textsystem.

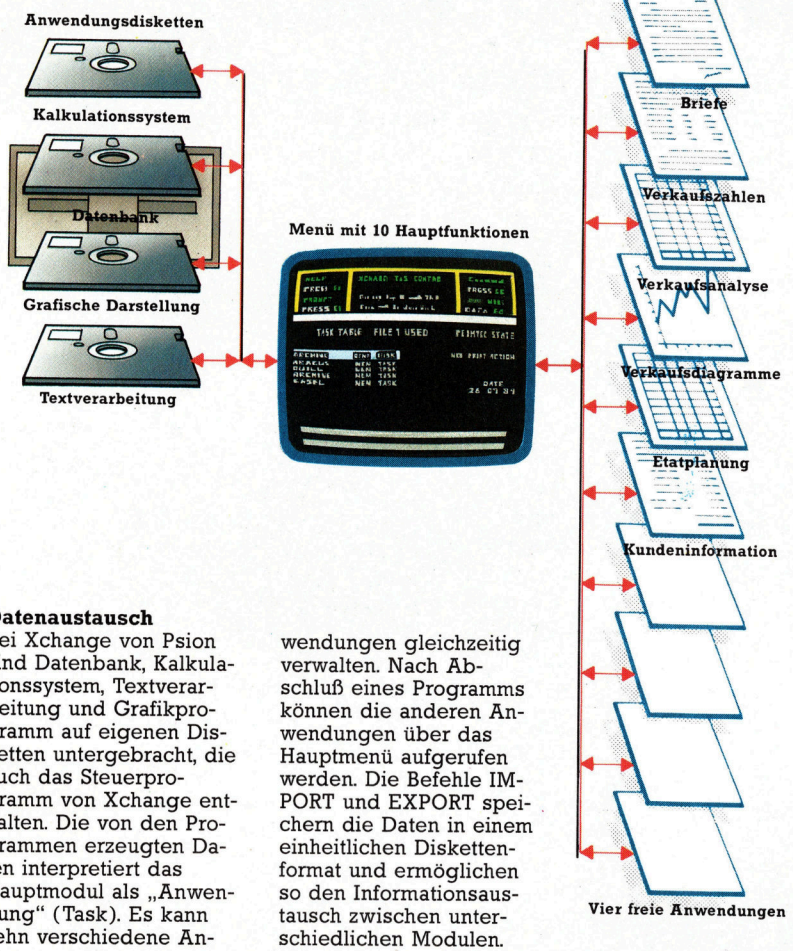
Verkaufsanalyse

Herkömmliche Programme könnten kaum ähnliche Ergebnisse wie 1-2-3 erzielen, da bei unterschiedlichen Anwendungen die Daten nicht in einem einheitlichen Tabellensystem enthalten sind. Nehmen Sie an, ein Anwender von 1-2-3 betreibt in mehreren Bezirken einer Stadt Zeitungskioske, und er benötigt deren wöchentliche, monatliche, vierteljährliche und jährliche Verkaufszahlen. Mit dem Tabellensystem läßt sich dies leicht bewerkstelligen. Der Besitzer trägt nur die wöchentlichen Verkaufszahlen der einzelnen Kioske ein, und mit

Variationen
Symphony von Lotus verwandelt den gesamten Arbeitsspeicher in ein riesiges Tabellensystem. Die Integration der Software entsteht durch Fenster, mit denen die in unterschiedlichen Bereichen gespeicherten Daten angesprochen werden. Je nach Programmfunktion interpretieren die Fenster ihre Daten als Text, Datenbank, Kalkulationssystem oder grafische Darstellung. Das Problem des Datenaustauschs ist damit zwar gelöst, das Softwarepaket funktioniert jedoch nur bei großen RAM-Kapazitäten.



Xchange



Datenaustausch
Bei Xchange von Psion sind Datenbank, Kalkulationssystem, Textverarbeitung und Grafikprogramm auf eigenen Disketten untergebracht, die auch das Steuerprogramm von Xchange enthalten. Die von den Programmen erzeugten Daten interpretiert das Hauptmodul als „Anwendung" (Task). Es kann zehn verschiedene Anwendungen gleichzeitig verwalten. Nach Abschluß eines Programms können die anderen Anwendungen über das Hauptmenü aufgerufen werden. Die Befehle IMPORT und EXPORT speichern die Daten in einem einheitlichen Diskettenformat und ermöglichen so den Informationsaustausch zwischen unterschiedlichen Modulen.

Hilfe der Formeln werden daraus automatisch alle anderen Werte berechnet.

Daten in Diagrammform

Bis zu diesem Punkt unterscheidet sich 1-2-3 nicht von anderen Kalkulationssystemen. Was passiert jedoch, wenn der Anwender seine Verkaufspositionen sortieren möchte, so daß der Stand mit den höchsten Verkaufszahlen an der Spitze der Liste steht? Da die Kioske anfangs in alphabetischer Reihenfolge eingegeben wurden, müßten sie jede Woche nach Eintrag der Verkaufszahlen neu sortiert werden. Lotus 1-2-3 erledigt diese Aufgabe schnell und einfach. Soll außerdem ein Wochendiagramm der Umsatzzahlen erstellt werden, genügen einige kurze Befehle, um diese Informationen aus dem Tabellen-Datenbanksystem herauszuziehen, in Diagrammform darzustellen und auszudrucken.

Nach 1-2-3 brachte Lotus das Programmpaket Symphony heraus, das auf dem gleichen Tabellenformat basiert. Mit Symphony kann der Anwender den Bildschirm jedoch in Fenster aufteilen, die unterschiedliche Teile des Tabellensystems anzeigen.

Besteht die Information beispielsweise aus Text, dann nimmt das Fenster die Form eines Textverarbeitungssystems mit Rändern und Tabulatormarken an. Bei einer grafischen Darstellung werden die Maßstäbe und Beschriftungen des Diagramms gezeigt, während in der Datenbank jede Information einen eigenen Rahmen enthält und der Eindruck einer Index-Kartei entsteht. Obwohl Symphony eigentlich nur ein erweitertes System für Tabellenkalkulation ist, verarbeitet es die vier Anwendungen gleichzeitig.

Wie 1-2-3 kann auch Symphony bestimmte Tastenfolgen „lernen", so daß der Anwender oft ausgeführte Abläufe automatisieren kann. Die Programmodule, die diese Tastenfolgen aufrufen, werden „Tastatur-Makros" genannt. Die Programme werden dabei wie alle anderen Daten in der Tabelle gespeichert, können aber Funktionen aufrufen. Symphony enthält vorprogrammierte Befehlsabläufe für das Zeichnen von Diagrammen und für das Suchen und Organisieren von Daten.

Großer Speicherbedarf

Symphony ist jedoch nur eins von mehreren ähnlichen Systemen. Eine große Konkurrenz ist „Framework" von Ashton Tate. Es enthält eine ähnliche Vielfalt von Funktionen, jedoch mit weniger offensichtlichen Datenstrukturen. Symphony und Framework sind etwa gleich teuer und benötigen einen großen Arbeitsspeicher. Symphony arbeitet zwar mit 320 KByte RAM, benötigt aber 512 KByte, um alle seine Fähigkeiten optimal einsetzen zu können, während Framework mit einem Minimum von 256 KByte auskommt. Beide Pakete funktionieren nur auf 16-Bit-Microcomputern.

Interessanterweise arbeiten weder Symphony noch Framework mit Overlays, das heißt, sie laden nicht – wie viele andere kommerzielle Programme – während der Arbeit Programmodule von der Diskette nach. Da die Chips und somit auch die Arbeitsspeicher der Computer immer billiger werden, liegt es für die Software-Architekten nahe, ihre Programme auf große Speicherkapazitäten auszurichten. Tatsächlich haben die Anwender jedoch diese Kapazitäten noch nicht zur Verfügung, und es wird einige Zeit dauern, bis speicherintensive Software dieser Art zur Selbstverständlichkeit wird. Programme wie Symphony setzen zwar einen neuen Standard, ihre Möglichkeiten werden jedoch zum gegenwärtigen Zeitpunkt durch die Grenzen der Hardware eingeschränkt.

In den letzten zwanzig Jahren wurden alternative Methoden der integrierten Software entwickelt, deren Anwendungen bereits auf dem Markt vorgestellt wurden. In der nächsten Folge werden wir auf die Zukunft der integrierten Software eingehen.

Kompakt verpackt

IBM-kompatible Maschinen dominieren am Markt der kommerziellen Microcomputer. Normalerweise sind sie billiger als das Original und haben sogar mehr Fähigkeiten. In diesem Artikel untersuchen wir ein tragbares Gerät mit IBM-Kompatibilität: den Compaq Plus.

Als IBM seinen Personal Computer in den Vereinigten Staaten vorstellte, erlangten die Microcomputer allgemein damit ein gutes Ansehen. Die Softwarehäuser konzentrierten sich auf den IBM, und in kurzer Zeit entstand eine große Zahl Programme für den PC. Viele Hardwarehersteller wollten auch von dieser breiten Softwarebasis profitieren und boten nur wenig später Computer an, auf denen die IBM-Software lief.

Einer der ersten war die Compaq Computer Corporation. Dieses Unternehmen entwickelte eine tragbare IBM-kompatible Maschine: den Compaq Plus mit 256 KByte und Doppellaufwerk. Es gibt aber auch Versionen mit nur einem Laufwerk oder einer Festplatte von zehn MByte. Der Arbeitsspeicher läßt sich zudem auf 640 KByte ausbauen.

Die Eigenschaft „tragbar" ist vielleicht etwas übertrieben für eine Maschine, die 13 kg wiegt und mit Sicherheit nur über kurze Strecken transportiert werden wird. Compaq hat das Gewichtsproblem jedoch erkannt und einen gepolsterten Griff eingebaut, der zumindest die Finger beim Tragen etwas schont.

Wie bei tragbaren Computern üblich, läßt sich die Tastatur über die Vorderseite des Computers klappen und einrasten. Da der Tragegriff auf der Rückseite der Maschine angebracht ist, dient die Tastatur beim Abstellen auch als Geräteboden. Der Compaq ist stabil gebaut, und die Tastatur scheint das Gewicht und auch Stöße gut zu vertragen. Die vollständige Einheit hat etwa die Ausmaße eines Nähmaschinenkoffers. Ein dickes Spiralkabel verbindet die Tastatur mit dem Gerät. Der Hersteller gibt zwar an, daß die Tastatur für ein komfortables Arbeiten frei beweglich sei, da das Spiralkabel aber schon bei einer Dehnung auf etwa 15 cm die Tastatur zurückzieht, ist der Bewegungsfreiraum doch begrenzt. Ein dünneres Kabel würde dieses Problem beseitigen.

Der Computer und die Tastatur sind mit versenkbaren Füßen ausgestattet, mit denen sich ein bequemer Arbeitswinkel einstellen läßt. Die Tasten des Compaq Plus sind leichtgängig und zuverlässig. Die zehn Funktionstasten links des alphanumerischen Feldes und der Zehnerblock auf der rechten Seite sind mit dem Aufbau der IBM-Tastatur identisch. Von Nachteil ist jedoch, daß die Unzulänglichkeiten der IBM-Tastatur ebenfalls übernommen wurden. So hat die „Enter"-Taste des IBM das gleiche Format wie die übrigen Tasten und ist leicht zu verfehlen. Beim Blindschreiben fällt unangenehm auf, daß die Position der „Shift"-Taste nicht unterhalb des „A" liegt. Dort befindet sich der umgekehrte Schrägstrich. Es braucht einige Zeit, bis man statt des Schrägstriches automatisch die richtige Taste links davon bedient.

Tastatur und Grundgehäuse zeichnen sich durch zweckmäßiges Design aus. Auf beiden Seiten des Gerätes ermöglichen Klappen den leichten Zugang zu Schnittstellen und zur Stromversorgung. Auch an die Unterbringung des Netzkabels und Steckers wurde gedacht. – Ein Punkt, der bei vielen „tragbaren" Geräten nicht berücksichtigt wird.

Belegung der Funktionstasten

Beim Aufruf von MS-DOS werden die Funktionstasten links des Tastenfeldes als Editierhilfen eingesetzt. Die Funktionen ändern sich je nach geladenem Anwendungsprogramm. Bei BASIC sind diese Tasten mit Schlüsselwörtern wie LOAD, SAVE und LIST belegt. Bei MSX-Computern wird bei der Belegung der Funktionstasten ein ähnliches Verfahren angewandt.

Der Zahlenblock rechts der Tastatur erfüllt eine Doppelfunktion. Normalerweise kann damit der Cursor über den Bildschirm bewegt werden, beim Drücken der „Num Lock"-Taste wird er jedoch zu einem Taschenrechner.

Der hochauflösende Monochrom-Monitor hat die Ausmaße 17,7 cm × 13,3 cm und zeigt 25 Zeilen mit je 80 Zeichen an. Die Zeichenmatrix setzt sich aus neun mal vierzehn Punkten zusammen, und die Textauflösung beträgt 720 x 350 Punkte. Im Grafikmodus beträgt die Bildschirmauflösung 640 x 200 Pixel. Texte und Grafiken lassen sich problemlos zusammen darstellen. Die Bildschirmdarstellung ist gut lesbar, wobei sich die Helligkeit mit einem Regler links des Bildschirms einstellen läßt. Auf der rechten Seite des Hauptgerätes befinden sich die beiden 5 1/4-Zoll-Diskettenstationen. Da der Compaq standardmäßig mit nur einem Laufwerk ausgerüstet ist, spricht die MS-DOS-Systemdiskette nur das Laufwerk A an. Besonders beim Kopieren ist dies lästig, da die Disketten ständig gewechselt werden müssen. Das MS-DOS läßt sich jedoch leicht umkonfigurieren. Zusätzlich lassen sich auf dem Compaq Plus CP/M 86, CCP/M, MP/M 86, UCSD-PASCAL und MBASIC 2.0 einsetzen.

