Einsteigen - Verstehen - Beherrschen

DM 3,80 öS 30 sfr 3,80

computer kurs

Heft 33

Ein wöchentliches Sammelwerk

Peripherie-Überblick

Marketing für Software

Spectrum-Nachfolger

Pacman-Varianten für Heimcomputer

Motoren per Rechner steuern

computer kurs

Heft 33

Inhalt

WIE SIE JEDE WOCHE IHR HEFT BEKOMMEN

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

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

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

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

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

Abonnement

Sie können Computer Kurs auch alle 2 Wochen (je 2 Ausgaben) per Post zum gleichen Preis im Abonnement beziehen. Der Abopreis für 12 Ausgaben beträgt DM 45,60 inkl. Mwst., den wir Ihnen nach Eingang der Bestellung berechnen. Bitte senden Sie Ihre Bestellung an: Marshall Cavendish Int. Ltd. (MCI), Sammelwerk Service, Postgiroamt Hamburg 86853-201, Postfach 105703, 2000 Hamburg 1, Kennwort: Abo Computer Kurs. Bitte geben Sie an, ab welcher Nummer das Abo beginnen soll und ob Sie regelmäßig für jeweils 12 Folgen einen Sammelordner wünschen. 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 füllen Sie eine Zahlkarte aus an: Marshall Cavendish International Ltd. (MCI), Sammelwerk-Service, Postgiroamt Hamburg 48064-202, Postfach 105703, 2000 Hamburg 1, Kennwort: Sammelordner Computer Kurs.

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

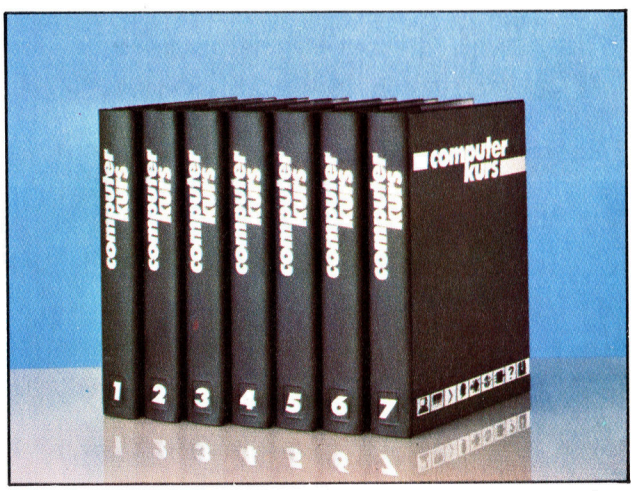
Schweiz: Der Sammelordner kostet sfr 15. Bitte wenden Sie sich an Ihren Kiosk; dort werden Sie jederzeit die gewünschten Exemplare erhalten.

INHALTSVERZEICHNIS

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

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

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



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

Das gewisse Etwas

Erfolg und Nichterfolg eines Produkts hängen weniger von der Herstellung als von der Vermarktung ab. Das gilt auch für den Computermarkt. Ob man nun Spielsoftware verkauft oder einen neuen Geschäftscomputer: Enscheidend ist das richtige Produkt-Image.

Diese beiden Diskettenprogramme kosten jeweils 160 Mark. Das eine ist aufwendig verpackt, erweckt Aufmerksamkeit und vermittelt „Wertigkeit", wogegen das andere in einer vergleichsweise simplen Schachtel zu einem scheinbar zu hohen Preis angeboten wird.

Marketing ist das Geschäft, eine Brücke zwischen dem Produkt, das der Anbieter verkaufen will, und dem Geld, das der potentielle Käufer in der Tasche hat, zu schlagen. Die Vermarkter müssen über die Bedürfnisse, Wünsche und finanziellen Möglichkeiten ihrer Käuferschicht Bescheid wissen (was zuweilen mit Marktforschung ergründet wird) und sollten erst daraufhin investieren.

Marketingentscheidungen sind bereits dann zu fällen, wenn sich die Produktion eines neuen Geräts noch im Planungsstadium befindet. Die Funktionen des Rechners, die Anzahl der zu produzierenden Einheiten und die Herstellungskosten pro Einheit sind die marketingbestimmenden Faktoren.

Ähnliche Überlegungen stehen beim Vermarkten von Software an. Was nützt es, große Summen für die Entwicklung und Herstellung eines Spiels auszugeben, wenn potentielle Käufer den Preis nicht zahlen können? Die Kosten können so nicht erwirtschaftet werden. Umgekehrt wird ein Softwarehaus, das in die Entwicklung seiner Produkte wenig investiert, sie zu dürftig ausstattet oder sogar fehlerhaft ausliefert, große Probleme haben. Der Markenname an sich könnte Schaden nehmen, und damit wären künftige Produkte von vornherein gefährdet. Wird viel Geld für die Entwicklung, aber wenig für die Vermarktung ausgegeben, bleibt man auf einem hervorragenden Produkt sitzen, von dem eigentlich niemand etwas weiß.

Das Image festlegen

Ebenso wichtig ist es, in einem sehr frühen Stadium festzulegen, wo das Produkt anzusiedeln ist. Dazu findet eine Konkurrenzanalyse statt. In diesem Zusammenhang wird das „Image" des Produktes festgelegt. Die Hersteller von Zigaretten und Seifen haben ein gemeinsames Problem: Verkauft man die „Wahrheit", ist ein Produkt wie das andere. Nun sind Computerhersteller zwar nicht ganz in dieser Situation, doch ist es nicht immer leicht, die Be-

Die recht gelungene Gestaltung eines wirksamen Produktimages vermittelt unser Beispiel: Commodores Elefant suggeriert das „Jumbo"-Gedächtnis des Commodore 64. Die Szenen, in denen der Acorn Electron zu sehen ist – zu Hause und in der Schule – vermitteln Vielseitigkeit, Anwenderfreundlichkeit und erzieherischen Wert. Das eher unwirkliche Apple-Umfeld mit dem Orwellschen Bezug spricht für Freiheit und Individualität, es stellt eine humanisierte Computerwelt dar. Doch aus der Visualisierung können sich unerwünschte Negativwirkungen ergeben: Es ist ja sprichwörtlich, daß ein Elefant vor einer „Maus" erschrickt (anders als der Macintosh oder die Lisa), er könnte folglich für ein überholtes Produkt stehen. Potentielle Electron-Käufer könnten das Umfeld Familie für stereotyp und langweilig halten oder lehnen gar die Verbindung zur Schule ab. Apple-Käufer könnten sich daran stören, als hirnlose Massenmenschen dargestellt zu werden.

sonderheiten eines bestimmten Rechners einem Käufer zu erklären, der möglicherweise die entscheidenden technischen Vorteile gar nicht versteht. Ähnlich ist es bei Spiel- und Anwendungsprogrammen, deren Güte häufig erst nach dem Kauf bei der Nutzung deutlich wird.

Das „industrielle Design" einer Maschine, das äußere Erscheinungsbild, die Anordnung von Schaltern und Tasten ist oft entscheidende Ausgangsbasis für die Entwicklung eines Produkts bzw. seines Images. So ist beispielsweise der Acorn B unkompliziert und sieht „nützlich" aus, womit der erzieherische Wert des Gerätes auch optisch vermittelt wird. Die Gestaltung anderer Rechner erfolgte unter anderen Gesichtspunkten. So fand man für die Computer der Atari-XL-Reihe eine rauhe, asymmetrische Form mit betonten Lüftungsschlitzen – im Zuge der Versuche, dem Produkt eine neue Marktposition zu geben und es besser zu verkaufen. Commodore verpflichtete Ferdinand Porsche, den Designer der berühmten Autokarosserie, zur Neugestaltung seiner Rechner. Die Firma wollte ein rundes, vertrauenerweckendes Produkt-Design ohne futuristische Gestaltungselemente.

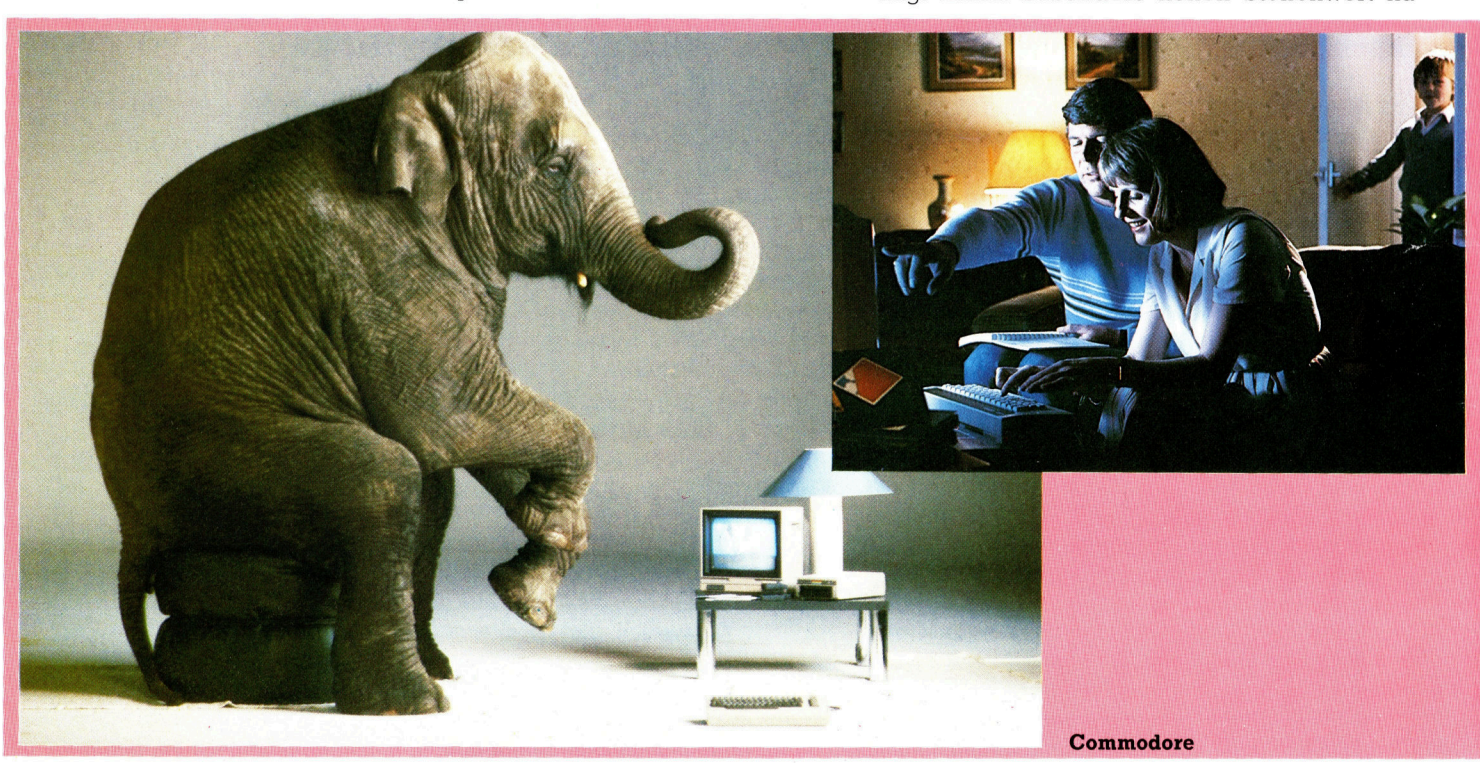
Das Produktprofil

Doch das „Produktprofil" ist nicht allein eine Frage der Gestaltung. Das „Image" muß darüberhinaus mit einer entsprechenden Werbekampagne verdeutlicht und intensiviert werden. Das „Elefantengedächtnis" des Commodore 64 beispielsweise wurde durch einen Elefanten in der Fernseh- und Zeitschriftenwerbung visualisiert.

Natürlich spielt auch der Produktname eine Rolle. Zu Beginn der Entwicklung des Computermarktes mußten die Hersteller gegen verbreitete Vorurteile angehen, die durch Filme der sechziger und frühen siebziger Jahre geschaffen worden waren. Der Computer wurde immer als der unmenschliche „Große Bruder" angesehen, der außerhalb der Kontrolle des Individuums gestellt war. Folglich gab man den Rechnern Namen, die Vertrauen weckten und auf technische Anmutung ganz verzichteten, so etwa PET (Liebling) und Apple.

Der Macintosh wurde in England und den USA mit einer „Teaser"-Fernsehkampagne eingeführt, die Ridley Scott in England gefilmt hatte. Man zeigte einen Riesenbildschirm mit dem „Großen Bruder", der von einer jungen Frau zerschlagen wurde. So versuchte man die neu gefundene Freiheit zu vermitteln, die Apples jüngstes Produkt gab. Der Slogan lautete „Dank Macintosh wird 1984 nicht 1984 sein". Auf dem amerikanischen Markt ist „Mackintosh" eine bekannte Apfelsorte. Das Entwicklungsteam hatte offensichtlich Rechtschreibprobleme und ließ das „k" weg. Der Fehler blieb im Produktnamen.

Nachdem das breite Publikum darauf eingestimmt war, daß „man" einen Computer braucht, und sich an den Gedanken mit dem Computer zu leben gewöhnt hatte, kam die alte „High Tech"-Tradition wieder zum Tragen. Der Erfolg der Produktbezeichnungen in dieser Kategorie basiert darauf, daß man Namen findet, die so weit wie irgend möglich vom üblichen Vokabular entfernt sind. Das heißt: Statt Namen zu benutzen, die man in jedem Wörterbuch findet, gibt man Acronymen oder Buchstabenfolgen ohne jede Bedeutung den Vorzug. Einen besonders hohen Stellenwert ha-



Commodore

ben hier die seltenen Buchstaben (die auch beim Scrabble viele Punkte bringen!). Daher kommen Rechnerbezeichnungen wie ZX 81, MTX 500 und MZ-700.

Die Verpackung, bei der Hardware mehr zum Schutz entwickelt, hat für das Image der Software eine besondere Bedeutung. Es gibt zwei Philosophien für die Verpackungsgestaltung: Entweder man beschränkt sich auf eine Minimalverpackung (Cassettenhülle mit farbigem Einlegeblatt) zu entsprechend vorteilhaftem Preis, oder aber man gestaltet die Verpakkung „wertig", liefert das Produkt in einer großen Hülle, die zuweilen Ähnlichkeit mit einem Buch oder einer Videocassettenhülle hat. Beigepackt werden Extras, die nicht unbedingt Bestandteil des Programms sein müssen.

Die Produkt-Promotion

Das Image wird durch Werbung vermittelt. Allgemein gesagt ist die Verwandtschaft zwischen Marketing und Werbung dieselbe wie die zwischen Strategie und Taktik. Die Fragen, wie geworben wird und wie groß der Werbeetat ist, sind eindeutig Marketingentscheidungen. Erfahrungsgemäß führen Heimcomputerhersteller neue Produkte mit Vorliebe kurz vor Weihnachten ein und investieren zu dieser Zeit sehr viel Geld in die Werbung.

Wenn aber das Image nicht stimmt, hilft die Produkt-Promotion wenig. In der Werbebranche gibt es dafür ein nahezu klassisches Beispiel aus dem Zigarettenmarkt. Man wollte den Namen „Strand" bekannt machen und versuchte das mit Filmen, die in Kinos und im Fernsehen gezeigt wurden. Zu sehen war nur ein Mann im weißen Regenmantel. Textgrundlage war die Aussage „Mit Strand sind Sie nie allein". Die Werbebotschaft aber, die verstanden wurde, war „Einsame Menschen rauchen Strand". Die Marke ließ sich nicht verkaufen.

Das Planen und Verwirklichen eines Produktprofils mag man als „kreative" Seite des Marketings bezeichnen (was viele Werber auch tun). Doch der Vertrieb spielt eine mindestens ebenso wichtige Rolle, da er im Zweifelsfall das schwächste Glied der Kette sein könnte. Es ist eine Sache, die Phantasie des Verbrauchers mit einem neuen Produkt anzuregen. Es ist aber eine andere Sache, die Ware tatsächlich in den Haushalt des Verbrauchers oder ins Büro zu bringen.

Soll das Produkt nur durch den Versandhandel zu beziehen sein? Der Weg ist kostengünstig, aber woher soll der Verbraucher wissen, daß er bei Bedarf auch Kundendienstleistungen bekommt? Man muß möglicherweise zu sehr niedrigen Preisen verkaufen, um den Ladenverkauf auszuschalten. Doch wenn die Gewinnspanne zu gering kalkuliert ist, kann es Probleme bei der Finanzierung der Produktion größerer Stückzahlen geben, was wiederum Lieferverzögerungen zur Folge hat.

Der Verkauf durch Einzelhandelsketten vermittelt dem Verbraucher Markenbewußtsein. Dafür muß aber bezahlt werden, weil den Ketten große Rabatte einzuräumen sind, die den Ertrag wiederum schmälern.

Beim Kampf ums Überleben in der Computerbranche haben die Pleiten so mancher ausgezeichneter Hard- und Softwarehersteller deutlich gemacht, daß die Produktion einer guten Ware nur die Hälfte des Erfolges ist. Märkte schaffen und erhalten bleibt wichtig – denn hier allein wird Umsatz gemacht.

Bedingungs-abfragen

Die Logikfunktionen AND und OR sind auch in der Software von Bedeutung: Die meisten BASIC-Dialekte und Maschinensprachen enthalten AND und OR als wichtige Teile ihres Befehlssatzes.

In Maschinensprache und auch im BASIC gibt es eine Vielzahl von Anwendungsmöglichkeiten für Logik-Befehle. Meist sollen zwei oder mehrere Behauptungen zueinander in Beziehung gesetzt werden. Welches Ergebnis würden Sie beim folgenden BASIC-Programm vorhersagen?

```
10 FOR I = 1 TO 5
20 FOR J = 1 TO 5
30 IF I = 3 AND J = 2 THEN PRINT I,J
40 NEXT J
50 NEXT I
60 END
```

Das Programm durchläuft die beiden verknüpften Schleifen, druckt die Werte für I und J aber nur aus, wenn I = 3 und J = 2 ist. Auf dem Bildschirm zeigt sich dieses Resultat:

```
3          2
```

OR kann ähnlich eingesetzt werden. Ändern Sie Zeile 30 folgendermaßen:

```
30 IF I=3 AND J=2 OR J=4 THEN PRINT I,J
```

Das neue Programm ergibt diese Werte:

```
1          4
2          4
3          2
3          4
4          4
5          4
```

Der Computer führt die AND-Funktion vor der OR-Funktion aus. I und J werden ausgegeben, falls I=3 und J=2 sind oder wenn J=4 ist. Die Rechenhierarchie kann durch den Gebrauch von Klammern geändert werden. Welches Ergebnis erscheint bei einer neuen Zeile 30?

```
30 IF I=3 AND (J=2 OR J=4) THEN PRINT I,J
```

Um die Rechnerfunktionen zu steuern, werden in vielen Heimcomputern spezielle Register benutzt. Jedes Bit eines solchen Registers kann dabei eine eigene Aufgabe haben. So hat etwa der Commodore 64 ein Acht-Bit-Register zum Ein- und Ausschalten der Sprites. Dabei ist ein Bit für jedes der acht Sprites zuständig. Wenn ein Bit des Registers auf Eins gesetzt wird, ist das dazugehörige Sprite auf dem Bildschirm sichtbar, während es bei einer Null ausgeschaltet bleibt. Mit BASIC-Befehlen kann durch POKEn der entsprechenden achtstelligen Binärzahl in das Register jede beliebige Sprite-Kombination verwirklicht werden. Allerdings berücksichtigt diese Methode nicht den Zustand des Registers vor dem POKE-Befehl – ein Sprite, das vorher eingeschaltet war, verschwindet möglicherweise vom Bildschirm. Die Lösung des Problems besteht darin, das gewünschte Bit zu isolieren und es zu verändern, ohne dabei die anderen Bits zu beeinflussen.