Verbindungen zur Außenwelt

Unter einer Klappe auf der rechten Seite des Gerätes liegen die Schnittstellenbuchsen. Der Compaq Plus besitzt einen parallelen Druckeranschluß im Centronicsformat und je einen RGB-Anschluß und eine RF-Buchse. Über drei Erweiterungssteckleisten lassen sich IBM-kompatible Platinen anschließen. Zusätzliche Erweiterungen sind einige RAM-Karten, eine Farbkarte für den Bildschirm und ein Modem für die Datenübertragung. Unter einer weiteren Klappe auf der linken Seite liegen der Netzanschluß und der Hauptschalter. Hier befindet sich auch der Ventilator, der das Innere der Maschine kühlt. Ein Metallgehäuse (es ist die Ursache für das große Gewicht des Gerätes) schützt die Schaltungen im Inneren des Computers vor Interferenzen. Das Metallgehäuse dient außerdem als Hitzeableitung und schützt gegen Druck und Stoß bei unsanftem Transport.

Im Lieferumfang sind drei Handbücher enthalten – eine Betriebsanleitung und zwei Nachschlagewerke für MS-DOS und BASIC. Sie sind nicht wie üblich als Ringbücher angelegt, es handelt sich vielmehr um flexible Taschenbücher. Da nur die Betriebsanleitung eine Einführung für Systemneulinge enthält, ist man für das Erlernen von BASIC oder MS-DOS auf andere Lehrbücher angewiesen.

IBM-kompatibel

Wie alle älteren IBM-kompatiblen Geräte setzt auch der Compaq den Intel-8088 und nicht den neueren 8086 ein, mit dem modernere Geräten wie der Olivetti M25 bereits arbeiten. Obwohl der 8088 ein 16-Bit-Prozessor mit einem 20-Bit-Adreßbus ist, hat sein Datenbus nur das Acht-Bit-Format und nicht 16 Bit wie der des 8086. Der Compaq ist daher langsamer als seine 8086-Konkurrenten, arbeitet aber mit der gleichen Geschwindigkeit wie der IBM PC.

Bei der Frage der Kompatibilität mit IBM-Software erzielt der Compaq gute Resultate. Sogar ein komplex aufgebautes Programm wie „Lotus 1-2-3" läuft auf der Maschine. Die Diagnosediskette des IBM ist eines der wenigen Programme, die auf dem Compaq nicht funktionieren. Da es jedoch direkt auf das ROM mit dem BASIC-Ein- und Ausgabesystem (BIOS) zugreift, hat dies nur wenig Gewicht.

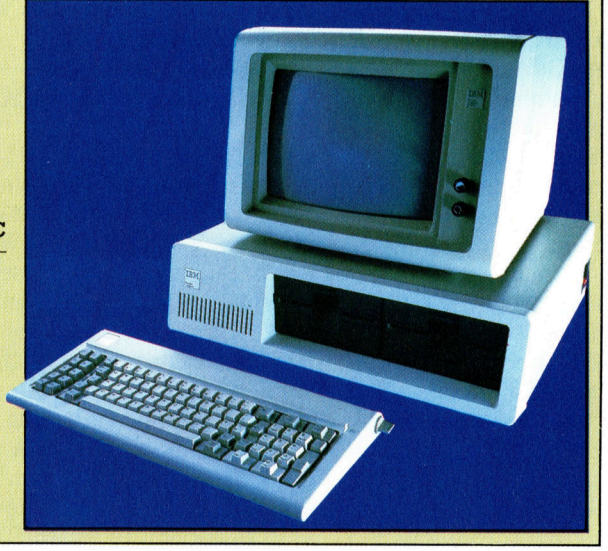
Der Compaq ist mit Abstand eine der zuverlässigsten und meistverkauften IBM-kompatiblen Maschinen auf dem Markt. Negativ fällt das Alter der Maschine ins Gewicht, und dies nicht nur wegen des überholten 8088-Chips. Es scheint, daß die Tage der „tragbaren" Geräte mit zehn und mehr Kilogramm gezählt sind. Je besser die echten Portables werden, desto eher wird auch IBM mit einem derartig kompakten Gerät auf den Markt kommen.

Viel fürs Geld

Hauptvorteil des Compaq Plus ist seine IBM-Kompatibilität und somit die umfangreiche Softwareauswahl. Die Eigenschaften des Compaq Plus und des IBM werden hier vergleichend dargestellt:

Eigenschaft	Compaq	IBM PC
Doppellaufwerk 360K	Standard	Extra
Hochauflösender Farbmonitor	*Standard	Extra
Karte für Monitorausgabe	Standard	Extra
Karte für Farbgrafik	Standard	Extra
128K-RAM	Standard	Extra
Diskettenbetriebssystem und BASIC	Standard	Extra
Tastatur	Standard	Extra

* Der Monitor des Compaq ist hochauflösend und monochrom, die Maschine besitzt jedoch standardmäßig einen RGB-Ausgang für einen Farbmonitor.



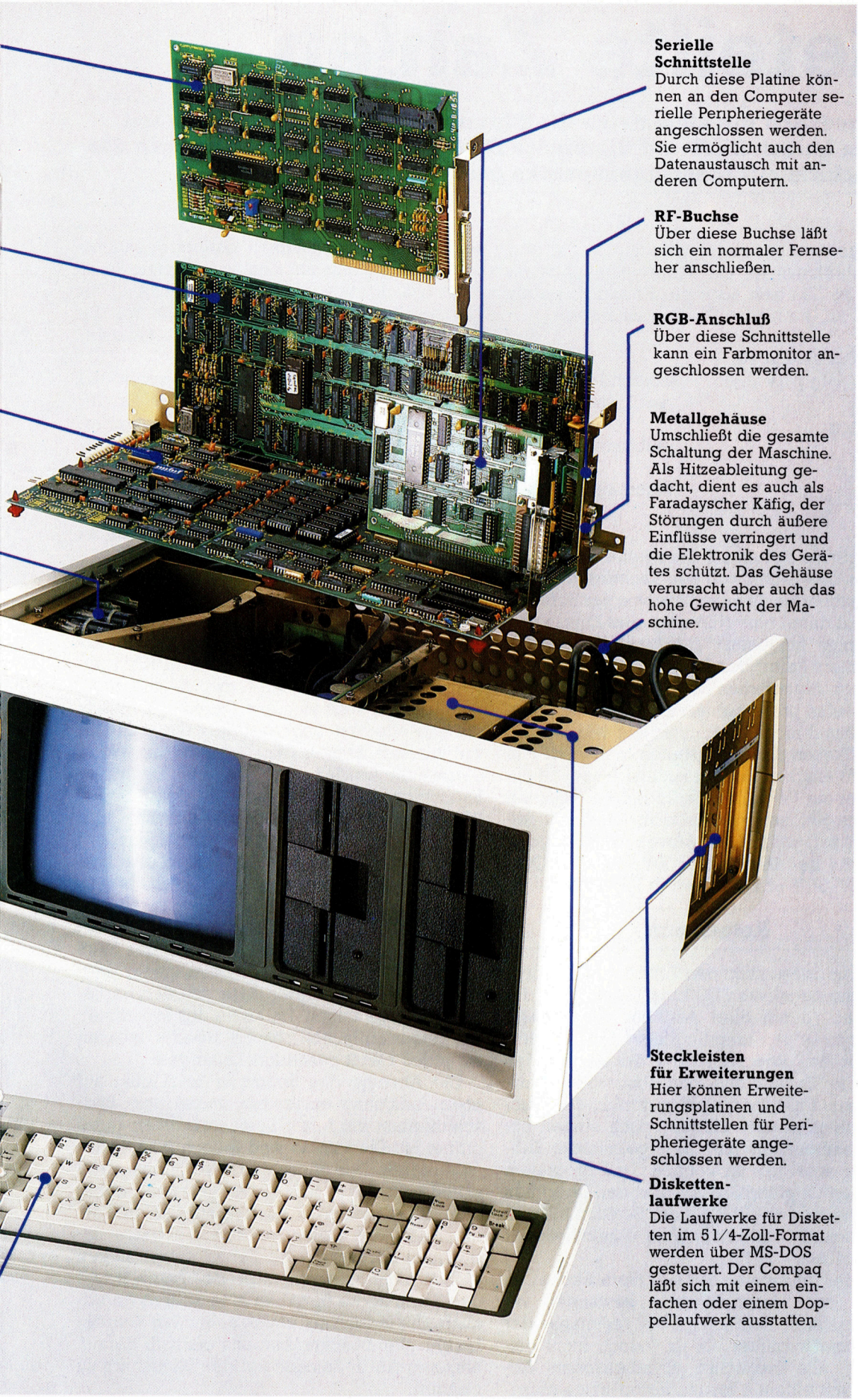
Druckerplatine
Über diese Platine mit paralleler Druckerschnittstelle läßt sich ein Drucker mit Centronics-Interface anschließen.

Monitorplatine
Diese Platine kann in eine der Erweiterungssteckleisten des Compaq eingesetzt werden. Sie enthält Schaltungen, mit denen sich ein Monitor oder Fernseher betreiben läßt.

Hauptplatine
Enthält außer 256K RAM, dem 8088-Prozessor und den Chips für Ein- und Ausgabe auch Steckleisten für zusätzliche Chips.

Stromversorgung
Das Netzteil sitzt neben dem Ventilator, der das Innere der Maschine kühlt.

Tastatur im IBM-Stil
Die Tastatur ist mit der des IBM PC identisch. Links liegen die Funktionstasten und rechts der Zehnerblock für Zahleneingaben.



Serielle Schnittstelle
Durch diese Platine können an den Computer serielle Peripheriegeräte angeschlossen werden. Sie ermöglicht auch den Datenaustausch mit anderen Computern.

RF-Buchse
Über diese Buchse läßt sich ein normaler Fernseher anschließen.

RGB-Anschluß
Über diese Schnittstelle kann ein Farbmonitor angeschlossen werden.

Metallgehäuse
Umschließt die gesamte Schaltung der Maschine. Als Hitzeableitung gedacht, dient es auch als Faradayscher Käfig, der Störungen durch äußere Einflüsse verringert und die Elektronik des Gerätes schützt. Das Gehäuse verursacht aber auch das hohe Gewicht der Maschine.

Steckleisten für Erweiterungen
Hier können Erweiterungsplatinen und Schnittstellen für Peripheriegeräte angeschlossen werden.

Diskettenlaufwerke
Die Laufwerke für Disketten im 5 1/4-Zoll-Format werden über MS-DOS gesteuert. Der Compaq läßt sich mit einem einfachen oder einem Doppellaufwerk ausstatten.

Compaq Plus

ABMESSUNGEN
480x400x200 mm

ZENTRALEINHEIT
Intel-8088, 4,7 MHz

SPEICHERKAPAZITÄT
256 K RAM, auf 640 KByte erweiterbar.

BILDSCHIRMDARSTELLUNG
Text: 25 Zeilen mit je 80 Zeichen; Grafik: 640x200 Pixel.

SCHNITTSTELLEN
Centronics parallel für einen Druckeranschluß; RGB- und RF-Buchse; Steckleisten für Erweiterungen, über die sich Zusatzplatinen anschließen lassen.

PROGRAMMIERSPRACHEN
BASIC, FORTRAN, PASCAL, COBOL.

TASTATUR
47 Schreibmaschinentasten, 14 Steuertasten, 10 Funktionstasten und 10 Tasten für die numerische Eingabe und Taschenrechnerfunktionen.

HANDBÜCHER
Drei Handbücher werden mitgeliefert: ein Betriebshandbuch und zwei Nachschlagewerke für MS-DOS und BASIC. Da nur das Betriebshandbuch eine schrittweise Einführung bietet, sollten Anfänger auf andere Bücher zurückgreifen.

STÄRKEN
Die Maschine ist stabil gebaut und mit dem IBM PC weitgehend kompatibel.

SCHWÄCHEN
Durch ihr Gewicht paßt die Maschine eher in die Klasse der „transportablen" als in die der „tragbaren" Geräte.

Geteilte Zahlen

Wir beenden die allgemeine Einführung in den Maschinencode mit einem kurzen Blick auf die binäre Division und sehen uns an, wie der Bildschirm mit Maschinensprache gesteuert werden kann.

Ebenso wie uns die Handmultiplikation den Algorithmus für die binäre Multiplikation liefert, so läßt sich auch die Handdivision als Modell für die binäre Division einsetzen:

```
00001110 Rest 00          Quotient
10011010:1011             Dividend mit Teiler
-1011                     subtrahiere Teiler
 10000
 -1011                    subtrahiere Teiler
   1011
  -1011                   subtrahiere Teiler
      00                  kein Rest
```

Die Methode beruht auf der wiederholten Subtraktion des Teilers von den hochwertigen Bits des Dividenden. Je nach Ergebnis der Subtraktion wird dabei eine Null oder eine Eins in den Quotienten eingesetzt. Die letzte Subtraktion des Teilers liefert den Rest.

Dieser Algorithmus läßt sich auf verschiedene Weise programmieren. Der Ablauf ist jedoch längst nicht so durchsichtig wie bei der Multiplikation. Dem Z80 stehen hier wiederum die Schnelligkeit und Flexibilität seiner 16-Bit-Register zur Verfügung, während der 6502 nur acht Bits gleichzeitig bearbeiten kann. Der Teiler (Divisor) wird in der Adresse DIVSR untergebracht, der Dividend in DVDND, der Quotient in QUOT und der Rest in RMNDR.

Rotationen

Wenn die Subtraktion des Teilers vom Teildividend ein negatives Ergebnis liefert, muß der Dividend (durch eine Addition des Teilers) wieder in seinen ursprünglichen Zustand versetzt werden. Beachten Sie in der Version des 6502 die Behandlung des Prozessor-Status-Registers: Das Übertragsflag muß in den Quotient rotiert werden, dabei jedoch seinen Zustand beibehalten, um das Ergebnis der Subtraktion anzeigen zu können. Das Programm stellt den ursprünglichen Status des Übertrags her, indem es das PSR vor der Rotation auf den Stack schiebt und unmittelbar danach wieder zurücklädt.

Mit der Umsetzung der vier Grundrechenarten in den Maschinencode gewannen wir einen tieferen Einblick in die Mechanik der Maschinenvorgänge. Es ist jedoch nicht notwendig, alle möglichen arithmetischen Vorgänge mit einem oder mehreren Bytes aufzuzeigen, da diese Routinen bereits in zahlreichen Lehrbüchern veröffentlicht wurden. Sollte es jedoch notwendig werden, die bekannten Routinen zu verändern, dann werden wir die Änderungen abdrucken oder als „Hausaufgabe" stellen.