Um diese Technik einmal vorzuführen, nehmen wir an, daß die Sprites 0, 1, 5 und 6 eingeschaltet sind. Die Werte im Spriteregister sehen dann so aus:

Spritenummer	7	6	5	4	3	2	1	0
Ein/Aus-Bit	0	1	1	0	0	0	1	1

Sprite 4 wird nun eingeschaltet, indem mit PEEK der Registerinhalt abgerufen und durch OR mit 16 (binär 00010000) verknüpft und das Ergebnis wieder ins Register gePOKEt wird.

Ursprüngliches Byte	0	1	1	0	0	0	1	1
Verknüpft (OR) mit	0	0	0	1	0	0	0	0
Ergibt neues Byte	0	1	1	1	0	0	1	1

Mit dem BASIC-Befehl POKE reg, PEEK(reg)OR16 läßt sich Bit 4 im Register einschalten. Zum Ausschalten wird der Registerinhalt mit PEEK abgerufen und durch AND mit 239 verknüpft:

Neues Byte	0	1	1	1	0	0	1	1
AND	1	1	1	0	1	1	1	1
Ursprüngliches Byte	0	1	1	0	0	0	1	1

Das Ergebnis (239) läßt sich durch Subtrahieren der 16 von der Zahl 255 berechnen. Mit POKE reg, PEEK(reg)AND239 ist das Register jetzt wieder im alten Zustand.

Besonders häufig kommt die beschriebene Technik in Maschinenspracheprogrammen vor – die Veränderung von Kontrollregistern stellt oft den Kernpunkt solcher Programme dar.

Lösungen zu Übung 3

1a) $A.(\overline{A}+\overline{B})$
$=A.\overline{A}+A.\overline{B}$ (Distributivgesetz)
$=A.\overline{B}$ ($A.\overline{A}=0$)

b) $X+Y.(X+Y)+X.(\overline{X}+Y)$
$=X+Y+X.(\overline{X}+Y)$ (Beziehung 5)
$=X+Y+X.Y$ (Beziehung 6)
$=X+Y$ (Absorption)

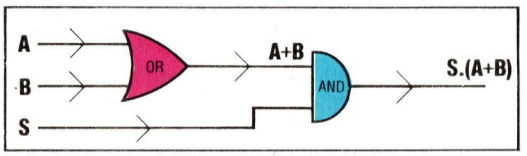
c) $P.Q+\overline{P}.Q+\overline{P}.\overline{Q}$
$=P.Q+\overline{P}.(Q+\overline{Q})$ (Distributivgesetz)
$=P.Q+\overline{P}$ ($Q+\overline{Q}=1$)
$=\overline{P}+Q$ (Gegenstück zur Beziehung 6)

d) $\overline{X.Y.Z.Z.Y}$
$=\overline{X}.\overline{Y}.\overline{Z}.\overline{Z}.\overline{Y}$ (de Morgan)
$=X.Y.Z.(Z+\overline{Y})$ ($\overline{X}=X$, de Morgan)
$=X.Y.Z.Z+X.Y.Z.\overline{Y}$ (Distributivgesetz)
$=X.Y.Z+0$ ($Z.Z=Z$, $Y.\overline{Y}=0$)
$=X.Y.Z$

2) Wertetabelle der Alarmanlage:

Eingänge			Ausgang
A	B	S	Alarm
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Alarm $=\overline{A}.B.S+A.\overline{B}.S+A.B.S$
$=\overline{A}.B.S+A.S.(\overline{B}+B)$ (Distributivgesetz)
$=\overline{A}.B.S+A.S$ ($\overline{B}+B=1$)
$=S.(A+\overline{A}.B)$ (Distributivgesetz)
$=S.(A+B)$ (Gegenstück zur Beziehung 6)



3) Wenn die drei Schalter mit X,Y und Z sowie die Lampe als P bezeichnet werden, ergibt sich diese Wertetabelle:

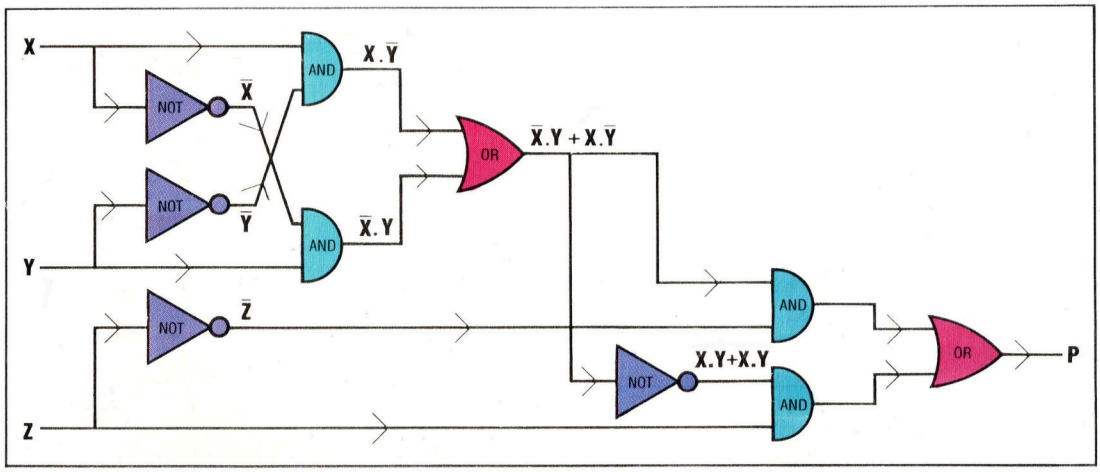
Eingänge			Ausgang
X	Y	Z	P
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

$P=\overline{X}.\overline{Y}.Z+\overline{X}.Y.\overline{Z}+X.\overline{Y}.\overline{Z}+X.Y.Z$
$=Z.(\overline{X}.\overline{Y}+X.Y)+\overline{Z}.(\overline{X}.Y+X.\overline{Y})$ (Distributivgesetz)
$=Z.(\overline{\overline{X}.\overline{Y}+X.Y})+\overline{Z}.(\overline{X}.Y+X.\overline{Y})$ (de Morgan)

4) Die Wahrheitstafel zeigt, daß man mit der Frage „Sagst Du die Wahrheit?“ nicht weit kommt – in beiden Fällen würde die Antwort gleich lauten. Die Tabelle entspricht in ihrer Form der Funktion $X.Y+\overline{X}.Y$, verkürzt als Y. Das bedeutet, die Antwort hängt nur von einer Variablen ab und kann deshalb nicht dabei helfen, zwischen den Personen zu unterscheiden. Bei der Frage „Haben Schweine Flügel?“ sieht die Tabelle zum Glück schon anders aus:

		Mögliche Antwort	
		JA	NEIN
Was tut der Gefragte möglicherweise	lügt	1	0
	sagt die Wahrheit	0	1

Damit haben wir die Wahrheitstabelle der Funktion $X.\overline{Y}+\overline{X}.Y$, die der Exklusiv-OR-Funktion entspricht – damit läßt sich der Lügner ohne weiteres identifizieren.



In neuem Gewand

Der Sinclair Spectrum ist einer der erfolgreichsten Heimcomputer, im Vergleich mit neuen Geräten wie dem Schneider CPC 464 oder dem Commodore 128 macht er inzwischen jedoch einen recht biederen Eindruck. Sinclair gab der alten Maschine daher ein neues Gesicht.

Der Sinclair Spectrum bot bei seiner Vorstellung im Jahre 1982 erstaunliche Fähigkeiten für wenig Geld. Die einzigen ernstzunehmenden Konkurrenten waren damals der VC 20 mit mageren 3,5 KByte Arbeitsspeicher und der Texas TI99/4A, der etwa doppelt soviel kostete. Dank seines akzeptablen Preises war der Spectrum im Handumdrehn ein Renner, der nicht nur bei Erstkäufern gut ankam, sondern auch bei Computerfans, die ihrem ZX 80 oder ZX 81 entwachsen waren. Die Maschine bot einen Arbeitsspeicher von 48 KByte, ein gutes BASIC, konnte acht Farben auf den Schirm bringen und verfügte über eine einfache Tonerzeugung. Auch die Tastatur war eine große Verbesserung gegenüber den flachen Folientasten des ZX 81.

Angst vor der Konkurrenz

In den zweieinhalb Jahren seit der Vorstellung des Spectrum brachte die Konkurrenz eine ganze Reihe von Geräten auf den Markt, die die beherrschende Stellung des Spectrum angreifen sollten. Dabei war der Commodore 64 trotz seines deutlich schwächeren BASICs der erfolgreichste Rivale. Er verfügte über einen größeren Speicher (der allerdings nur mit Maschinencode voll ausgenutzt werden konnte), ausgezeichnete Möglichkeiten der Tonerzeugung und über eine „echte" Schreibmaschinentastatur.

Zu diesem Zeitpunkt stellte sich heraus, daß die Tastatur des Spectrum – früher eine interessante Alternative – ein Nachteil des Gerätes war. Wenn auch der Spectrum durch viele „professionelle" Programmpakete eine breite Softwarebasis hatte, vermittelte die Tastatur den Eindruck, als hätte man bei der Bedienung Handschuhe an. Viele Anwender investierten daher zusätzlich in eine gute Tastatur. Als das Interface 1/Microdrive angeboten wurde, verstärkte sich dieser Trend weiter. 1984 zeigte sich, daß die Anwender Sinclairs Vorstellung über eine brauchbare Tastatur nun nicht mehr akzeptierten.

Sinclair Research gab dem Spectrum daraufhin ein neues Gewand. Der Spectrum+ unterschied sich in den technischen Daten kaum von der ursprünglichen Maschine, wurde jedoch mit einer QL-ähnlichen Tastatur ausgestattet, mit Zusatztasten, einem Reset-Schalter und versenkbaren Füßen. Alle Peripheriegeräte, die mit dem alten Spectrum liefen, funktionierten auch mit dem neuen Gerät. Sinclair versäumte es jedoch, den Spectrum durch eine Verbesserung der Klangmöglichkeiten, eine zusätzliche Monitorbuchse oder durch Integration des Interface 1 wirklich entscheidend auf Vordermann zu bringen. Die primitive Tonerzeugung erweist sich heute dabei als das größte Handicap der Maschine. Durch die zwei ausziehbaren Beine kann der Sound jetzt zwar besser durch die Bodenplatte ans Ohr des Spielers dringen, aber dadurch werden auch die Lade- und Speichergeräusche doch ziemlich störend verstärkt.