Bis jetzt haben wir den Arbeitsspeicher und die CPU als Rechenmaschine behandelt, deren Ergebnisse wir uns mit einem Monitorprogramm im RAM angesehen haben. Selbstverständlich genügt diese Anzeigemethode nicht. Es hatte jedoch wenig Sinn, vor Abschluß der arithmetischen Mechanik und dem Aufruf von Unterroutinen auf die Anzeigemöglichkeiten des Maschinencodes einzugehen.

Numerierung der Pixelbytes

Die Anzeigen der meisten Microcomputer arbeiten mit Memory-Maps (bestimmte RAM-Bereiche enthalten Abbilder des Bildschirms). Die Anzeige setzt sich aus Pixeln zusammen, die entweder ein- oder ausgeschaltet sind und von binären Einsen (An) oder Nullen (Aus) dargestellt werden. Der gesamte Inhalt des Bildschirms findet sich daher in den Bits des Bildschirmspeichers wieder. Obwohl der Acorn B, der Spectrum und der Commodore diese Methode verwenden, setzen alle drei Geräte andere Versionen ein. Für uns würde es genügen, jede Bildschirmzeile in Pixelbytes aufzuteilen, die von links nach rechts fortlaufend numeriert wären. Dabei würde das links außen stehende Byte einer Zeile immer dem rechts außen stehenden Byte der darüberliegenden Zeile folgen. Da keine der Maschinen mit dieser einfachen Lösung arbeitet, müssen wir jedes Gerät gesondert behandeln.

Die Anzeige des Spectrum ist ständig auf hohe Auflösung eingestellt, wobei dem Bildschirminhalt ein fester Speicherbereich zugeordnet ist. Die Punkte sind jedoch kompliziert angeordnet: Der Bildschirm wird horizontal in drei Blöcke mit je acht Zeilen unterteilt, wobei jede Zeile wiederum acht Pixelzeilen enthält. Innerhalb der Zeilen werden die Bytes sequentiell adressiert, die Zeilen selbst stehen jedoch nicht hintereinander. Der Acorn B und der Commodore 64 haben eine andere Anordnung, die jedoch auf einem ähnlichen Konzept beruht. Wir beschränken uns deshalb darauf, einzelne ASCII-Zeichen auf den Bildschirm zu bringen.

Im ROM befinden sich einige Maschinencoderoutinen für die Steuerung der Anzeige. Mit einer genauen Beschreibung dieser Routinen lassen sie sich von der Assemblersprache aus aufrufen. Für diesen Vorgang muß man jedoch die Anfangsadresse, die eingesetzten Register sowie alle notwendigen Vorbedingungen genau kennen.

Auf dem Spectrum brauchen keine Vorbedingungen erfüllt zu werden. Als Register wird der Akkumulator verwandt, der den ASCII-Code des darzustellenden Zeichens enthalten muß. Bei der Ausgabe des Befehls RST $10 wird das Zeichen, dessen Code im Akkumulator gespeichert ist, an der augenblicklichen Cursorposition angezeigt. Auch die beiden anderen Systeme arbeiten nach dieser Methode. Den Op-code RST (ReSTart) gibt es jedoch nur im Befehlssatz des Z80. Er ist ein Ein-Byte-Befehl, der die Zero-Page anspricht und nur einen von acht möglichen Operanden annehmen kann – $00, $08, $10, $18 bis $38. Jede dieser Speicherstellen zeigt auf die Anfangsadresse einer ROM-Routine auf der Zero-Page, die die Ein- und Ausgabe steuert. Da RST schneller ist (RST läuft mit größerer Geschwindigkeit ab als CALL – allerdings würde nur die CPU den Unterschied merken) und Programme sich dadurch auf andere Geräte übertragen lassen, werden diese Routinen über den RST-Befehl aufgerufen und nicht durch ihre direkten Anfangsadressen. Wenn allen Z80-Programmierern bekannt ist, daß auf Z80-Maschinen der Befehl RST $10 die PRINT-Routine aufruft, dann braucht sich niemand Gedanken darüber zu machen, wo die Konstrukteure sie untergebracht haben. Die Konstrukteure ihrerseits können sie an jede beliebige Position setzen, wenn die RST-Speicherstellen auf die richtigen Anfangsadressen zeigen.

Auf dem Acorn B läuft der Vorgang ähnlich ab: Bei Aufruf des Befehls JSR $FFEE wird der im Akkumulator gespeicherte ASCII-Code an der augenblicklichen Cursorposition des Bildschirms angezeigt. Bei $FFEE beginnt die OSWRCH-Routine, die in der Dokumentation des Acorn B häufig erwähnt wird.

Übertragung der Daten

Auch der Commodore 64 folgt diesem Muster. Mit dem Befehl JSR $FFD2 wird der im Akkumulator gespeicherte ASCII-Code an der augenblicklichen Position des Cursors als Zeichen auf den Schirm gebracht. Die entsprechende CHKOUT-Routine ist im Handbuch beschrieben.

Die Abläufe zeigen, wie sich ROM-Routinen einsetzen lassen und wie Kommunikationsregister funktionieren. Die Datenübertragung zwischen dem aufrufenden Programm und der Unterroutine funktioniert in beiden Richtungen. So kann eine Eingaberoutine zum Beispiel die Zeichen eines Peripheriegerätes über den Akkumulator an die CPU übermitteln. Selbst wenn dabei keine gültigen Daten übertragen werden, so kann die Unterroutine über eines dieser Register zumindest eine Fehlermeldung ausgeben. Auf Tastatureingaben und die anderer Peripheriegeräte werden wir in späteren Folgen eingehen, ebenso wie auf das Plotten mit hoher Auflösung.

16-Bit- und 8-Bit-Division

Z80			6502		
START	LD	A,(DIVSR)	START	LDA	#$00
	LD	D,A		STA	QUOT
	LD	E,$00		STA	RMNDR
	LD	HL,(DVDND)		LDX	#$08
	LD	B,$08		LDA	DVDHI
LOOP0	AND	A		SEC	
	SBC	HL,DE		SBC	DIVSR
	INC	HL	LOOP0	PHP	
	JP	P,POSRES		ROL	QUOT
NEGRES	ADD	HL,DE		ASL	DVDLO
	DEC	HL		ROL	A
POSRES	ADD	HL,HL		PLP	
	DJNZ	LOOP0		BCC	CONT1
	LD	(QUOT),HL		SBC	DIVSR
	RET			JMP	CONT2
DIVSR	DB	$F9	CONT1	ADC	DIVSR
DVDND	DW	$FDE8	CONT2	DEX	
QUOT	DB	$00		BNE	LOOP0
RMNDR	DB	$00		BCS	EXIT
				ADC	DIVSR
				CLC	
			EXIT	ROL	QUOT
				STA	RMNDR
				RTS	
			DIVSR	DB	$F9
			DVDLO	DB	$E8
			DVDHI	DB	$FD
			QUOT	DB	$00
			RMNDR	DB	$00

Der Maschinensprachkurs begann mit einem generellen Überblick. Sie haben gesehen, wie sich die Bytes des RAM als Folge von ASCII-Zeichen interpretieren lassen oder als aneinandergereihte Zwei-Byte-Adressen oder auch als Befehlssequenz des Maschinencodes. Der Maschinencode-Monitor zeigte, daß sich die gleichen Bytes als drei völlig verschiedene, aber dennoch gültige Befehlssequenzen interpretieren lassen – abhängig davon, ob die Disassemblierung beim ersten, zweiten oder dritten Byte der Folge anfängt.

Danach wurde genauer auf Speicherorganisation und Adreßformate eingegangen. Schon kurze Zeit, nachdem Sie sich mit binärer Arithmetik beschäftigt hatten, konnte festgestellt werden, welchen Begrenzungen die CPU unterworfen ist: Acht-Bit-Prozessoren sind – bis auf einige Ausnahmen – an die Grenzen eines Bytes gebunden (anders ausgedrückt: an den Dezimalbereich von 0 bis 255). Mit der Einführung der binären Arithmetik wurde deutlich, wie unhandlich in diesem Umfeld das Dezimalsystem ist. Die Vorstellung der Spei-

cherseiten zeigte, wie die logische Größe der Seiten aus der Zahlenbasis entsteht, und machte deutlich, daß in einem binären System die Grundlage immer ein Vielfaches von Zwei sein muß. Sie haben in der Zahl 256 (zwei hoch acht) die magische Zahl des Systems der Acht-Bit-Prozessoren erkannt.

Da sich auch das Binärsystem sehr schnell als unhandlich und fehleranfällig erwies, gingen wir auf das Hexadezimalsystem (Zahlenbasis 16) und die dazugehörige Mathematik über. Sie sahen, wie ein Acht-Bit-Byte durch zwei Hexadezimalstellen (von $00 bis $FF) dargestellt werden kann, wobei eine Hexstelle jeweils die vier niederwertigen Bits anzeigt und die andere die vier höherwertigen Bits eines Bytes.

Zusammenfassung

Auf dieser Grundlage wurde untersucht, wie BASIC-Programme im RAM-Bereich gespeichert sind. Mit der Beschreibung der Token erhielten Sie Einblick in die Mechanik der Betriebssysteme, während die Zeilenendmarkierung deutlich machte, wie ein BASIC-Interpreter das Ende einer Zeile erkennen kann. Die Link-Adressierung des Commodore führte schließlich in die lo-hi-Adreßkonvention und in die indirekte Adressierung ein.

Von diesem Punkt an beschäftigten wir uns mit der Assemblersprache. Dabei wurde zunächst untersucht, wie die CPU mit Hilfe von Acht-Bit-Op-codes einfache Aufgaben ausführt. Von diesem Konzept war es nur ein kleiner Schritt zu den mnemotischen Kürzeln der Assemblersprache. Es wurde deutlich, daß die Programmierung im Maschinencode, in der Assemblersprache oder in BASIC nur unterschiedliche Methoden sind, Abläufe festzulegen. Dabei ist die logische Lösung eines Problems wesentlich wichtiger als die eigentliche Umsetzung in den Programmcode.

In den nächsten Folgen beschäftigten wir uns dann mit dem Laden und Starten des Maschinencodes mit Hilfe von BASIC-Programmen. Dabei sahen wir uns auf dem Acorn B, Spectrum und Commodore 64 die Systemvariablen und Pointer der Betriebssysteme genauer an und fanden heraus, wie man dem BASIC Speicherplatz „stehlen" kann.

Nach einem kurzen Blick auf kleine Computersysteme und auf die Zentraleinheiten Z80 und 6502 haben Sie Ihre ersten Assemblerprogramme geschrieben, die den Inhalt des Speichers und des Akkumulators veränderten. Mit den Assembleranweisungen oder Pseudo-Opcodes entfernten wir uns von den verwirrenden Einzelheiten des Maschinencodes und der Handassemblierung und kamen der komfortablen „realen" Assemblerprogrammierung einen großen Schritt näher.

An diesem Punkt gingen wir auf die logische Struktur der Programmiersprache und damit auf die Aufgabe des Prozessor-Status-Registers (PSR) ein. Seine Rolle als Anzeigeinstrument der CPU-Vorgänge wurde bei der Einführung binärer mathematischer Abläufe – und besonders bei dem Befehl „Addiere mit Übertrag" – deutlich. Von nun an drehte sich der Kurs hauptsächlich um das PSR und die damit eingesetzten Befehle.

Nachdem wir die verschiedenen Adressierarten gestreift hatten, lernten Sie die indizierte Adressierung ausführlicher kennen, da sie für Schleifen, Listen und Tabellen wichtig ist. Eine Steuerung des Programmflusses wurde notwendig. Daher untersuchten wir die Befehle der bedingten Verzweigung. Mit der Programmsteuerung, einfacher Arithmetik und matrixähnlichen Tabellen hatten wir nun fast das gesamte Grundgerüst einer Programmiersprache. Jetzt fehlten nur noch Übung und die systematische Untersuchung aller Einsatzmöglichkeiten.

Der Aufruf von Assembler-Unterroutinen und die Rücksprungtechnik führten uns in einen der letzten noch unerwähnten Bereiche des Betriebssystems: den Stack. Dieses Instrument zeigte Ihnen einige neue Kniffe der Maschinencodeprogrammierung. Dabei eröffnete eine genauere Untersuchung der CPU-Register und ihrer Anwendung weitere Möglichkeiten zur Steuerung von Speicher und Microprozessor.

Schließlich waren Sie mit dem Aufbau des Microprozessors vertraut und hatten auch genügend Op-code-Befehle zur Verfügung, um genauer auf die binäre Arithmetik einzugehen. Die Eigenheiten der Subtraktion und des Zweierkomplements und auch der Aufbau von Multiplikation und Division mit Verschieben und Rotation wurden untersucht.

In der Fortsetzung des Kurses werden wir uns mit der Programmierung im Maschinencode weiter vertraut machen und die beiden Prozessoren – Z80 und 6502 – für spezielle Aufgaben einsetzen. Dabei behandeln wir auch die 6809-CPU des Dragon 32 und 64.

Lösungen

1) Die schnellste Lösung ist sicherlich eine Routine, die speziell auf 16-Bit-Multiplikanden ausgerichtet ist – ähnlich wie die Acht-Bit-Routine der letzten Folge. Sie können aber auch eine 16-Bit-Multiplikation in zwei separate Acht-Bit-Multiplikationen aufteilen (zunächst mit dem niederwertigen Byte, dann mit dem höherwertigen Byte multiplizieren). Dabei wird die bestehende Acht-Bit-Routine zweimal aufgerufen, der Übertrag aus dem niederwertigen Byte berücksichtigt und das Resultat in den Ergebnisbytes gespeichert.

2) Eine Multiplikation durch fortgesetzte Addition läßt sich leicht über eine Schleife programmieren, deren Zähler als Multiplikator dient. Bei jedem Schleifendurchgang wird der Multiplikand auf das Ergebnis addiert.

ROR – Bit nach rechts rotieren

6502

Register – 6A (1 Byte)

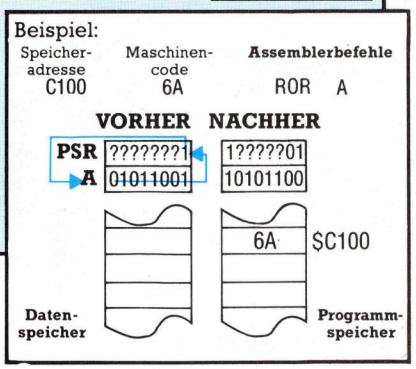
Der Inhalt des Bytes wird über den Übertrag um ein Bit nach rechts verschoben.

Bewirkt beim PSR:

	S V		B D	I	Z C
MSB	X				X X LSB

Beispiel:

Speicheradresse	Maschinencode	Assemblerbefehle
C100	6A	ROR A



RR – Bit nach rechts rotieren

Z80

Register – CB (2 Bytes)

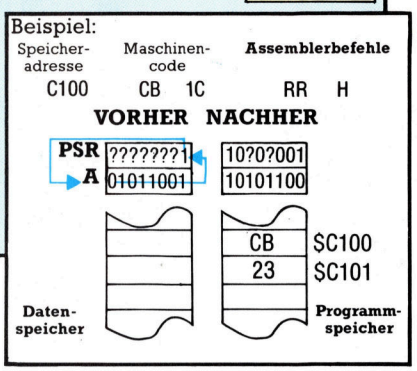
Der Inhalt des Bytes wird über den Übertrag um ein Bit nach rechts verschoben.

Bewirkt beim PSR:

	S Z		H		V N C
MSB	X X		0		X 0 X LSB

Beispiel:

Speicheradresse	Maschinencode	Assemblerbefehle
C100	CB 1C	RR H



SBC – Subtrahieren mit Übertrag

6502

Absolut – ED (3 Bytes)

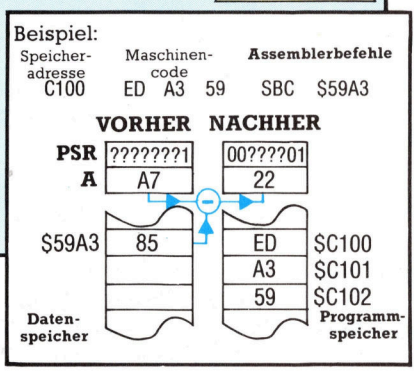
Der Inhalt der Speicherstelle wird vom Akku. subtrahiert. Der Übertrag zeigt den Status des Borgens.

Bewirkt beim PSR:

	S V		B D	I	Z C
MSB	X X				X X LSB

Beispiel:

Speicheradresse	Maschinencode	Assemblerbefehle
C100	ED A3 59	SBC $59A3



SBC – Subtrahieren mit Übertrag

Z80

Konstante – DE (2 Bytes)

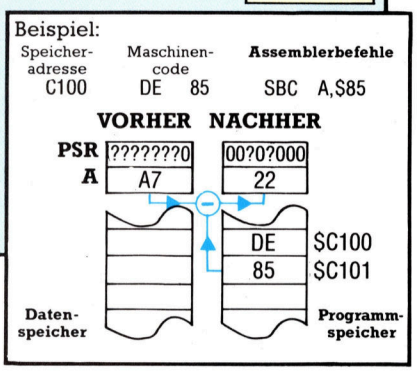
Der Inhalt der Speicherstelle, die dem Op-code folgt, wird vom Akku subtrahiert.

Bewirkt beim PSR:

	S Z		H		V N C
MSB	X X		X		X X LSB

Beispiel:

Speicheradresse	Maschinencode	Assemblerbefehle
C100	DE 85	SBC A,$85



ASL – Arithm.nach links schieben

6502

Register – OA (1 Byte)

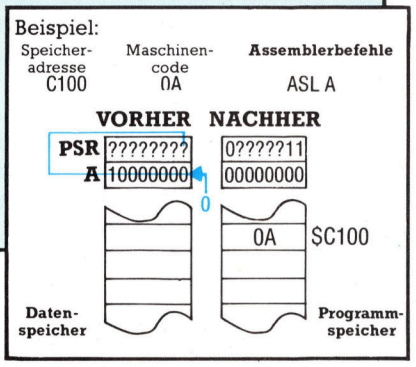
Der Inhalt des Bytes wird um ein Bit nach links in den Übertrag verschoben; das niederwertigste Bit wird Null.

Bewirkt beim PSR:

	S V		B D	I	Z C
MSB	X				X X LSB

Beispiel:

Speicheradresse	Maschinencode	Assemblerbefehle
C100	0A	ASL A



SLA – Arithmetisch nach links schieben

Z80

Register – CB (2 Bytes)

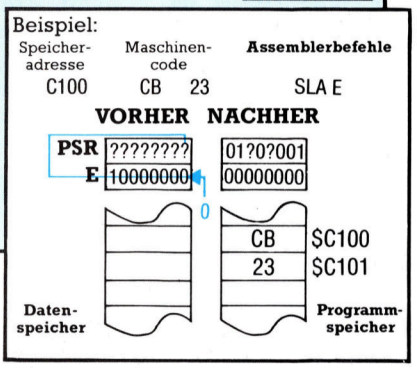
Der Inhalt des Bytes wird um ein Bit nach links in den Übertrag verschoben; das niederwertigste Bit wird Null.

Bewirkt beim PSR:

	S Z		H		V N C
MSB	X X		0		X 0 X LSB

Beispiel:

Speicheradresse	Maschinencode	Assemblerbefehle
C100	CB 23	SLA E



CMP – Mit Akkumulator vergleichen

6502

Absolut – CD (3 Bytes)

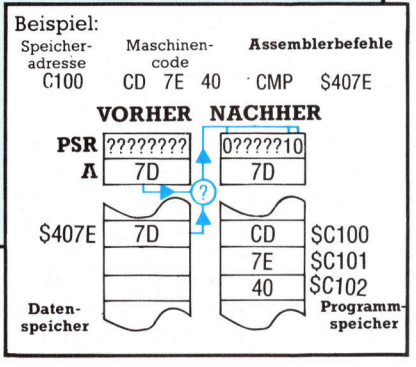
Der Inhalt der Speicherstelle wird mit dem Akkumulator verglichen.

Bewirkt beim PSR:

	S V		B D	I	Z C
MSB	X				X X LSB

Beispiel:

Speicheradresse	Maschinencode	Assemblerbefehle
C100	CD 7E 40	CMP $407E



CP – Mit Akkumulator vergleichen

Z80

Konstante – FE (2 Bytes)

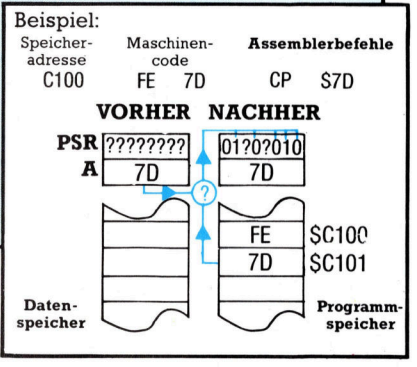
Der Inhalt der Speicherstelle, die dem Op-code folgt, wird mit dem Akku verglichen.

Bewirkt beim PSR:

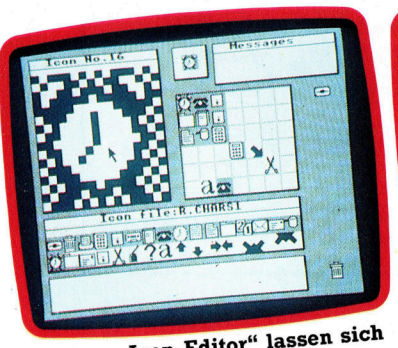
	S Z		H		V N C
MSB	X X		X		X 1 X LSB

Beispiel:

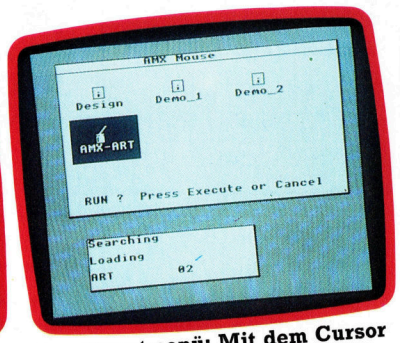
Speicheradresse	Maschinencode	Assemblerbefehle
C100	FE 7D	CP $7D



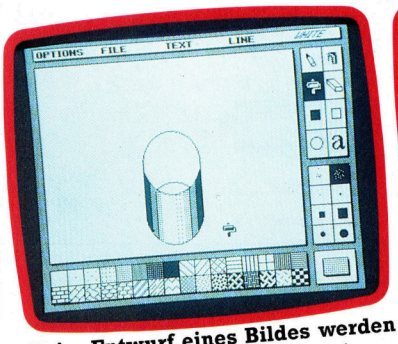
Piktogramme



Mit dem „Icon-Editor" lassen sich maßgeschneiderte Piktogramme (Icons) erzeugen.



Das Hauptmenü: Mit dem Cursor kann eine der vier Varianten gewählt werden.



Beim Entwurf eines Bildes werden zuerst die Umrisse gezeichnet.

Für die Linienstruktur gibt es ein weiteres Menüangebot.

Die „Maus" als Eingabegerät ist schon ein paar Jahre alt. Das AMX-Maus-Paket enthält ein speziell für den Acorn B entwickeltes Malprogramm, das wegen seiner einfachen Handhabung sicher viel Zuspruch finden wird.

Das AMX-Maus-Paket besteht aus der Maus selbst, einem ROM mit der Betriebssoftware, Anwenderprogrammen auf Cassette und Diskette, einer allgemeinen Anleitung und einem Handbuch für das Malprogramm. Im Gebrauch zeigt das System eine starke Ähnlichkeit mit der Macintosh-Software von Apple.

Das AMX-Maus-Paket für den Acorn B enthält außer der Maus selbst ein Anschlußkabel für den User Port, ein ROM, zwei Handbücher sowie Software auf Cassette und Floppy.

Die schwarze Plastik-Maus ist ein japanisches Erzeugnis. Sie liegt gut in der Hand, wobei Zeige-, Mittel- und Ringfinger die drei Tasten „Move", „Execute" und „Cancel" (Bewegen, Ausführen, Löschen) bedienen. Die Metallkugel, mit der die Maus „läuft", ist jedoch ein Mißgriff des Konstrukteurs, denn sie rutscht auf glatten Holz- und Kunststoff-Tischen, statt zu rollen.

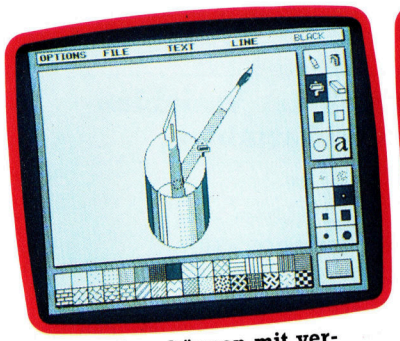
Das Innenleben der Maus besteht im wesentlichen aus zwei Walzen, die die Kugel umgeben. Die Walzen tragen an ihren Enden Schlitzscheiben. Wenn die Scheibe rotiert, wird ein Lichtstrahl periodisch unterbrochen, und der Lichtdetektor gibt an den Rechner eine Folge elektrischer Impulse ab, die der Bewegung der Kugel entspricht. Diese Impulse werden in numerische Werte umgewandelt, die anschließend vom System gelesen und entsprechend verarbeitet werden.

Das Betriebssystem im mitgelieferten ROM sorgt für die Datenübertragung zwischen Rechner und Maus. Abgesehen von Prozeduren für die Einbindung in kommerzielle Software enthält es auch mehrere hilfreiche Routinen für die Verwendung der Maus in selbstgeschriebenen Programmen. Dazu gehören unter anderem die Angabe der aktuellen Position der Maus (in Form von Bildschirmkoordinaten), die Abfrage der Bedienknöpfe und die Bewegung von Piktogrammen auf dem Bildschirm. Unter Verwendung dieser Routinen – die alle über die "*"-Befehle des Acorn-Betriebssystems aufgerufen werden – kann der Benutzer mühelos BASIC-Programme schreiben, die die Möglichkeiten der Maus-Steuerung voll ausschöpfen.

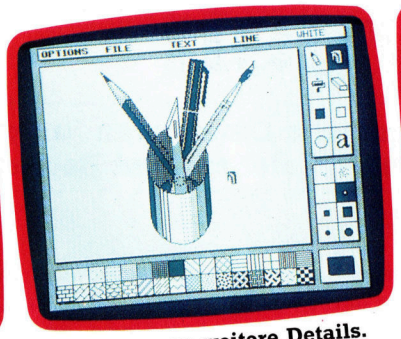
Maus ersetzt Cursortasten

In Verbindung mit vorhandener kommerzieller Software kann die Bewegung der Maus die Funktion von Cursortasten übernehmen, und die drei Bedienknöpfe können verschiedene Funktionen aufrufen und beliebige Keyboard-Tasten ersetzen, einschließlich der Shift- und CTRL-Tasten.

In vollem Glanz zeigt sich die Maus aller-



Die Umrisse können mit verschiedenen Flächenmustern ausgefüllt werden.



Hinzu kommen weitere Details.

Wörter und Buchstaben lassen sich im „Text Mode" einfügen.

Die letzten Feinheiten ergänzen Sie am besten frei Hand.

dings erst im Zusammenwirken mit Software, die speziell für die Steuerung per Maus geschrieben wurde. Das AMX-Paket enthält zwei derartige Programme, die umfassend demonstrieren, wie man mit der Maus arbeiten kann. Das eine ist ein BASIC-Programm zum Entwurf von Piktogrammen, die in eigene Programme eingebaut werden können.

Piktogramme nach Wunsch

Die Piktogramm-Software präsentiert Ihnen zunächst ein Raster mit 16 × 16 Elementen, in dem Sie durch Ausfüllen der gewünschten Felder eigene Piktogramme entwerfen können. Die Tastatur wird nur benötigt, um den Dateinamen einzugeben, unter dem eine Piktogramm-Serie gespeichert oder aufgerufen werden soll. Noch besser kommen die Vorteile der Maus aber beim AMX-Malprogramm zur Geltung, das ebenfalls zum Lieferumfang gehört. Es ist ein komplettes Paket für grafische Entwürfe mit ausschließlicher Maus-Eingabe. In der Handhabung ist es dem „Mac paint"-System für den Apple Macintosh durchaus ebenbürtig. Auch die Schirmaufteilungen sind beim AMX-System dem des Macintosh sehr ähnlich.