Sinnvolle Zusatztasten

Der Spectrum+ hat die Abmessungen 319x149x38 mm. Zusatztasten gestalten die Programmierung einfach: Es gibt Tasten für den Aufruf des Grafikmodus, normale und inverse Darstellung, Löschen (Delete), Unterbrechung (Break) und weitere Tasten für oft verwandte Satzzeichen wie Semikolon, Anführungsstriche, Komma und Punkt. Auch an eine Umschaltung auf Symboldarstellung wurde gedacht, und die Cursortasten rückten neben die Leertaste. Obwohl die alten Tastenkombinationen nicht verändert wurden, können Besitzer

Trendgerecht wurde in den Lieferumfang des Spectrum+ ein Softwarepaket mit sechs Programmen eingeschlossen. Der „Sechserpack" besteht aus einer Textverarbeitung, einem Kalkulationssystem, zwei Spielen und zwei Grafikpaketen. Die Software hat einen hohen Qualitätsstandard.

Der kleine Unterschied

Im Inneren des attraktiven Gehäuses im QL-Stil befindet sich die Platine mit der Versionsnummer 4.5. Sie ist fast identisch mit der Platinen-Version 3, die im August 1983 herausgegeben wurde. Die zwei freien Kabel verbinden den Reset-Knopf mit der Platine.

der älteren Maschine mit dem neuen Modell Probleme haben. Da die neue Edit-Taste direkt neben dem "A" liegt, kann ein Fehlgriff beim Editieren einer Programmzeile zum Löschen derselben führen.

Sechs Programme im Zubehör

Um den Spectrum+ attraktiver zu gestalten, wird ein „Sechserpack" von Programmen mitgeliefert – Psion Chess, Make-a-Chip, Scrabble, Chequered Flag, Vu-3D und das ausgezeichnete Textsystem Tasword Two. Alle diese Programme haben einen hohen Standard. Das außerdem mitgelieferte Handbuch informiert den Anwender ausführlich über das System, die Zusammenarbeit der einzelnen Bauteile sowie den Speicheraufbau.

Der Spectrum+ eignet sich für Spiele besser als das alte Gerät. Spielefans werden jedoch in jedem Fall Joysticks und Schnittstelle kaufen müssen. Die von Kempston und Fuller angebotenen Schnittstellen lassen sich auch an den Spectrum+ anschließen. Allerdings arbeitet die Software beim Einsatz der Fuller-Soundbox, wie auch auf der alten Maschine, nicht einwandfrei. Das Centronics-Interface von Kempston funktioniert ebenso wie das Wafadrive.

Der Spectrum+ ist eine spürbare Verbesserung gegenüber der alten Version. Obwohl die Aufwertung des Spectrum mit der Technik des QL ein cleverer Schritt war, schneidet die Tastatur des Spectrum+ im Vergleich zu denen anderer Hersteller schlecht ab. Die Tasten sind schwergängig und liegen immer noch zu dicht beieinander.

Als Einstiegsgerät kann der Spectrum+ jedoch durchaus in die engere Wahl gezogen werden. Allerdings liegt die Vermutung nahe, daß Sinclair den Spectrum+ nur herausgebracht hat, um den Preis zu erhöhen – wobei es kaum erstaunlich wäre, wenn der Vorgänger bald vom Markt verschwinden würde. Genauer besehen ist die Einführung dieses Modells eine halbherzige Geste. Sinclair hätte besser daran getan, den Preis der alten Maschine herabzusetzen (und auch den des Interface 1/Microdrive). Die Firma hätte den Preis auch geringfügig erhöhen können, wenn sie den Spectrum mit allem wirklich Notwendigen ausgestattet hätte, wie einer vernünftigen Tonerzeugung, Schreibmaschinentastatur, Monitoranschluß und möglicherweise sogar einem Microdrive.

Spectrum+

ABMESSUNGEN

319x149x38 mm

SPEICHER UND SCHNITTSTELLEN

Die gleiche Ausstattung wie 48K-Spectrum, eingebautes BASIC, volle Kompatibilität der Software.

TASTATUR

58 geformte Tasten (darunter eine separate Leertaste); Membrantastatur.

HANDBÜCHER

Gut ausgestattetes Handbuch mit Cassette.

STÄRKEN

Durch die Bandbreite der Spectrum-Software, Anwenderclubs und Spezialveröffentlichungen ist das Gerät sehr attraktiv.

SCHWÄCHEN

Trotz der neuen Tasten bleibt die schwergängige Tastatur ein wesentlicher Schwachpunkt des Gerätes.

Tastaturbelegung

Ob Sie die QL-ähnliche Tastatur des Spectrum+ für eine Verbesserung halten, hängt von Ihrem Eingabestil ab. Die zusätzlichen Funktionstasten sind jedenfalls ein echter Vorteil. Mit den Tastenfolgen der älteren Geräteversion (zum Beispiel [SYM SHIFT] + [0]) lassen sich die Funktionen der neuen Tasten ebenfalls aufrufen.

Definitionen

In diesem Teil unseres Grafik-Kurses führen wir in die Technik anwenderdefinierter Grafik auf dem Commodore 64 ein und fahren mit der Entwicklung des U-Boot-Jagdspieles fort.

Der Vorgang, einen eigenen Zeichensatz auf dem Commodore 64 zu erstellen, ist nicht gerade problemlos: Da es keine speziellen Befehle im Commodore-BASIC gibt, muß der Zugriff auf den Speicher mit den Befehlen PEEK und POKE durchgeführt werden.

Der Zeichensatz des Commodore 64 besteht aus einem ROM-Bereich, der bei Speicherstelle 53248 beginnt. Jedes Zeichen erscheint auf dem Bildschirm als ein Muster in einer Matrix aus acht mal acht Punkten. Um ein Muster von 64 Punkten darzustellen, braucht man 64 Bits bzw. acht Bytes. Die acht Bytes von Adresse 53248 bis 53255 repräsentieren das „@"-Zeichen. Es hat den Bildschirm-Code 0. Das bedeutet, daß dieses Zeichen auf dem Bildschirm erscheint, wenn man den Wert Null in eine der Adressen des Bildschirmspeichers POKEt. Die nächsten acht Bytes, von 53256 bis 53263, repräsentieren den Buchstaben „A" (Bildschirm-Code 1) und so weiter.

Der ROM-Zeichensatz teilt seinen Adressen-Bereich des Speichers mit den Ein-/Ausgabe-Geräten. Die 6510A-CPU kann nun so umprogrammiert werden, daß sie diesen Bereich als Position für den Zeichensatz akzeptiert. Dies mag merkwürdig erscheinen, doch normalerweise ist die CPU nicht dafür verantwortlich, Zeichendefinitionen aus dem ROM auf den Bildschirm zu übertragen. Diese Aufgabe ist einem untergeordneten Chip zugeteilt, der unter der Kontrolle der CPU steht. Der Inhalt von Speicherstelle 1 bestimmt den Status der E/A-Operationen, und Bit 2 dieser Adresse agiert als Schalter, um zu bestimmen, in welcher Art die CPU das Zeichensatz-ROM behandelt. Wird dieses Bit auf Null gesetzt, belegt die CPU diesen Bereich für die E/A-Geräte. Die anderen Bits der Speicherstelle 1 haben ähnliche spezielle Funktionen in bezug auf die Kontrolle des Systems, so daß man sehr vorsichtig sein muß, wenn man den Wert von Bit 2 ändert. Am besten verwendet man hierzu die logischen Operatoren AND und OR.

Stellen Sie sich vor, der Inhalt der Adresse 1 sähe wie folgt aus:

Bit	7	6	5	4	3	2	1	0
	0	1	1	0	1	0	1	1

Wir wollen Bit 2 auf den Wert Null ändern. Ein Weg wäre, den Dezimalwert von 01101011 zu berechnen und diesen Wert dann in Speicherstelle 1 zu POKEn. Das funktioniert aber nur dann, wenn wir wissen, daß der vorherige Inhalt dieser Speicherstelle wirklich 01101111 war. Ein besserer Weg ist, die Anweisungen AND und PEEK zu verwenden. Die folgende Befehlszeile verwendet den Befehl PEEK, um den ursprünglichen Inhalt von Speicherstelle 1 zu ermitteln, verwendet AND, um ihn mit 251 (11111011 binär) zu verknüpfen und POKEt das Ergebnis wieder in Speicherstelle 1:

```
POKE 1, PEEK (1) AND 251
```

Die Auswirkungen dieses Befehls:

Bit	7	6	5	4	3	2	1	0	
	0	1	1	0	1	1	1	1	← ursprünglicher Inhalt
	1	1	1	1	1	0	1	1	← 251 binär
	0	1	1	0	1	0	1	1	← Ergebnis der Verknüpfung der Bit-Paare mit AND

Ganz gleich, welchen Originalwert Bit 2 hat, eine AND-Verbindung mit Null ergibt als Ergebnis Null. Die Binär-Zahl 11111011 (251 dezimal) nennt man eine Maske oder auch „overlay". In unserem Beispiel verwenden wir sie als „AND-Maske".

Um Bit 2 auf den Wert 1 zu setzen, ohne die anderen Bits zu beeinflussen, verwenden wir den folgenden Befehl:

```
POKE 1, PEEK(1) OR 4
```

Bit	7	6	5	4	3	2	1	0	
	0	1	1	0	1	0	1	1	← ursprünglicher Inhalt
	0	0	0	0	0	1	0	0	← 4 binär
	0	1	1	0	1	1	1	1	← Ergebnis der Verknüpfung der Bit-Paare mit OR

Wenn die Kopie vollständig ist, kann die CPU wieder zurückgesetzt werden, um die E/A-Geräte zu adressieren und den Zeit-Interrupt wieder neu zu starten.