Zum Malen wird beim Acorn B der „Mode 4"-Bildschirm (zweifarbig, 320 × 256 Punkte) aufgerufen. Am rechten Schirmrand werden zwei Piktogramm-Menüs angeboten: Zeichenhilfen wie Linienziehen, Löschen, Farbsprüh-Effekte und Ausfüllen von Feldern, die jeweils durch ein leicht verständliches Symbol dargestellt sind. Zum Aufruf einer Funktion wird der Cursor auf das entsprechende Piktogramm gesetzt und dann einer der Bedienknöpfe betätigt. Beim Aufbau des Bildes führen Sie den Cursor über die Malfläche und können dabei mit Hilfe der Bedienknöpfe beliebige Zeichenfunktionen ausführen lassen. Da der Cursor alle Bewegungen der Maus exakt nachvollzieht, können Sie mit der Software wie selbstverständlich arbeiten. Die beigefügte (ausgezeichnete) Anleitung ist also fast überflüssig.

Das zweite Menü dient zur Auswahl von Strichstärken und geometrischen Figuren. Den unteren Bildrand säumt ein drittes Menü mit diversem Füllmaterial für das Belegen von Feldern – von schlichtem Schwarz bis zu komplizierten Gittermustern. Weil die Wiedergabe wegen der begrenzten Speicherkapazität nur zweifarbig erfolgt, sind diese Strukturen zur gegenseitigen Abgrenzung von Flächen sehr angebracht.

Andere Hilfsfunktionen wie SAVE und LOAD (Speichern und Laden von Bildern), etliche Systembefehle sowie der Aufruf des integrierten Hardcopy-Programms können über die Menüs am oberen Bildrand angewählt werden. Normalerweise sind nur die Titel dieser Menüs zu sehen, aber wenn Sie den Cursor daraufsetzen und einen Bedienknopf drücken, wird das ganze Angebot auf dem Bildschirm dargestellt. Nachdem Sie mit der Maus Ihre Wahl getroffen haben, verschwindet das Menü, und es erscheint wieder das ursprüngliche Bild.

Bereits nach kurzer Zeit arbeiten Sie mit dem System genauso bequem wie mit Zeichenfeder und Papier, abgesehen davon, daß die AMX-Malfeder sich nicht breitdrückt oder kleckst und daß das Bild beliebig oft reproduzierbar ist.

Software in Vorbereitung

Das Malprogramm allein wäre Grund genug, sich das Maus-Paket anzuschaffen. AMX plant, hiervon ausgehend ein vollständig mausgesteuertes Softwaresystem für den Acorn B zu entwickeln. In Vorbereitung ist ein „Desk-top"-Programm, das ähnlich wie bei den Apple-Rechnern Macintosh und Lisa funktioniert. Ferner sind eine mausgesteuerte Datenbank und ein weiterer Ausbau des AMX-Malprogramms mit differenzierter Farbwahl und Vergrößerungstechnik vorgesehen.

Die AMX-Maus und das Malprogramm wie auch die in Kürze erscheinende Maus- und Piktogramm-orientierte Software kommen allen entgegen, die aus ihrem Acorn B mehr machen wollen. Mit ihrer Benutzerfreundlichkeit setzen sie neue Maßstäbe.

AMX-MAUS

ABMESSUNGEN

85 × 65 × 35 mm

ANSCHLUSS

Schnittstellenkabel für das User Port des Acorn B

ANLEITUNG

Zwei leicht verständliche und informative Handbücher.

STÄRKEN

Einfache Bedienung; hervorragende Anleitung; vielseitiges Malprogramm mit sorgfältiger Abstimmung auf das Rechner-Betriebssystem.

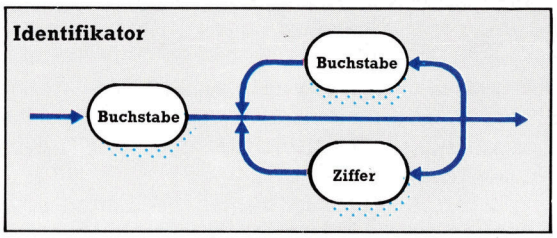
SCHWÄCHEN

Abgesehen vom Malprogramm sind die Möglichkeiten der Software begrenzt; die Metallkugel der Maus neigt auf glatten Flächen zum Rutschen.

Datentypen

PASCAL verwendet vier einfache Datentypen innerhalb der Variablen-Deklarationen: Integer-, Real-, Zeichen- und Boolesche Variablen.

PASCAL bietet vier vordefinierte Datentypen, die als Integer, Real, Char und Boolean bezeichnet werden. Zahlen werden als reale Zahlen bezeichnet, wenn sie einen abtrennbaren Teil haben („Real"). Natürliche ganze Zahlen sind „Integer". Der tatsächlich verfügbare Bereich von Zahlen ist davon abhängig, wieviele Bytes zum Speichern des jeweiligen Datentyps verwendet werden.



Der Bereich der Integer-Zahlen wird bei Ihrem Compiler entweder von -32 768 bis 32 767 oder von -2.147.483.648 bis 2.147.483.647 reichen. Dies hängt davon ab, ob zum Speichern zwei oder vier Bytes verwendet werden. PASCAL bezeichnet den vorgegebenen Wert der höchsten Integer-Zahl als MaxInt. Somit läßt sich dieser Wert mit der folgenden Anweisung herausfinden:

```
WriteLn ('MaxInt ist: ',MaxInt)
```

Reale Zahlen können ebenfalls nur in einem bestimmten Bereich und mit einer bestimmten Genauigkeit gespeichert werden. Er reicht im allgemeinen von minus bis plus 1,7E38 mit einer Genauigkeit von mindestens sechs bis sieben Stellen nach dem Komma.

Beachten Sie, daß eine reale Zahl (für den Rechner) immer einen Dezimalpunkt haben muß (in normaler Schreibweise ein Dezimalkomma), der die ganze Zahl von den Dezimalstellen trennt. Beides muß eingegeben werden. Somit sind 0.1 und 1.0E-1 möglich, doch .1 oder 1E-1 sind illegal.

„Char" ist die Abkürzung für Character. Ein Wert dieses Datentyps ist ein Zeichen aus dem ASCII-Satz des Computers. PASCAL gewährleistet seine eigene Anpassungsfähigkeit, indem es folgende Punkte festlegt:

– Die Zeichen A bis Z werden alphabetisch geordnet. Das bedeutet, daß A einen kleineren Wert hat als B, B einen kleineren Wert als C und so weiter.

– Die Zahlen-Zeichen 0 bis 9 werden der bekannten Reihenfolge nach geordnet.

Jedes ASCII-Zeichen erhält einen numerischen Code, der ein Wert des untergeordneten Bereiches der Integer-Datentypen ist. Die ASCII-Codes sind in einem Bereich von 0 bis 127 definiert. Bei vielen Computern geht der Bereich sogar bis 225, wobei die zusätzlichen Codes spezielle Grafikzeichen repräsentieren. Man kann jeden gewünschten Zeichensatz problemlos auf der Skala der Ordnungszahlen darstellen, die vom Computer intern verwendet werden. PASCAL verfügt über die vordefinierte Funktion Ord: Sie gibt als Ergebnis den Integer-Wert des angegebenen Argumentes aus. Somit ist Ord(A) das Äquivalent des Wertes 65 im ASCII-Zeichensatz. Eine andere Funktion, chr, kehrt diesen Vorgang um – chr (65) ergibt das Zeichen A.

Der Bereich der „Zeichen"-(Char) und der Integer-Werte ist für jede PASCAL-Version fest definiert, das heißt, er basiert auf einer geordneten Skala bekannter Konstanten. Aus diesem Grund werden sie als „Ordnungs"- bzw. „Skalar"-Datentypen bezeichnet. Ganz gleich, welcher Wert vorliegt, man kennt immer den vorangegangenen und folgenden Wert. Diese angrenzenden Werte können durch die beiden folgenden Skalar-Funktionen ganz einfach ermittelt werden.

```
pred(item) (vorangegangener Wert)
succ(item) (nachfolgender Wert)
```

Somit ergibt succ(3) als Ergebnis immer den Zeichenwert 4, wogegen pred(Z) nur bei bestimmten Zeichensätzen (zum Beispiel ASCII) Y als Ergebnis ausgibt. Pred(MaxInt) ergibt entweder den Wert 32 766 oder 2.147.483.646. Die chr-Funktion kann nur mit einem Argument verwendet werden, das ein Zeichen-Code ist. Alle anderen Skalar-Funktionen können dagegen mit beliebigen Skalar-Datentypen verwendet werden.

Boolesche Variablen

„Boolesche Variablen" sind die einfachsten aller Skalar-Datentypen, da nur zwei Werte möglich sind – falsch und wahr (in dieser Reihenfolge). Die Skalar-Funktionen können daher auf jeden dieser Werte angewendet werden – der Ordnungs-Wert von falsch ist 0, und ord(wahr) ergibt 1.

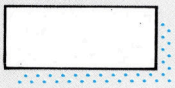
Der folgende Programmteil zeigt alle einfa-

Symbole

Die folgenden drei Symbole werden allgemein in PASCAL-Syntax-Diagrammen verwendet.

● Das ovale Symbol repräsentiert reservierte PASCAL-Worte oder Zeichen, die keiner weiteren Erklärung bedürfen (wie "Buchstabe" oder "Ziffer").

● Ein Kreis repräsentiert einen Pascal-Operator (+, -, *, . usw.).



● Das rechteckige Symbol steht für ein Wort oder einen Abschnitt, der sein eigenes separates Syntax-Diagramm hat.

chen Ordinal-Datentypen, die in einem PASCAL-Programmtext auftreten können.

```
CONST
  VAT         =0.15;
  spalten     =40;
  leerstellen ='';
  fehlersuche =falsch;
```

Mit dem Gleichheits-Zeichen werden die Identifikatoren bestimmten Werten zugeordnet. Der Doppelpunkt(:) trennt im VAR-Abschnitt neu deklarierte Variablen-Identifikatoren vom jeweiligen Datentyp. Betrachten Sie das folgende Beispiel:

```
VAR
  verhaeltnis :real;
  zahl        :integer;
  symbol      :char;
  fertig      :boolean;
```

Der Zuordnungs-Operator

Wenn wir diesen Variablen Werte zuordnen wollen, wird der sogenannte zusammengesetze „Zuordnungs-Operator" (:=) verwendet. Dies ist für die eindeutige Unterscheidung zwischen den drei Arten von Operationen sehr hilfreich.

CONST-Definitionen ordnen permanente Werte zu. VAR-Deklarationen reservieren lediglich Speicherplatz. Durch Zuordnung erhält der Identifikator einen Wert.

Wenn zwei oder mehrere Anweisungen als Teil eines Arbeitsvorganges ausgeführt werden müssen, kann man sie als „zusammengesetzte Anweisung" (durch ein Semikolon getrennt) zwischen die Worte BEGIN und END schreiben. Sie haben sicherlich bereits zusammengesetzte Anweisungen gesehen; denn diese Form kommt in jedem umfangreicheren Programm vor. Im folgenden sehen Sie ein PASCAL-Programm, das viele der bereits besprochenen Details beinhaltet. Übrigens werden ab dieser Folge die reservierten Wörter in Großbuchstaben geschrieben, damit Sie sie besser von den Identifikatoren unterscheiden können.

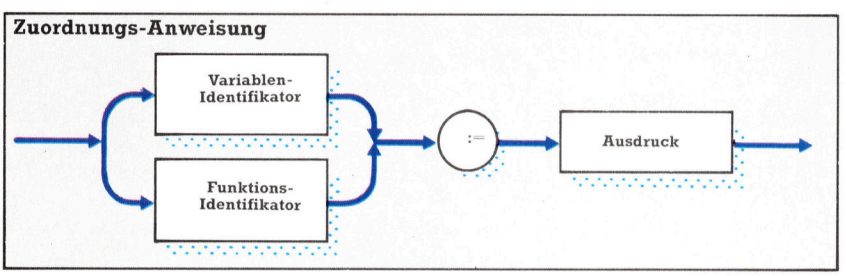
```
PROGRAM Kreis (input, output);
CONST
  pi          =3.1415926536;
  meldung     ='Gib Radius ein:';

VAR
  radius,
  umfang      :real;

BEGIN
  WriteLn;
  write(meldung);
  read (radius);
  umfang := pi*radius*radius;
  WriteLn;
  WriteLn ('Der Umfang eines Kreises',
    'mit dem Radius', radius:8:3);
  WriteLn ('betraegt:', umfang:10:3)
END.
```

In diesem Beispiel gibt es zwei Aspekte, die beachtet werden sollten. Der erste ist, daß im VAR-Abschnitt zwei Identifikatoren desselben Datentypes deklariert werden – beide real. Wie Sie sehen, sind zwei separate Deklarationen nicht notwendig, da eine größere Anzahl von Namen in PASCAL einfach durch Kommata getrennt werden kann. Der zweite Aspekt ist die Art der Bildschirmausgabe. Sie wird verwendet, um die vorgegebene wissenschaftliche Schreibweise von realen Zahlen zu umgehen. Wie Sie sehen, kann man ganz nach Wunsch zwei Integer-Zahlen, getrennt durch Doppelpunkte, spezifizieren. Damit wird eine bestimmte Feldbreite zur Darstellung der gesamten Zahl und ihres abtrennbaren Teiles festgelegt.

In unserem Kreis-Programm sind sowohl für den Radius als auch für den Umfang drei Dezimalstellen vorgesehen. Da der Umfang eine größere Zahl sein wird, sind insgesamt zehn Zeichenpositionen vorgesehen, zwei mehr als für den Radius. Diese Integer-Werte müssen ein eventuelles Vorzeichen sowie die Darstellung des Dezimalpunktes vor dem abtrennbaren Teil zulassen.

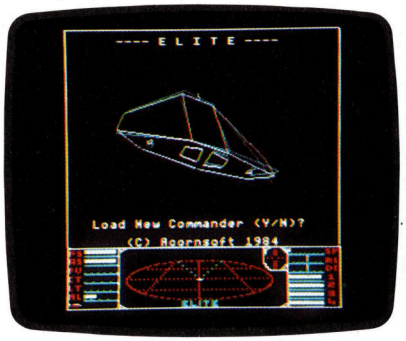


PASCAL rundet automatisch die letzte Stelle, um die maximale Genauigkeit in einem bestimmten numerischen Bereich anzugeben. Außerdem kann jede Variable oder jeder Ausdruck verwendet werden, – nicht nur Konstanten. Bei allen anderen Datentypen wird nur ein Integer-Wert zur Bestimmung der Darstellungsform benötigt. Wenn man eine Feldbreite von eins festlegt, werden Integer-Werte ohne Leerstellen ausgegeben. Soll die Bildschirmausgabe tabellarisch gegliedert sein, müssen Sie daran denken, zusätzliche Leerstellen einzusetzen. Betrachten Sie das folgende Beispiel:

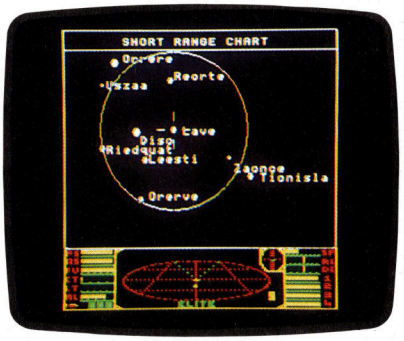
```
WriteLn ('Total:' :20,gewicht:1,' Kg.')
```

Achten Sie darauf, zum Trennen der einzelnen Ausgaben Leerstellen einzufügen. Wenn Sie beispielsweise die Zahlen 12 und 34 in dieser Form ausgeben lassen, so erhalten Sie auf dem Bildschirm 1234.

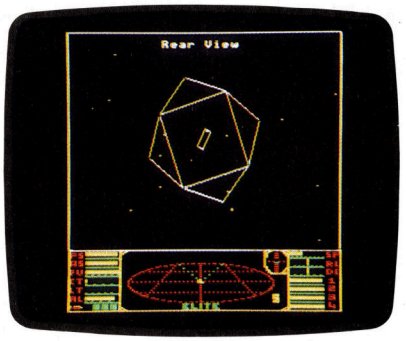
Die Aufgabe beginnt



Die Übersichtskarte



Die Raumstation



Hier sehen Sie drei Screens von Elite. Der Auftaktbildschirm stellt das Cobra-Mark-III-Schiff dar. Das Spiel kann Monate währen, bis man ans Ende gelangt. Deshalb wurde die SAVE-Funktion integriert. Am unteren Bildschirmrand findet man mehrere Indikatoren, die angeben, was man auf der Reise benötigt, sowie eine dreidimensionale Karte des Gebietes. Der zweite Schirm zeigt die Sterne in bezug auf die gegenwärtige Position. Durch Steuerung des Zielkreuzes auf einen bestimmten Stern gibt ein neuer Bildschirm sämtliche Daten über den Planeten – Regierungsform, Wirtschaftstyp usw. Im dritten Screen ist die Raumstation abgebildet. Um an die Station anzudocken, muß der Spieler sein Schiff in eine kleine rechtwinklige Schleuse steuern.

Handel im All

„Elite", ein Programm, das man inzwischen als Kultspiel bezeichnen kann, setzte einen neuen Programmier-Standard. Die Elemente von schneller Aktion in 3 D, Abenteuer und Strategie wurden zu einer echten Herausforderung kombiniert.

In Elite sind alle Qualitäten guter Spiele vereint. Ziel ist es, Mitglied einer Elite-Truppe zu werden. Um das zu schaffen, muß der Spieler ein erfolgreicher Händler zwischen Tausenden von Planeten in acht Galaxien werden.

Damit diese Aufgabe bewältigt werden kann, ein Vermögen gemacht und die Zugehörigkeit zur Elite erlangt wird, ist nicht nur Voraussetzung, daß der Spieler zu einem ausgezeichneten Navigator und Kämpfer wird. Er muß auch das richtige Gespür für ein gutes Geschäft bekommen, über ein gewisses Maß an Gewitztheit verfügen und auf dem schmalen Grad zwischen Legalität und Illegalität arbeiten können. Viel Geld läßt sich schnell machen, wenn man zwischen den Planeten mit Schmuggelware handelt oder Pirat wird. In diesem Fall heißt es für den Spieler jedoch vorsichtig sein. Die Polizei ist schwer zu erkennen und immer im Bilde, und schon bald findet man sich als steckbrieflich Gesuchter wieder.

Bei Spielbeginn befindet sich der Spieler im Dock einer Raumstation über dem Planeten Lave. Das Fahrzeug ist ein Cobra Mark III, ausgerüstet mit einem Front Puls Laser und einer Ladekapazität von 20 Tonnen. Vor dem Start sollte man sich mit den Bedienungseinheiten vertraut machen und Dockmanöver üben, da man sonst schon bei der ersten Mission eine Bruchlandung hinlegt.

Nach dem Docken geht das Geschäft los. Nach Vergleichen der Preise, die auf dem Planeten für die mitgeführten Waren gelten, ist zu entscheiden, ob man warten oder gleich verkaufen will. Vielleicht gibt es an anderen Orten noch viel bessere Gewinne.

Die eigentliche Handelsaufgabe besteht darin zu entscheiden, welche Produkte auf welchem Planeten benötigt werden. So kann man Nahrungsmittel billig auf einem landwirtschaftlich orientierten Planeten erwerben und für gutes Geld auf einem Planeten mit Industrie verkaufen. Maschinen können wiederum auf dem Agrar-Planeten abgesetzt werden. Dazu muß der Spieler die Staatsform des Planeten in Betracht ziehen und ebenso, welche Art von Lebewesen ihn bewohnen.

Die dreidimensionalen Grafiken von Elite sind ausgezeichnet. Planeten, Raumstationen und Raumschiffe sind in Diagrammform dargestellt, an die man sich anfangs gewöhnen muß. Bald aber wird der Vorteil größerer Auflösung und die Möglichkeit, sich in drei Dimensionen drehen zu können, deutlich. Beim Kampf wie beim Docken ist es besonders wirkungsvoll. Denn Raumstationen haben nur eine Schleuse, die in einem ganz bestimmten Winkel angeflogen werden muß.

Neben dem Programm auf Cassette oder Diskette wird ein umfangreiches, detailliertes Handbuch mitgeliefert, das alle Gesichtspunkte des Spiels aufzeigt. Ferner enthält es die dem Spiel zugrundeliegende Geschichte „The Dark Wheel", eine Karte, auf der die Kontrollfunktionen zusammengefaßt dargestellt sind, einen Überleger für die Funktionstasten sowie ein Poster, auf dem verschiedene Typen von Raumfahrzeugen dargestellt sind. Seit kurzem ist Elite auch für den C 64 erhältlich. Bei dieser Version wurden sämtliche Programmtexte sowie das Handbuch ins Deutsche übersetzt.

Elite: Für Acorn B, Electron (wahlweise für Joysticks) und Commodore 64

Hersteller: Acornsoft Ltd., Betjemin House, 104 Hills Road, Cambridge, CB2 1LQ

Autoren: Ian Bell, David Braben

Format: Cassette oder Diskette

Schreiben für den Bildschirm

Viele Heimcomputer-Programmierer träumen davon, einen Bestseller zu schreiben. Doch sie sind sich selten dessen bewußt, welcher Aufwand damit verbunden ist.

Natürlich bedeutet der Besitz einer teuren Ausstattung noch lange nicht, daß auch die Programme erfolgreich sind. Es gibt Amateure, die mit ihrer auf dem kleinen Spectrum zu Hause entwickelten Software ein Vermögen verdient haben. Und doch sind diese „Heimhexer" auf dem Computer vom „Aussterben" bedroht, wenn man die Entwicklung der großen Software-Häuser in den vergangenen Jahren betrachtet. Dank ihrer Großrechner und ausgetüftelten Programmierhilfen sind sie den Amateur-Programmierern gegenüber im Vorteil.

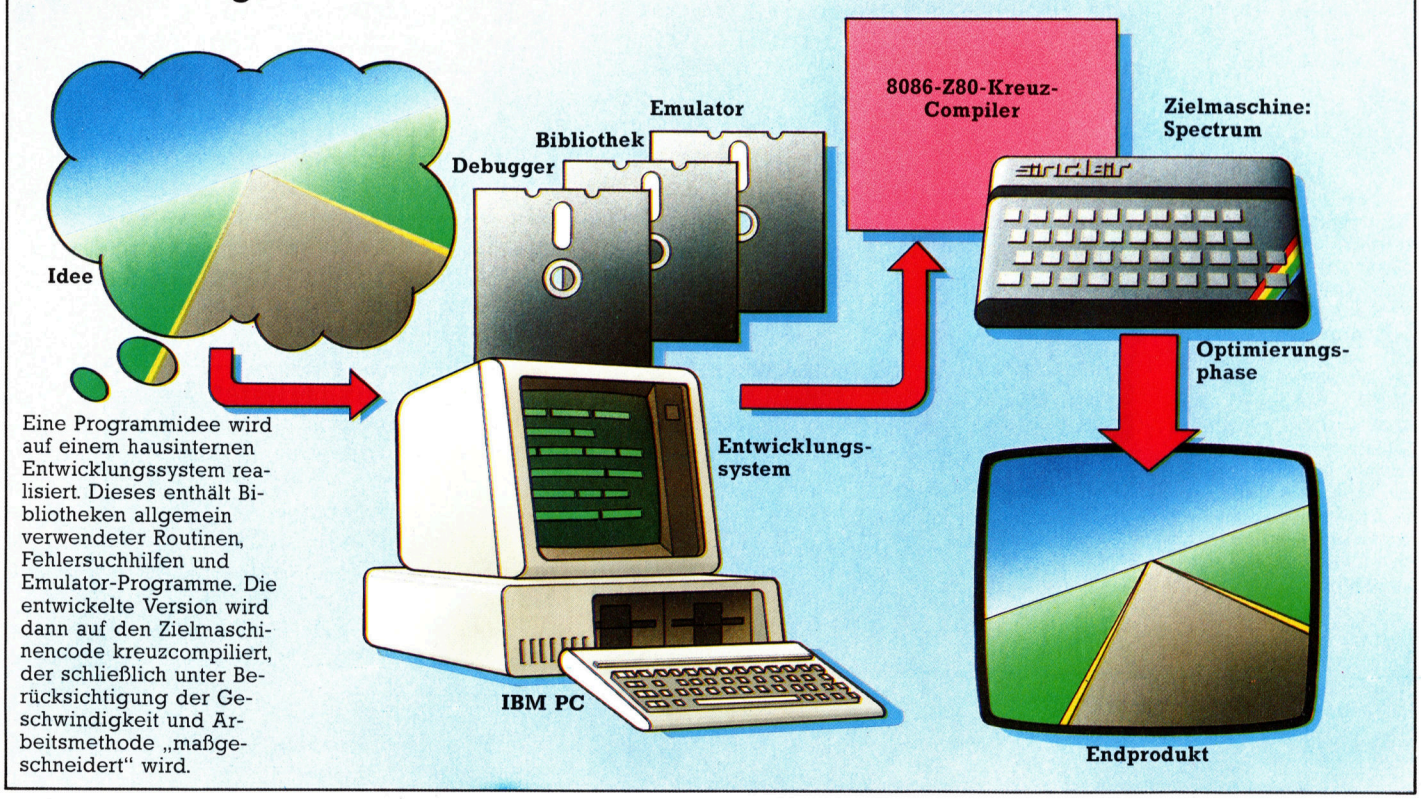
Einer der wichtigsten Punkte bei der Software-Entwicklung für den Heimcomputermarkt ist die Verarbeitungsgeschwindigkeit. Daraus folgt, daß die Programme in Maschinensprache geschrieben sein müssen. Doch die Maschinensprache zu beherrschen, ist nicht leicht. Programmierer, die damit arbeiten, benötigen besondere Software, um ihre Programme schreiben zu können. Zumindest ist ein Assembler-Programm zur Übersetzung des Quellcodes in den Objektcode erforderlich, den die Maschine versteht. Die Aufgabe ist besonders schwer, wenn das Programm umfangreich ist.

Heimcomputer genügen nicht

Die Qualität der für Heimcomputer zur Verfügung stehenden Assembler-Programme ist recht dürftig. Selbst die einfachsten Programmpakete dieser Art belegen viel Speicherplatz, womit auch die Programmlänge begrenzt wird. Viele Heimcomputer haben konstruktionsbedingte Nachteile, wie schlechte Tastaturen oder mangelhafte Bildschirmdarstellung, die das Erarbeiten von Programmen zur Qual machen.

Aus diesen Gründen schreiben die meisten kommerziellen Unternehmen die Programme nicht auf den Rechnern, für die sie gedacht

Entwicklungsabläufe



Eine Programmidee wird auf einem hausinternen Entwicklungssystem realisiert. Dieses enthält Bibliotheken allgemein verwendeter Routinen, Fehlersuchhilfen und Emulator-Programme. Die entwickelte Version wird dann auf den Zielmaschinencode kreuzcompiliert, der schließlich unter Berücksichtigung der Geschwindigkeit und Arbeitsmethode „maßgeschneidert" wird.

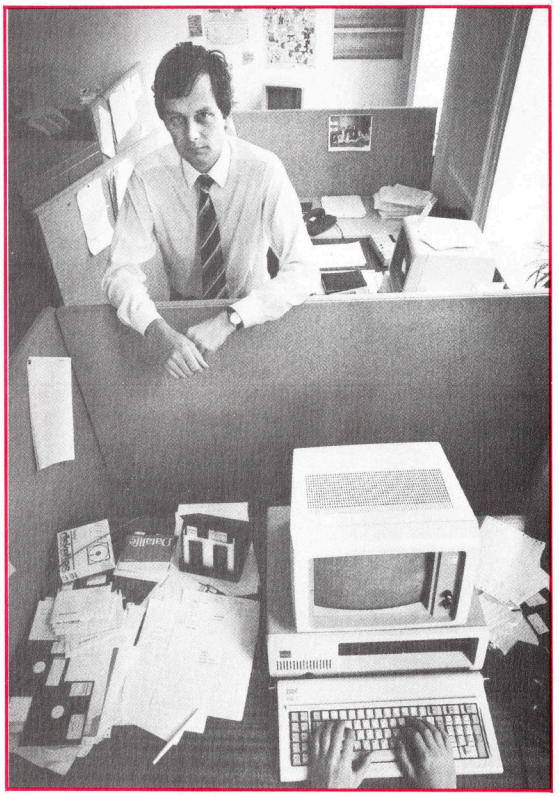
Auf strategische Spiele wie Schach spezialisiert, verwendet IS bei der Software-Entwicklung IBM- und Apple-Computer, die mit selbstentwickelten Schnittstellen ausgestattet sind. Die Aufteilung der Programme in maschinenabhängige und universell einsetzbare Segmente vereinfacht es, gleichzeitig mehrere Computer und spezielle Schachcomputer mit Programmen zu versorgen.

sind (die sogenannte „Zielmaschine"), sondern benutzen zur Programmentwicklung professionelle Systeme mit spezieller Software (man nennt diese „Entwicklungssysteme"). Programmierer, die diese Systeme benutzen, schreiben in Sprachen wie PASCAL oder C. Dabei kommen Sprachversionen zur Anwendung, die als Kreuz-Compiler oder Kreuz-Assembler bezeichnet werden. Das bedeutet: Man kann auf einem Rechner arbeiten, der einen 8086-Prozessor hat, und die erzeugten Programme laufen auf einem Rechner, der mit einem Z80 ausgestattet ist.