Nun müssen wir den Video-Chip dazu „zwingen", unseren Zeichensatz anstelle des ROM-Zeichensatzes zu verwenden. Die Bits 0 bis 3 der Speicherstelle 53272 zeigen auf die Startadresse des Zeichensatzes. So verwendet der Commodore 64 die Werte dieser Bits als Adressen-Zeiger:

Dezimalwert der Bits 0 bis 3	Bits 3,2,1,0	Speicherstelle, auf die gezeigt wird
0	0000	0
2	0010	2048
4	0100	4096
6	0110	6144
8	1000	8192
10	1010	10240
12	1110	12288
14	1110	14336

Der Wert von Bit 0 in diesem Register ist unwichtig, da die Bits 4 bis 7 andere Funktionen kontrollieren und deshalb nicht verändert werden dürfen. Wir verwenden 11110000 (240 dezimal) als „AND-Maske" und 00001110 (14 dezimal) als „OR-Maske", damit die Adresse auf 14336 zeigt – die Start-Adresse unseres Zeichensatzes:

POKE 53272, (PEEK(53272) AND 240) OR 14

Jetzt können wir unter Verwendung einer FOR . . . NEXT-Schleife die Kopie beginnen.

Während das Programm den ROM-Zeichensatz in das RAM kopiert, kann die CPU keine Ein-/Ausgabe-Operationen handhaben. Wenn die CPU durch eine E/A-Operation unterbrochen würde, während der Zeichensatz den E/A-ROM-Bereich belegt, würde das System wahrscheinlich zusammenbrechen, und man müßte den Computer aus- und wieder einschalten. Glücklicherweise kann man diese Unterbrechungen vermeiden, indem Bit 0 von Speicherstelle 56334 auf den Wert Null gesetzt wird. Die anderen Bits dieser Adresse müssen unverändert bleiben. Also muß der folgende POKE-Befehl verwendet werden:

POKE 56334, PEEK (56334) AND 254

Nehmen wir einmal an, wir wollten die 64 Zeichen von „@" bis „?" kopieren. Hierzu müssen wir insgesamt 512 Speicherstellen (8x64=512), angefangen von 53248, in einen entsprechenden Speicherbereich des RAM kopieren. Wir haben uns für einen Bereich mit der Startadresse 14336 entschieden. Er liegt zwar normalerweise im BASIC-Speicherbereich, doch kann man diesen schützen, indem man den Zeiger für die Obergrenze des Speicherbereiches verlegt. Dies ist mit dem Inhalt von Speicherstelle 56 möglich:

POKE 56,32

Jede Reihe der Punkt-Matrix wird als Binärzahl interpretiert (Punkte, die aktiviert sind, zählen als 1, und Punkte, die auf dem Bildschirm nicht dargestellt werden, haben den Wert 0) und benötigt somit ein Byte Speicherplatz, das gesamte Zeichen also acht Bytes. Die Anfangsadresse der Bytes, die ein Zeichen repräsentieren, kann mit Hilfe der Startadresse des gesamten Speicherbereiches sowie des Bildschirm-Codes des gesuchten Zeichens wie folgt ermittelt werden:

Zeichenbeginn = 14336+8x(Bildschirm-Code)

Nun kann man neue Werte in diese Adressen POKEn, um das auf dem Bildschirm erscheinende Muster zu ändern. Solange der neue Zeichensatz aktiviert ist, wird beim Drücken der entsprechenden Taste das neue Zeichen auf dem Bildschirm erscheinen. In unserem Beispielprogramm haben wir die Zeichen [,], £ und ↑ (Codes 27–30) als Teile einer Figur definiert. Und durch Darstellung der verschiedenen Zeichen mit PRINT wurde eine Animation bewirkt.

```
130 :
140 REM**** COPY ROM CHAR SET ****
150 POKE56,32
   :REM LOWER TOP OF MEMORY
160 POKE56334,PEEK(56334)AND254
   :REM TURN OFF INTERRUPT TIMER
165 POKE1,PEEK(1)AND251
   :REM FLIP TO CHAR ROM
170 FOR I=0 TO 511
   :REM COPY
180 POKE14336+I,PEEK(53248+I)
   :REM 64
190 NEXTI
   :REM CHARACTERS
200 POKE1,PEEK(1)OR4
   :REM FLIP BACK TO I/O
210 POKE56334,PEEK(56334)OR1
   :REM TURN ON INTERRUPT TIMER
220 POKE 53272,(PEEK(53272)AND240)
OR14:REM SET CHAR POINTER
230 REM****  COPY COMPLETE  ****
240 :
250 :
300 REM**** ATHLETIC ARTHUR ****
310 FOR I=14552TO14552+31
   :REM READ
320 READ A:POKEI,A:NEXT I
   :REM CHAR DATA
330 PRINTCHR$(147)
   :REM CLEAR SCREEN
340 POKE55338,14:POKE55378,14
   :REM COLOUR CHRCTR. LIGHT BLUE
350 POKE1066,27:POKE1106,28
   :REM ARMS UP
360 FORI=1TO500:NEXT I
   :REM DELAY LOOP
370 POKE1066,29:POKE1106,30
   :REM ARMS DOWN
380 FORI=1TO500:NEXT I
   :REM DELAY LOOP
390 GOTO350
480 :
490 :
500 REM**** ARMS UP DATA ****
510 DATA129,153,189,153,66,60,60,60
520 DATA60,60,36,36,66,66,195,0
530 REM**** ARMS DOWN DATA ****
540 DATA0,24,60,24,0,126,189,189
550 DATA189,189,36,36,36,36,102,0
```

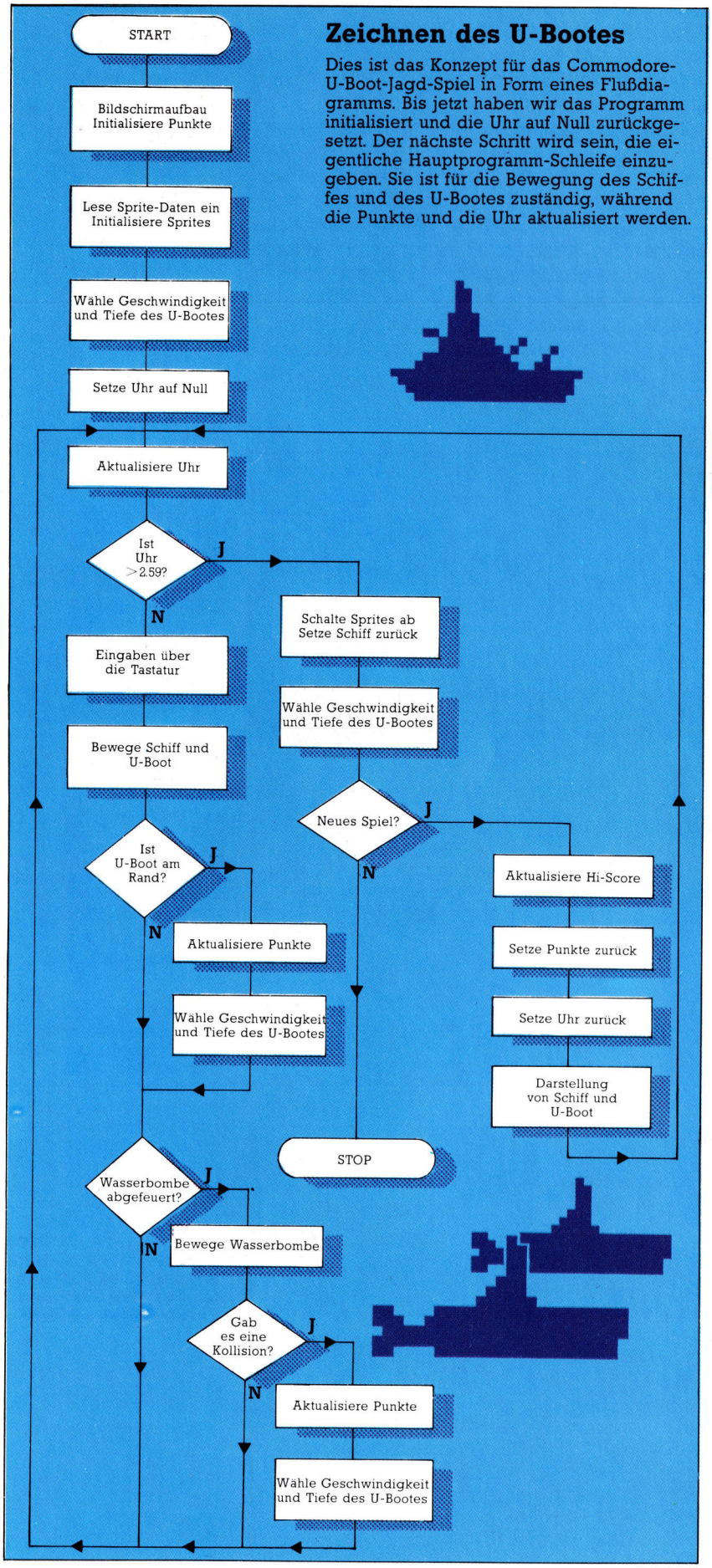
128	64	32	16	8	4	2	U	Dezimal
1	0	0	0	0	0	0	1	129
1	0	0	1	1	0	0	1	153
1	0	1	1	1	1	0	1	189
1	0	0	1	1	0	0	1	153
0	1	0	0	0	0	1	0	66
0	0	1	1	1	1	0	0	60
0	0	1	1	1	1	0	0	60
0	0	1	1	1	1	0	0	60
0	0	1	1	1	1	0	0	60
0	0	1	1	1	1	0	0	60
0	0	1	0	0	1	0	0	36
0	0	1	0	0	1	0	0	36
0	1	0	0	0	0	1	0	66
0	1	0	0	0	0	1	0	66
1	1	0	0	0	0	1	1	195
0	0	0	0	0	0	0	0	0

128	64	32	16	8	4	2	U	Dezimal
0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	24
0	0	1	1	1	1	0	0	60
0	0	0	1	1	0	0	0	24
0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	0	126
1	0	1	1	1	1	0	1	189
1	0	1	1	1	1	0	1	189
1	0	1	1	1	1	0	1	189
1	0	1	1	1	1	0	1	189
0	0	1	0	0	1	0	0	36
0	0	1	0	0	1	0	0	36
0	0	1	0	0	1	0	0	36
0	0	1	0	0	1	0	0	36
0	1	1	0	0	1	1	0	102
0	0	0	0	0	0	0	0	0

Zeichen werden in einer Acht-mal-acht-Punkt-Matrix aufgebaut und werden somit durch acht zusammengehörige Bytes repräsentiert. Jede Reihe eines Zeichens wird als eine Ein-Byte-Binärzahl verstanden, wobei Punkte eine Eins und Leerstellen eine Null repräsentieren. In Commodore-64-Programmen müssen diese Binärzahlen in Dezimalzahlen umgewandelt werden.