Natürlich hat ein solches Entwicklungssystem erhebliche Vorteile gegenüber einem Heimcomputer. Ein Assembler auf Diskette oder ergänzende RAM-Kapazitäten zur Speicherung größerer Programme erleichtern die Arbeit. Debugging-Routinen können in die Rohversion des Programms geladen werden, ohne daß man Sorge haben muß, daß der Speicherplatz nicht reichen könnte. Dazu kommen die besseren Arbeitsmöglichkeiten auf einem Rechner mit guter Tastatur, brillanter Bildschirmdarstellung und Diskettenstationen.

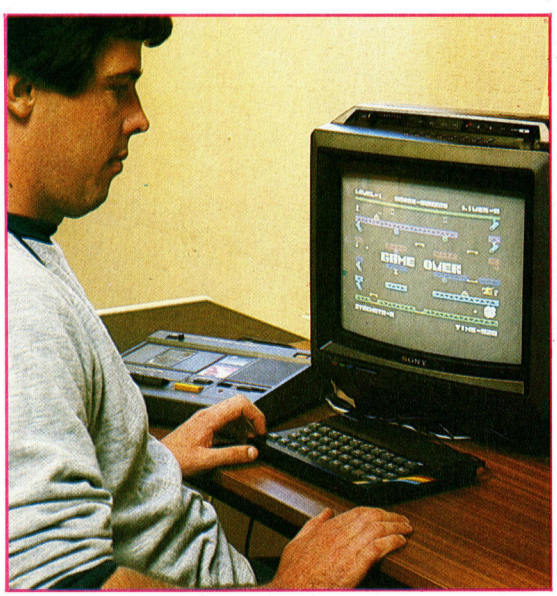
Eine der Firmen, die sich dieser Programmentwicklungstechnik bedient, ist Intelligent Software (IS). Sie wurde 1981 von erfahrenen Programmierern gegründet, so dem Schachspezialisten David Levy und Robert Madges von ANT Microware. Das Unternehmen ist auf strategische Spiele spezialisiert, die zumeist Auftragsproduktionen bekannter Heimcomputerhersteller sind. Das Haus entwickelt außerdem Software für Schachcomputer.

Neben den Computern, die für die Entwicklung eingesetzt werden, benutzt IS IBM PCs und Apple-Rechner mit speziell entwickelten Schnittstellen, die die Übertragung von Codes zwischen den verwendeten Rechnern ermöglichen. Das Unternehmen wird häufig damit beauftragt, Programme zu übertragen – etwa ein vorhandenes Schachprogramm auf einen anderen Computer umzuschreiben. Folglich mußten die Programmierer lernen, den Code in segmentierter Form zu schreiben. Ein Teil dieser Segmentierung hat sich als gerade dann besonders nützlich erwiesen, wenn der Code vom einen Prozessor auf den anderen übertragen wird. Und zwar ist dies die Teilung des Programms in einen Spielcode und einen Eingabe/Ausgabe-Code. Der Input/Output-Code eines neuen Rechners hat häufig einen anderen Port oder muß anders adressiert werden oder ist vielleicht sogar von der Strategie her anders (beispielsweise muß eine Unterbrechung erfolgen).



Da IS die Programmierer so wenig wie möglich einschränken will, gibt es für die Erstellung der Codes nur wenige „Hausregeln". Ein wichtiger Punkt, auf dem man besteht, ist jedoch, daß der Quellcode reichlich Anmerkungen enthalten soll, damit man immer weiß, was welche Routine tut.

Die meisten Vision-Programme werden von zu Hause arbeitenden Programmierern auf den Zielmaschinen entwikkelt. Nachdem Spielkonzept und Bilschirmdarstellung festgelegt wurden, erfolgt die Entwicklung der einzelnen Routinen in herkömmlicher Assembler-Sprache (Z80 oder 6502) unter Verwendung von Assemblern wie dem HiSoft DevPak auf dem Spectrum.



Gut ausgerüstet: Psion

Das Softwarehaus Psion benutzt Computer, die leistungsfähiger sind als der IBM PC. Psion hat unter den vielen britischen Softwarehäusern, die Programme für Heimcomputer entwickeln, eine Sonderstellung, da die meiste Arbeit auf Minicomputern ausgeführt wird.

Psion begann als Unternehmen, das Software für den ZX 81 schrieb – und verwendete den ZX 81 auch dafür. Als es an die Programmierung von „Horizons" ging, das mit jedem Spectrum geliefert wird, erwarb Psion einen TRS 80 mit Diskettenstation, einen Rechner,

der den gleichen Z80-Prozessor hat, und baute ein spezielles Interface für die beiden Maschinen. Doch im August 1982 befand das Haus, daß es unmöglich sei, jedesmal dann, wenn ein neuer Heimcomputer auf den Markt kommt, ein völlig neues und anderes Entwicklungssystem zu schaffen. Deshalb investierte man in leistungsfähigere Hardware mit hoher Rechengeschwindigkeit. Diese Hardware wird allen Problemstellungen gerecht, die künftig entwickelte Computer mit sich bringen. Man entschied sich für zwei Vax 750 und das Betriebssystem VMS von DEC.

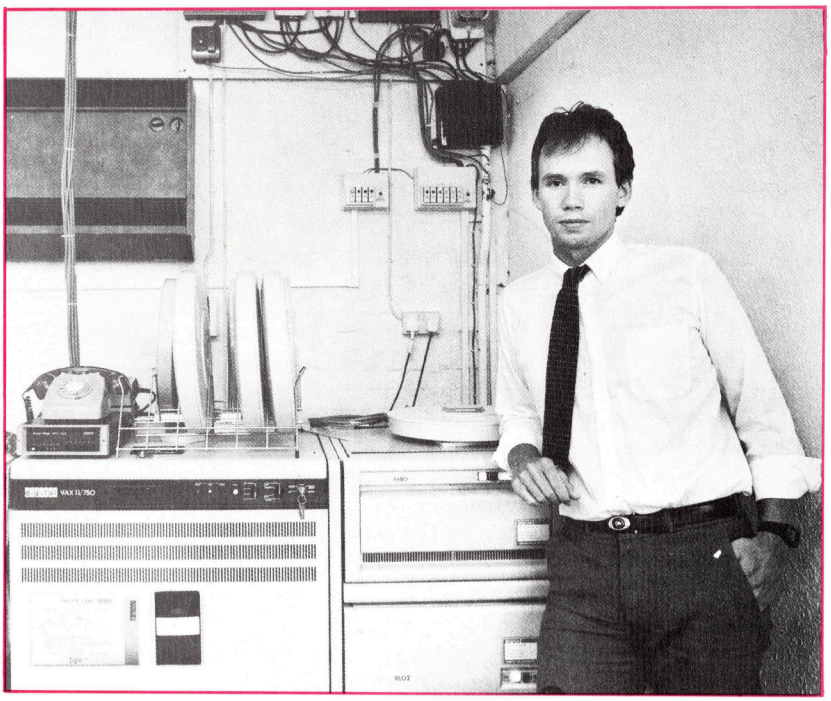
Vorteile der Vax

Die Vax 750 brachten Psion zwei Vorteile: die Qualität der Software, die DEC bietet, verbunden mit der Möglichkeit, speziell entwickelte Software-Hilfen zu kreieren, einerseits und die Leistungsfähigkeit des Betriebssystems und der Hardware-Kombination andererseits. Es gibt genügend Speicherplatz für eine ganze Software-Hilfs-Bibliothek, wie Compiler, Sammlungen üblicher Subroutinen und Fehlerbeseitigungs-Programme, die sechzehn bis zwanzig Programmierer gemeinsam nutzen können, die gleichzeitig am System [illegible]n".

Die neueren Vax-Rechner erlauben Autorenteams gemeinsames Arbeiten, Teilung von Projektbibliotheken, aus denen fast augenblicklich Module abgerufen werden können, und die vorhandenen Bibliotheken können sogar gleichzeitig von an unterschiedlichen Projekten arbeitenden Teams genutzt werden. Darin besteht der große Vorteil dieses sogenannten „Timesharing"-Systems. Dazu kommt, daß darauf die gesamte Verwaltungsarbeit erledigt werden kann, ohne daß die Programmierer ihre Arbeit unterbrechen müssen. Psion beabsichtigt, eine dritte Vax zu installieren, die ausschließlich für die Abwicklung administrativer Arbeiten verwendet werden soll.

Selbst wenn man es sich leisten könnte, so würde man durch den Erwerb einer Vax nicht automatisch zum Konkurrenten von Psion werden. Nur ein Bruchteil des verwendeten Materials wurde Psion von DEC als Fertigprogramm geliefert. Harte Arbeit war erforderlich, um einfache Aufgaben wirtschaftlich durchführen zu können. In C geschriebene Software-Hilfen und Utilities hat Psion selbst dem Grundsystem hinzugefügt.

Psion verwendet C, eine „Mittelhoch"-Sprache, mit der sehr kompakte und schnelle Objekt-Codes für 16-Bit-Chips wie den 8086 geschrieben werden können. Die Firma mußte jedoch eigene Techniken entwickeln, um Programme für Zielmaschinen wie den Spectrum erstellen zu können. Natürlich gibt das Unternehmen seine Geheimnisse nicht preis. Doch man weiß, daß Psion C benutzte, um einen eigenen Compiler zu schreiben, der einfach „Tischsprache" genannt wird.



Psion erwarb 1982 zwei Vax-750-Minicomputer, um eine Basis für die Softwareentwicklung zu schaffen. An jedem Rechner können bis zu 20 Programmierer gleichzeitig arbeiten und sämtliche Kreuz-Compiler nutzen, zudem Software-Bibliotheken und Fehlersuchprogramme, um so Programme erstellen und übersetzen zu können.

Es gibt eine Grundregel, die besagt, daß die Systemwartung und die Erstellung von Programmierhilfen wie der „Tischsprache" etwa 30 Prozent aller Programmierarbeit in Anspruch nehmen. Psion sieht diesen Zeitaufwand als gerechtfertigt und die Investition als Anlage an. Ein im eigenen Haus entwickelter Quellcode bietet zahlreiche Vorteile: Man kann ihn zerlegen und verbessern oder auf eine Art umwandeln, die bei der Verwendung fertig konfektionierter Software ganz unmöglich ist. Bei Software, die von Dritten erstellt wurde, ist es schwer, wenn nicht unmöglich, einmal vorhandene Fehler auszumerzen.

Die von Psion erworbene Software beinhaltet Programme, die exakte Simulationen populärer Microprozessoren wie des Z80 oder des 6502 sind. So lassen sich die DEC-Vax-Computer programmieren, als handelte es sich bei ihnen um einen Commodore 64 oder einen Spectrum. Trotz der Kapazität der Vax-Rechner laufen die Simulatoren nur mit einem Bruchteil der Geschwindigkeit der Zielmaschinen. Vorteil dabei ist, daß der Programmierer jederzeit auf den Inhalt jedes Registers im Microprozessor Zugriff hat, das Programm also in jedem Stadium genau verfolgen kann.

Die neueren Entwicklungen Psions sind vier Standard-„Business"-Programme für den Sinclair QL, die zum Lieferumfang des Rechners gehören. Die Motorola-68000-Chip-Familie, von denen einer die Zentraleinheit des QL ist, wurde auf Hochsprachenebene entwickelt. C-Programme kompilieren so gut auf diese Chips, daß sich das Schreiben in Maschinensprache erübrigt. Würden alle Heimcomputer dem QL-Vorbild folgen, könnte C die Maschinensprache völlig ersetzen. Softwarehäuser wie Psion könnten dann die Übertragung per Hand auf alle Zeiten vergessen.

Imagepflege

Melbourne House ist für Abenteuerspiele bestens bekannt. Darunter sind solche Hits wie „The Hobbit" und „Mugsy", die sich durch hohen Grafikstandard und gute Geschichten auszeichnen. Doch das Unternehmen produziert auch Computerbücher.

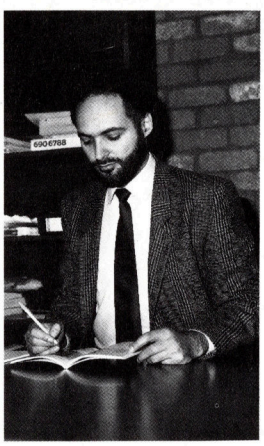
Melbourne House wurde 1977 von dem Australier Alfred Milgrom gegründet. Die Einführung des Sinclair ZX 80 brachte Milgrom auf die Idee, mit dem ertragreichen Verlegen von Heimcomputerbüchern zu beginnen. 1980 veröffentlichte Melbourne House „30 Programme für den ZX 80". Der Erfolg dieses Buches wurde mit einer ganzen Serie von Büchern für den Sinclair-Rechner fortgesetzt.

Im Jahr darauf kam der ZX 81 auf den Markt. Die Nachfrage nach Büchern und Programmen für den ZX 80 ließ daher nach. Die Gesellschaft überlebte diese Phase nur, weil die Verkäufe in den USA trotzdem gut waren. Melbourne House lernte aus dieser Lektion, und man erkannte die Notwendigkeit, sich auch mit anderen Computern zu befassen.

Aufgrund des Erfolges des Sinclair Spectrum produzierte Melbourne House Spielprogramme, die die Grafik- und Soundmöglichkeiten dieses Rechners voll ausnutzten. Das Spielhallenspiel „Penetrator" verkaufte sich ausgezeichnet, doch das erfolgreichste Programm des Unternehmens war „The Hobbit", ein Grafikabenteuer, das auf Tolkiens gleichnamigem Roman beruht. Es wurde mit dem „Goldenen Joystick" als bestes strategisches Spiel des Jahres ausgezeichnet. Die Programmcassette wurde als Paket vermarktet, in dem auch ein Exemplar von Tolkiens Buch war. Dies war eine Auflage der Nachlaßverwalter von Tolkien und hatte zur Folge, daß The Hobbit dreimal so teuer war wie andere für den Spectrum erhältliche Software. Trotz des Preises lief der Verkauf gut. Jetzt ist das Spiel auch für andere Heimcomputer erhältlich.