Zeichnen des U-Bootes

Dies ist das Konzept für das Commodore-U-Boot-Jagd-Spiel in Form eines Flußdiagramms. Bis jetzt haben wir das Programm initialisiert und die Uhr auf Null zurückgesetzt. Der nächste Schritt wird sein, die eigentliche Hauptprogramm-Schleife einzugeben. Sie ist für die Bewegung des Schiffes und des U-Bootes zuständig, während die Punkte und die Uhr aktualisiert werden.



U-Boot-Jagd

Der Commodore 64 hat seine eigene interne Uhr, die zur Steuerung von BASIC-Programmen verwendet werden kann. Die Uhr hat sechs Stellen, entsprechend einer Digital-Uhr, die die Stunden (00-23), Minuten (00-59) und die Sekunden (00-59) repräsentieren. Auf die Uhr kann von BASIC aus mit Hilfe der String-Variablen TI$ zugegriffen werden. Der Wert von TI$ gibt die Zeit an, die seit Einschalten des Computers verstrichen ist. Das folgende kurze Programm demonstriert, wie die Uhr arbeitet.

```
10 REM **** UHR****
20 PRINT CHR$ (147) :REM LOESCHT
   BILDSCHIRM
30 TI$="000000" :REM SETZT UHR AUF
   NULL
40 PRINT CHR$(145);TI$ :REM DRUCKT
   GEGENWAERTIGEN WERT VON UHR
50 :REM CHR$(145)=CURSOR HOCH
60 GOTO 40
```

Das Programm läuft in einer Endlos-Schleife und stellt den Inhalt der Uhr auf dem Bildschirm dar, bis Sie RUN/STOP drücken.

Bei dem U-Boot-Jagd-Spiel wird eine Uhr gezeigt. Das Spiel ist beendet, wenn drei Minuten vergangen sind. Die Spieluhr benötigt daher nur die Minuten- und Sekunden-Bereiche von TI$. TI$ können wir so unterteilen:

RIGHT$(TI$.2)
↓
TI$=HH(MM)(SS)
↑
MID$(TI$,3,2)

Die beiden Sekunden-Stellen können durch RIGHT$(TI$,2) und die Minuten-Stellen durch MID$(TI$,3,2) isoliert werden.

Die Hauptprogramm-Schleife unseres Spieles beginnt mit Zeilennummer 200 und endet mit Zeile 390. Laden Sie nun die Unterroutine aus dem letzten Teil unseres Kurses. Fügen Sie dann die folgenden Programmzeilen ein:

```
140 TI$="000000"
200 REM **** HAUPTSCHLEIFE ****
210 REM ** UHR **
220 PRINT CHR$(19);:TAB(14)CHR$(5)
    "ZEIT";MID$(TI$,3,2);":";RIGHT$
    (TI$,2)
225 IF VAL (TI$)> 259 THEN 400:REM
    ENDE DES SPIELES
390 GOTO 200:REM WIEDERHOLE
    HAUPTSCHLEIFE
400 END
```

Durch Zeile 140 wird die Uhr beim Programmstart zurückgesetzt. In Zeile 220 wird der gegenwärtige Wert der Uhr in Minuten und Sekunden, getrennt durch einen Doppelpunkt, auf den Bildschirm gePRINTet. TAB(14) bewirkt, daß 14 Leerstellen vor der Darstellung der Zeit freigelassen werden. Überschreitet die Spielzeit zwei Minuten und 59 Sekunden, wird das Spiel beendet.

Ein Paket mit Überraschungen

Der Begriff „integrierte Software" ist zu einem Modewort der Programmierbranche geworden. Wir untersuchen, was Integration in diesem Bereich bedeutet und welche Vor- und Nachteile diese Systeme bringen. Im weiteren Verlauf der Serie werden wir uns einzelne Programmpakete genauer ansehen.

Wichtigstes Merkmal integrierter Programme ist die Möglichkeit, zwischen unterschiedlichen Anwendungen schnell und problemlos umschalten zu können. In einem idealen System sollte es dabei nicht einmal notwendig sein, zum Betriebssystem zurückzukehren, Disketten zu wechseln und das andere Programm starten zu müssen. Brauchbare Systeme sollten die einzelnen Anwendungen fast auf Knopfdruck wechseln können. Einige Programme wie „Lotus 1-2-3" und „Framework" von Ashton-Tate erfüllen diese Bedingungen bereits.

In dieser Serie werden wir unter anderem Lotus 1-2-3, Open Access, Symphony und Framework untersuchen, die auf größeren Systemen laufen.

Kalkulation und Text

Wichtig ist auch, daß die einzelnen Anwendungen untereinander problemlos Daten austauschen können. So lassen sich zum Beispiel die jährlichen Verkaufszahlen mit einem Kalkulationssystem erstellen, dann direkt in das Textprogramm übertragen und dort in den Jahresbericht einfügen. Über die Namen und Adressen des Datenbanksystems senden Sie dem gewünschten Personenkreis dann diesen Bericht mit einem persönlichen Begleitbrief. Auf dem Lisa und dem Macintosh wurde diese Möglichkeit so weit ausgebaut, daß Sie im Grafikprogramm eine Zeichnung erstellen und sie dann direkt ohne Qualitätsverluste in das Textsystem übertragen können.

Alle unterschiedlichen Programme sollten nach dem gleichen Prinzip funktionieren. Bildschirmaufbau, Befehlstasten, Eingabeaufforderungen und Fehlermeldungen müssen bei einem ausgereiften Programm identisch sein. Ist dies der Fall, kann der Anwender von einem Programm zum anderen übergehen, ohne sich auf eine veränderte Betriebsumgebung einstellen zu müssen.

Eine angenehme Nebenwirkung der Integration ist die Leichtigkeit, mit der sich die Bedienung eines derartigen Paketes erlernen läßt. Um fünf verschiedene Anwendungen bedienen zu können, von denen einige per Menü und andere über Befehle gesteuert werden, wäre viel Zeit nötig. Sind jedoch alle Programme nach dem gleichen Prinzip aufgebaut, braucht nur ein Befehlssatz erlernt zu werden. Die einheitliche Bedienung ist daher ein wesentlicher zeitsparender Bestandteil integrierter Software.

Es lassen sich also drei grundlegende Merkmale der integrierten Software aufzeigen: die Leichtigkeit, mit der von einer Anwendung in eine andere übergewechselt werden kann; die Möglichkeit, Daten zwischen Programmen auszutauschen, und das einheitliche Format verschiedenartiger Anwendungen.

Hoher Speicherplatzbedarf

Integrierte Software hat jedoch auch Nachteile, wobei der große Speicherbedarf an erster Stelle steht. Stellen Sie sich vor, Sie müßten ein Textprogramm, ein Kalkulationssystem und eine Datenbank (die Bestandteile der meisten integrierten Systeme) in 16 oder 32 KByte unterbringen. Diese Aufgabe ließe sich sicherlich bewältigen, doch bliebe nur wenig Platz für Daten übrig. Integrierte Software läuft daher fast ausschließlich auf Maschinen mit großem Arbeitsspeicher – das heißt Computer mit 128 oder mehr KByte.

Aus dem gleichen Platzproblem ergibt sich noch eine weitere Schwäche der integrierten Software: Die Textverarbeitung eines integrierten Systems mit zwei oder drei weiteren Programmen kann nie so umfassend sein wie ein eigenständiges Textprogramm, das den gesamten Speicherplatz zur Verfügung hat.

Universelle Befehlssätze

Zwei Programme, die auf dem IBM PC und ähnlichen Computern laufen, sind Beispiele für diese Schwierigkeit. „Multimate" wurde auf die Software der auf Wang ausgerichteten Textverarbeitung zugeschnitten. Das Programm besitzt für die Erstellung auch umfangreicher Schriftstücke viele Aufbau- und Formatierungsmöglichkeiten, die es auf kleineren Systemen nicht gibt. Multimate belegt dabei 192 KByte des Arbeitsspeichers. Lotus 1-2-3, mit integrierter Textverarbeitung, Kalkulationssystem und Datenbank, braucht ebenfalls 192 KByte RAM. Der gleiche Platz, der bei Multimate für eine einzige Anwendung eingesetzt wurde, muß bei Lotus 1-2-3 für drei völlig verschiedene Programme ausreichen. Das Textsystem des Lotus läßt sich daher nur für die Erstellung einfacher Schriftstücke einsetzen.

Der dritte Nachteil integrierter Programme resultiert aus dem zwangsweise gleichen Aufbau und den identischen Befehlssätzen. Die beste Methode, ein Kalkulationssystem zu bedienen, mag nicht in gleicher Weise für eine Datenbank oder ein Textprogramm geeignet sein. Die auf einzelne Programme ausgerichteten Befehle werden daher oft zu einer Mixtur verschnitten, die sich universell einsetzen läßt.