Die hauseigenen Programmierer der Firma arbeiten in Melbourne, Australien. Jeweils vier Leute befassen sich bei der Programmentwicklung mit einem Spielaspekt. So braucht man für die Fertigstellung viel Zeit, doch der Aufwand scheint durch Erfolg und Umsätze gerechtfertigt. Die Loyalität der Kunden will das Unternehmen nutzen, um so mehr Bücher verkaufen zu können. Paula Byrne, Pressesprecherin des Hauses, sagt dazu: „Wenn Leute ein Buch kaufen wollen, wissen sie nicht, für was sie sich entscheiden sollen. Vielleicht kaufen sie am Ende etwas, das ihnen nicht gefällt. Und das entmutigt sie, weitere Bücher zu kaufen." Dieser Verwirrung will Melbourne House dadurch entgegenwirken, daß man die Bücher kennzeichnet als geeignet für Einsteiger, Fortgeschrittene oder erfahrene Leser. Durch das übereinstimmende Erscheinungsbild von Büchern und Software soll der Konsument die hochwertigen Bücher und die gute Software als gleichwertig ansehen. Ferner ist jedem Exemplar eine Karte beigefügt, mit der Leser aufgefordert werden, ihre Meinung zur Qualität des Produktes mitzuteilen.

Ein neueres Spiel des Hauses, „Mugsy", wird als der „beste interaktive Computer-Comic-Strip der Welt" bezeichnet. Dabei übernimmt der Spieler die Rolle eines Bandenchefs im Chicago der zwanziger Jahre. Die Grafiken sind ausgezeichnet und sehr detailliert. Melbourne Hose arbeitet außerdem an einem Spiel in Hobbit-Art, betitelt „Sherlock Holmes", das jetzt fertiggestellt sein müßte. Es soll ebenso innovativ wie The Hobbit sein. Man benötigte für die Entwicklung fünfzehn Monate. Über den Spielinhalt ist bisher kaum etwas bekannt. Man weiß lediglich, daß gute Kenntnisse „viktorianischen Transportwesens" erforderlich sind, um erfolgreich zu sein.



Alfred Milgrom, Direktor und Verleger, Melbourne House

Philip Mitchell, Autor von The Hobbit und Sherlock Holmes.

Melbourne House hat mit The Hobbit ein witziges und aufregendes Spiel geschaffen. Zum Lieferumfang gehört auch ein Exemplar von Tolkiens Meisterwerk – eine ausgezeichnete Marketingidee.

Ein Teil der Melbourne-Mannschaft

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.

Decrement = Dekrementieren
Dekrementieren heißt Vermindern. In der Datenverarbeitung ist dabei die Verkleinerung eines Zählerinhalts gemeint. In BASIC wird zum Beispiel durch die Anweisung LET A = A- 1 der Index A dekrementiert. Wichtigster Anwendungsbereich sind Schleifen, wobei mit jedem Durchlauf der Schleifenzähler um 1 dekrementiert wird. Bei den meisten BASIC-Versionen sind dafür mehrere Anweisungsformen mit automatischer Indexfortschaltung und Endabfrage vorgesehen, etwa FOR ... NEXT, REPEAT ... UNTIL oder WHILE ... END.

Im Maschinencode gibt es für das Dekrementieren spezielle Befehle, etwa das Kommando DEC, das ein im Adreßteil angegebenes Byte dekrementiert. DEX und DEY können zum Beispiel die Indexregister X und Y dekrementieren. Eine andere wichtige Anwendung ist die indizierte Adressierung, wenn eine ganze Serie von Speicherbytes nach der gleichen Vorschrift verarbeitet werden soll. Das Gegenteil von Dekrementieren, nämlich die schrittweise Erhöhung eines Indexregisters, heißt Inkrementieren.

Degaussing = Entmagnetisierung
Nach längerem Gebrauch wird der Schreib/Lese-Kopf eines Cassettenrecorders allmählich permanentmagnetisch. Dieser Effekt führt zu Signalverzerrungen, die außerordentlich störend werden können. HiFi-Fans begegnen dem durch regelmäßiges Entmagnetisieren.

Wer als Heimcomputer-Benutzer auf den Cassettenrecorder angewiesen ist, sollte die Wartung nicht vernachlässigen. Magnetköpfe und Laufwerk müssen regelmäßig gereinigt werden, um den normalen Bandabrieb zu entfernen, und alle paar Monate sollten auch hier die Köpfe entmagnetisiert werden. Dazu brauchen Sie ein Gerät, das ähnlich aussieht wie eine Taschenlampe. Es enthält eine Spule, deren Eisenkern einseitig aus dem Gehäuse hervorragt. Der Apparat wird ans Netz angeschlossen, an den Magnetkopf gehalten und vor dem Abschalten langsam wieder vom Recorder entfernt. Dadurch wird der Kopf einem magnetischen 50-Hz-Wechselfeld mit abnehmender Amplitude ausgesetzt, womit die Dauermagnetisierung auf einfachste Weise gelöscht wird.

Delimiter = Trennzeichen
Trennzeichen markieren Anfang und Ende von Datenblöcken im RAM, können aber auch Datensätze bei Diskettendateien abgrenzen.

Der Gebrauch von Trennzeichen läßt sich anhand der Speicherung eines BASIC-Programms erläutern. In den beiden ersten Bytes jeder Zeile steht die Zeilennummer als 16-Bit-Integer. Die beiden nächsten Bytes enthalten die RAM-Adresse, unter der die nächste Zeile steht – das beschleunigt das Aufsuchen von Zeilennummern bei GOTO- oder GOSUB-Anweisungen. Danach steht der Zeilentext, gefolgt von einem Trennzeichen, meist in Form eines Nullbyte, zur Markierung des Zeilenendes. Das Trennzeichen darf dabei keinesfalls als Teil der Information zu mißverstehen sein.

Diagnostic Routine = Diagnoseprogramm
Die Vorstellung, daß ein Rechner Störungen von sich aus mitteilt, hat einigen Reiz, obwohl damit vielen Zukunftsromanen der Boden entzogen wäre. So etwas gibt es tatsächlich schon in Form von Diagnoseprogrammen bei einigen Großrechenanlagen und (leider nur wenigen) Microcomputern.

Kein System kann alle Fehler anzeigen. Wenn etwa das Netzteil ausfällt, rührt sich auch das Diagnoseprogramm nicht mehr. Dagegen kann es durchaus die Funktion von Speicher- und Schnittstellenbausteinen oder ganzen Logikplatinen testen. Bei den meisten Micros ist eine einfache RAM-Prüfung vorgesehen, bei der jedes Byte einmal beschrieben und wieder gelesen wird. Dieses Verfahren dient nicht nur zum Erkennen von Speicherfehlern.

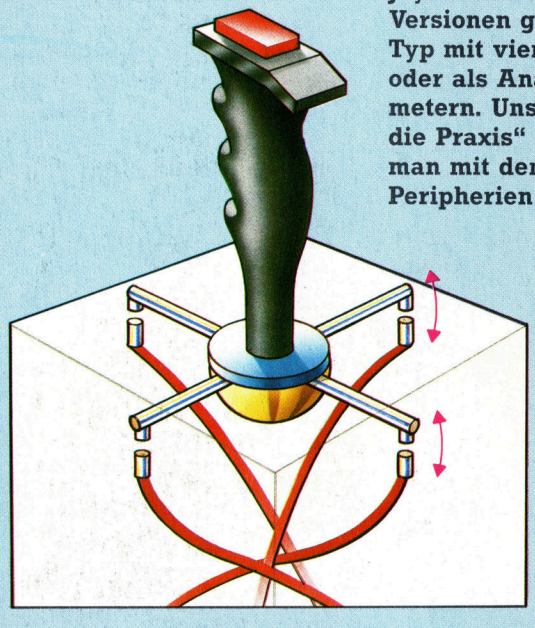
Eines der besten Diagnose-Systeme im Microcomputer-Bereich bietet der „Rainbow" von Digital Equipment. Jede Platine kann softwaremäßig vollständig getestet werden, und bei Fehlermeldungen wird die Lage des defekten Teils auf dem Bildschirm angezeigt.

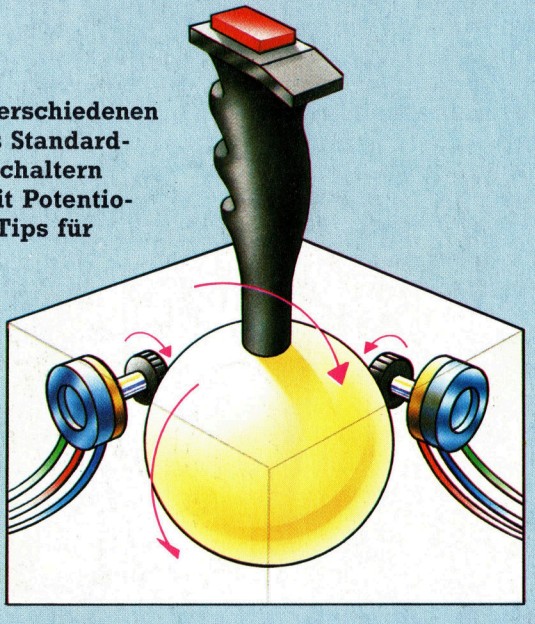
Beim NewBrain-Rechner erscheinen auf dem einzeiligen Display geheimnisvolle Hieroglyphen, während das Diagnoseprogramm läuft.

Digital = Digital
Bei einem digitalen Signal kann der Spannungspegel nur „diskrete", das heißt, abgestufte, vorgegebene Werte annehmen – bei Digitalrechnern genau zwei. Im Gegensatz dazu ist ein Analogsignal „stetig" veränderbar und kann beliebige Zwischenwerte annehmen.

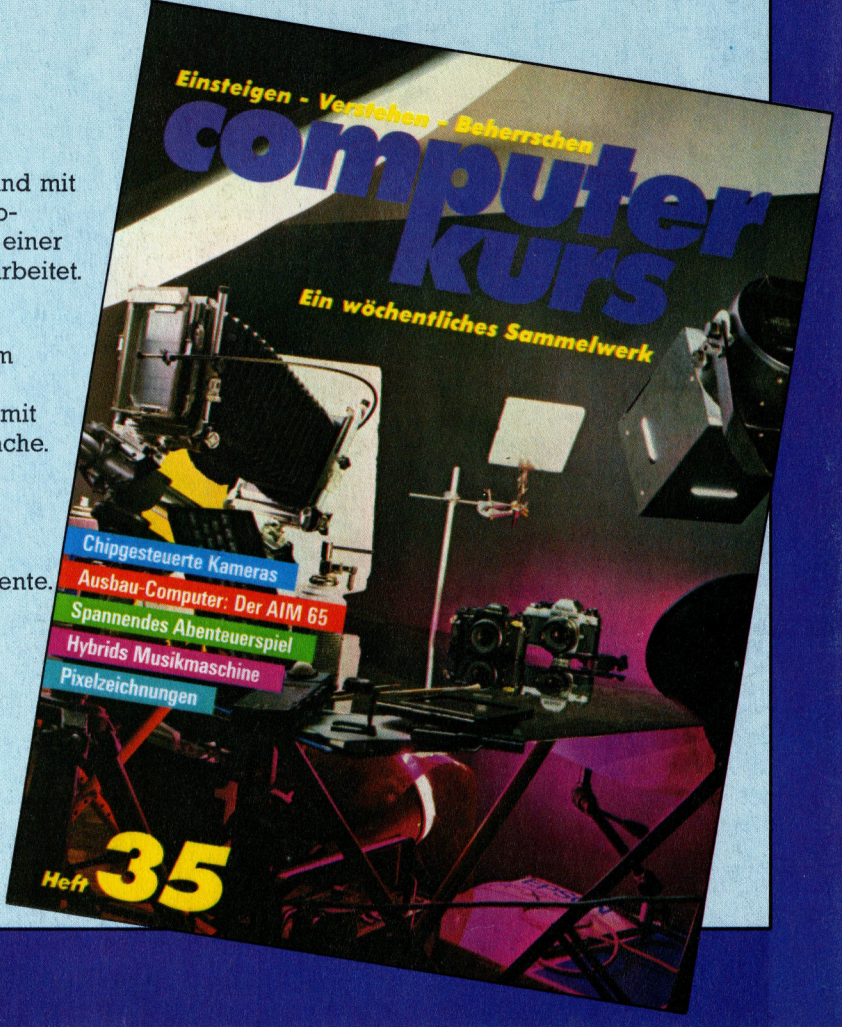
Bildnachweise
925: Steve Cross
927: Liz Heaney
928, 946, 947: Liz Dixon
932, 944, 945, 952, U3: Ian McKinnell
933, 936, 949: Kevin Jones
937, 938, 939: Chris Stevens
950: Bob Bromide, Tony Sleep
951: Tony Sleep

computer kurs Heft 35





Joysticks können in zwei verschiedenen Versionen gebaut sein: Als Standard-Typ mit vier integrierten Schaltern oder als Analog-Modell mit Potentiometern. Unser Beitrag in „Tips für die Praxis" erklärt, wie man mit dem Joystick Peripherien steuert.



Kameras mit Prozessor

Fotoapparate der neuesten Generation sind mit Microprozessoren bestückt, die dem Fotografen Arbeit abnehmen sollen. Anhand einer Kamera wird erklärt, wie die Elektronik arbeitet.

Die Musik-Maschine

Ungeahnte Möglichkeiten, Musik mit dem Heimcomputer zu erzeugen, bietet das System „Music 500": eine Klaviertastatur mit 49 Tönen und eine eigene Programmsprache.

Platinencomputer

Aus einzelnen Modulen besteht der AIM 65 und ist so beliebig erweiterbar. Ein besonders geeignetes Gerät für Experimente.

Mehrfach-Auswahl

In dieser Folge unseres PASCAL-Kurses erklären wir die Vergleichsstrukturen wie IF und CASE.

Pixelzeichnungen

Mit Hilfe der Assemblersprache lassen sich die Prozessoren 6502 und Z80 für grafische Anwendungen einsetzen.