Arbeitserleichterung

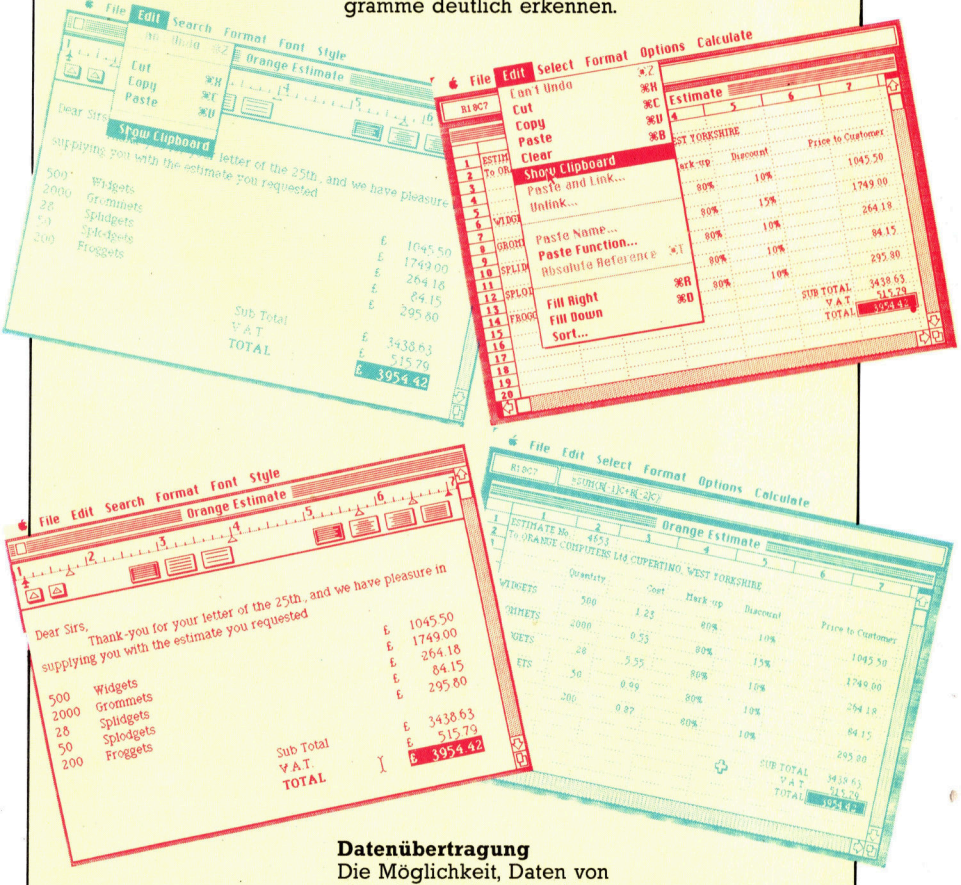
Wichtig ist dabei, daß die Software die Wünsche des Anwenders erfüllen muß. Sollen mehrere Aufgaben ausgeführt werden, wie das Schreiben von Briefen, einfache Buchhaltung und der Druck von Adreßlisten, läßt sich durch ein integriertes System die Arbeit sehr vereinfachen. Da die Module fortwährend verkleinert werden, während die Kapazitäten der Heimcomputer ständig wachsen, wird integrierte Software – auch für die Besitzer von Heimcomputern – mehr und mehr an Bedeutung gewinnen.

Im weiteren Verlauf dieser Serie werden wir einige integrierte Programme untersuchen, die die Softwareentwicklung wesentlich beeinflußt haben. Wir werden dabei zwei verschiedenartige Ansätze verfolgen: zunächst das in Lotus 1-2-3 und ähnlichen Programmen verwirklichte Konzept, das das gewohnte Format von Computerprogrammen hat, und weiterhin die Systeme, die auf Maschinen wie Lisa und Macintosh eingesetzt werden und bei denen die gesamte Betriebsumgebung auf die Integration ausgerichtet ist.

Spielregeln

Volle Ausstattung
Integration heißt das Grundprinzip des Macintosh von Apple, auf dem diese Illustrationen erstellt wurden. Multiplan (das Kalkulationssystem), MacWrite (die Textverarbeitung) und MacPaint (das Grafikprogramm) können über das Betriebssystem ihre Daten miteinander austauschen. Die drei unterschiedlichen Programme verschmelzen damit zu einer einzigen Anwendung.

Einheitlicher Befehlssatz
In dem Bildschirmaufbau des Kalkulationssystems und der Textverarbeitung läßt sich das einheitliche Format der integrierten Programme deutlich erkennen.



Datenübertragung
Die Möglichkeit, Daten von einer Anwendung in eine andere zu übertragen (hier zwischen dem Kalkulationssystem und der Textverarbeitung), ist einer der wesentlichen Bestandteile integrierter Systeme.

Rundumblick

Der Kauf eines Heimcomputers ist oft nur der erste Schritt zu einem kompletten System. Der Rechner kann durch zusätzliche Peripheriegeräte erheblich erweitert werden. Wir geben hier eine Übersicht einiger populärer Erweiterungen und dazu Kauftips.

Bis vor kurzem war einer der wichtigsten Erweiterungsbausteine des privaten Anwenders ein Speichererweiterungs-Steckmodul. Speicherchips waren teuer, und Rechner wie der ZX 81 und VC 20 wurden unter dem Gesichtspunkt niedriger Produktionskosten hergestellt. Der ZX 81 in der Grundversion verfügt denn auch nur über 700 Bytes zur Programmerstellung. Die neuen Rechner sind mit 64 KByte RAM oder mehr ausgestattet. Speichererweiterungsmodule als Peripherie sind folglich kaum mehr gefragt. Der Anwender hat heute eine schier unüberschaubare Menge an peripheren Bausteinen zur Auswahl: Modems erlauben die Kommunikation zwischen Computerfreunden, die Hunderte von Kilometern voneinander entfernt wohnen. Motorisierte Fahrzeuge und Roboterarme können mit einem passenden Interface gesteuert werden.

Obwohl das Peripherie-Angebot auf dem Markt sehr groß ist, stehen die meisten Zusatzgeräte nur für die bekannteren Rechner zur Verfügung. Daran sollte man denken, bevor man sich für ein System entscheidet. Es dauert gewisse Zeit, bis ein umfangreiches Peripherieangebot für neuere Rechner auf den Markt gelangt. Allerdings wird der unlängst eingeführte MSX-Standard diesen Prozeß vereinfachen, da die Peripherie dann zu allen Rechnern paßt, die die MSX-Spezifikation haben.

Der wichtigste Punkt, den man beim Kauf eines Peripheriegerätes beachten sollte, betrifft die Kompatibilität: Jede Erweiterung muß mit anderen Peripheriegeräten, die man später kauft, zusammenarbeiten können. Das klassische Negativbeispiel ist der Sinclair Spectrum. Viele Spectrumbesitzer kauften das Interface 1 und einen oder zwei Microdrives, nur um feststellen zu müssen, daß verschiedene Peripheriegeräte – und ein Teil der Software – nach Installation des Interface 1 nicht korrekt arbeiten bzw. gar nicht laufen.

Ist die Frage nach der Kompatibilität aber erst einmal geklärt, erhöhen die Erweiterungen den Spaß am Computer. Mit geeigneten Peripheriegeräten kann man ein Computersystem nach eigenen Wünschen und Bedürfnissen zusammenstellen.

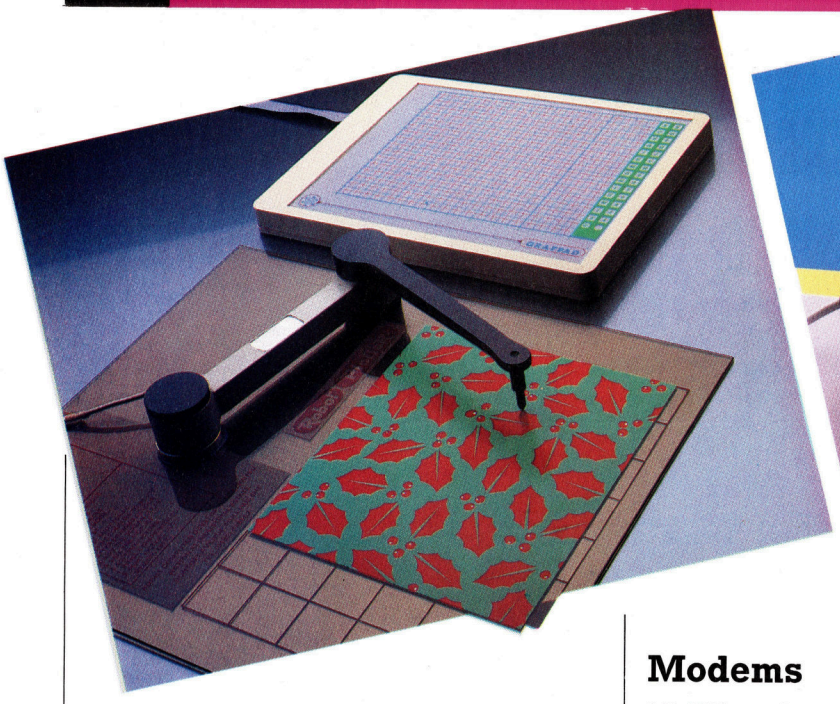
Speichersysteme

Der am meisten verbreitete Massenspeicher für Microcomputer ist der gewöhnliche Cassettenrecorder. Er ist leicht zu bedienen und preiswert, seine Schwächen werden aber rasch offensichtlich. Es dauert lange, bis die Programme geladen sind, und der Zugriff auf gespeicherte Daten ist sehr umständlich. Diskettenstationen arbeiten schneller und exakter, kosten aber mehr. Viele Heimcomputer arbeiten nur mit bestimmten Typen von Diskettenstationen zusammen, von denen einige sehr langsam laufen. Die meisten Diskettenstationen für den Heimgebrauch entsprechen der 5 1/4-Zoll-Norm, doch nach und nach werden auch die 3-Zoll- und 3 1/2-Zoll-Stationen gebräuchlich. Der Oric-Atmos-Drive arbeitet beispielsweise mit 3-Zoll-Disketten, die pro Seite eine Kapazität von 160 KByte haben. Das „Torch Disc Pack" verwandelt den AcornB in einen völlig „neuen" Computer, da es mit einem Z80-Prozessor ausgestattet ist und zusätzliche 64 KByte bringt.

Sinclair entwickelte das Interface 1/Microdrive-System, bei dem ein Endlosband zur Speicherung von rund 85 KByte an Daten verwendet wird. Das Band wird vom Computer gesteuert und ermöglicht so einen Zugriff auf jede beliebige Stelle in maximal zehn Sekunden. Damit wurde ein Mittelding zwischen Cassettenrecorder und Diskettenstation geschaffen, – erhältlich zu einem Preis, der deutlich unter dem einer Floppy liegt. Das Interface 1 enthält zudem ein RS232-Interface und kann zur „Vernetzung" verwendet werden.

Auch beim Rotronics Wafadrive wird eine Schleife zur Datenspeicherung verwendet, doch im Preis sind außerdem RS232-und Centronics-Schnittstellen sowie ein Textverarbeitungsprogramm enthalten.

Im Bild sind der Rotronic Wafadrive, die Oric-Atmos-Diskettenstation, das Torch Disc Pack sowie das Sinclair Interface 1 zu sehen.



Grafikhilfsmittel

Die Erzeugung von Grafiken auf einem Heimcomputer wird durch spezielle Grafikgeräte erheblich vereinfacht. Am billigsten ist ein Lichtgriffel, der zum direkten „Zeichnen" auf dem Bildschirm verwendet wird. Durch eine Fotozelle an der Spitze des Lichtgriffels kann die genaue Position festgestellt werden. Eine Weiterentwicklung daraus ist die Stack Light Rifle, die als Steuereinheit bei manchen Spielen verwendet werden kann.

Bei Grafiktabletts werden spezielle Stifte verwendet, die die Bewegung auf der Oberfläche des Tabletts auf den Computer übertragen. Andere „Zeichen"-Geräte arbeiten digital, so etwa der Tracer. Hier befinden sich Widerstände in einem mechanischen Arm, mit denen die Position des Griffels bestimmt wird.

Im Bild sind das Grafpad von British Micro, der Digitalzeichner Robot Plotter sowie Lichtgriffel und Light Rifle von Stack zu sehen.

Modems

Mit Hilfe preiswerter Modems können Heimcomputer mittels Telefon miteinander kommunizieren. Für Rechner, die über eine Standard-RS232-Schnittstelle verfügen, gibt es eine Fülle von Modems. Die entsprechende Software vorausgesetzt hat man Zugriff auf unterschiedliche Datenbanken. Modems können außerdem dazu benutzt werden, mit anderen Anwendern über Mailboxen zu kommunizieren. Das sind Datenbanken, die von Computerfans als Hobby betrieben werden. Auch hier ist die Kompatibilität zu beachten. Unterschiedliche Mailboxen verwenden unterschiedliche Übertragungsraten, und ein Modem, das für Prestel geeignet ist, kann für die Verbindung mit einer Mailbox ungeeignet sein.

Das in England bestverkaufte Modem für den Spectrum ist das Prism VTX5000.

Es verfügt über integrierte Software, die den Zugriff zu Prestel ermöglicht. Auf Band gespeicherte Software erlaubt darüber hinaus den Datenaustausch zwischen Spectrum-Rechnern, sofern diese mit dem Prism ausgerüstet sind.

Hier sind das Prism VTX 5000 und das Commodore-Modem abgebildet.

Sprach-Synthesizer

Die im Angebot befindlichen Geräte lassen sich in zwei Klassen unterscheiden: Der eine Typ verfügt über ein festgelegtes Vokabular von rund 100 verschiedenen Wörtern. Der andere arbeitet mit sogenannten „Allophonen" – einer Sammlung verschiedener Laute, aus denen Wörter gebildet werden.

Currah verwendet für seine Synthesizer die Allophon-Technik. Dieses Haus bietet Sprach-Synthesizer für den Spectrum (Microspeech) und den C 64 (Speech 64) an. Manche Spectrum- und C-64-Programme sind mit Sprache versehen, die automatisch erzeugt wird, wenn ein Currah-System angeschlossen worden ist.

Im Bild sind der Currah Speech 64 und der Cheetah Sweet Talker für den Spectrum.

Computergesteuerte Geräte

Man kann Computer auch dazu benutzen, Geräte zu steuern. Die am meisten zitierte Anwendung dieser Art ist die Kontrolle und Steuerung eines Heizungssystems. Weitaus mehr Vergnügen macht das Steuern von Bodenrobotern. Die „Valiant Turtle" hat eine entfernte Ähnlichkeit mit einer Schildkröte, und kann unter Einsatz von LOGO Grafiken erzeugen. Sie arbeitet mit dem Spectrum, dem C 64 und dem Acorn B und wird über Infrarot vom Computer gesteuert. Der BBC-Buggy wird durch Draht mit dem Computer verbunden. Er kann ebenfalls zum Zeichnen von Linien benutzt werden und ist mit Sensoren ausgestattet.

Unser Foto zeigt die Valiant Turtle und den BBC Buggy.

Drucker/Plotter

Jeder, der seinen Heimcomputer zum Programmieren oder für Textverarbeitung verwendet, benötigt einen Drucker. Bei Matrix-Druckern wird ein Raster aus winzigen Punkten zur Erzeugung der Buchstaben verwendet. Dieser Drukkertyp arbeitet schnell, liefert aber kein besonders gutes Schriftbild. Anders der Schönschreibdrucker, der im Grunde nichts anderes als eine gute computergesteuerte Schreibmaschine ist.

Alternativ wird ein kombinierter Drucker/Plotter für Tandy, Atari, Commodore und Oric angeboten. Hierbei wird ein etwa zehn Zentimeter breites Papier verwendet, auf dem mit vier kleinen Farbstiften die Darstellung von Grafiken oder mehrfarbigem Text möglich ist. Der Text wird ebenso „gezeichnet" wie die Grafiken. Das Gerät ist mit einem kompletten Zeichensatz programmiert. Eine weitere Möglichkeit stellt der Epson-P40-Thermodrucker dar, bei dem die Textdarstellung auf Spezialpapier erfolgt. Das System ist sehr preiswert, liefert ein brauchbares Schriftbild und wird mit wiederaufladbaren Batterien betrieben.

Im Bild sind Printer/Plotter (Ausführung Tandy/Radio Shack) und Epson-P40-Thermodrucker zu sehen.

Joysticks

Das erste Peripheriegerät, das Heimcomputerbesitzer normalerweise kaufen, ist der Joystick. Viele Computer sind mit passenden Schnittstellen ausgestattet und einige der neueren Rechner werden sogar mit Joysticks geliefert. Der verbreitetste Joystick ist mit der neun-Pin-"D"-Buchse ausgestattet.

Sinclair hat das Interface Zwei auf den Markt gebracht, eine Joystick-Schnittstelle und zugleich einen ROM-Steckmodulschacht für den Spectrum. Bis zu dieser Entwicklung gab es für den Rechner keine „offizielle" Joystick-Schnittstelle. Standard war das Kempston-Interface, dessen technische Merkmale von vielen anderen Herstellern übernommen wurden. Leider sind die beiden Schnittstellen nicht kompatibel. Kempston hat allerdings inzwischen ein Interface entwickelt, das mit der Software kompatibel ist, die sowohl für das alte Kempston-Interface als auch für das Interface Zwei geschrieben wurde.

Unter den vielen auf dem Markt befindlichen Joysticks ist der Cheetah-RAT der wohl ungewöhnlichste. Er ist nicht durch ein Kabel mit dem Computer verbunden, sondern sendet und empfängt Infrarot-Signale. Bisher gibt es diesen Joystick nur für den Spectrum.

Unser Foto zeigt (von links nach rechts): Schneider Joystick, Cheetah-RAT, Kempston-PRO-5000 und das Kempston-Interface für den ZX-Spectrum.

Monitore

Die meisten Computer werden mit einem Fernseher als Bildausgabeeinheit betrieben. Das schafft oft Probleme, da Familienmitglieder lieber Fernsehen wollen, wogegen sich der Computerbesitzer auf Pacman freut. Zudem ist die Bildqualität meist nicht sonderlich gut. Abhilfe schafft da nur ein Monitor. Es gibt zwei Standards – RGB und Composite-Video. Composite-Video-Monitore werden bei Atari und Commodore-Rechnern benutzt, wogegen für den Acorn, Oric Atmos und Sinclair QL RGB-Monitore erforderlich sind. Einige Microcomputer nutzen den Fernsehlautsprecher für die Klangausgabe, also sind dafür Monitore mit integrierten Lautsprechern erforderlich. Mehrere Fernsehhersteller produzieren inzwischen Geräte, die mit Monitor-Schnittstellen ausgestattet sind.

Abgebildet ist der Microvitec Cub.

Wort für Wort

Im zweiten Teil unserer PASCAL-Serie befassen wir uns mit grundlegenden Voraussetzungen wie der Syntax und dem Vokabular.

Compiler
Eine komplette professionelle PASCAL-Ausrüstung kann oft mehr kosten als ein Heimcomputer. Doch gibt es inzwischen viele Compiler, die zu einem vernünftigen Preis erhältlich sind. Wir zeigen Ihnen hier eine Auswahl der zur Zeit erhältlichen Programmpakete.

Das erste Problem bei der Verwendung einer Compilersprache ist der Umgang mit dem mehrstufigen Arbeitsprozeß, selbst bei einem kleinen Programm. Als erstes muß der Source-Code mit Hilfe eines Editors oder eines Textverarbeitungsprogramms eingegeben werden. Dann, nachdem man den Source-Code auf Cassette oder Diskette gespeichert hat, muß der Compiler geladen und anschließend instruiert werden (oftmals in Form einer komplexen Befehlszeile), den Source-Code in Maschinen-Code umzuwandeln bzw. zu compilieren. Abschließend muß dann das „Object"-File einem Speicherbereich zugeordnet und mit den entsprechenden notwendigen „Run-Time-Library"-Routinen verbunden werden. In den meisten Fällen kann das Programm dann ohne weiteren Aufwand geladen und gestartet werden, doch sollte der Compiler einen sogenannten „Pseudo-Code", auch Zwischencode genannt, verwenden, muß ein Run-Time-Interpreter benutzt werden, um das Programm auszuführen.

Doch fast alle für Heimcomputer erhältlichen PASCAL-Versionen vermeiden diese Probleme weitestgehend. Bei den besten Versionen ist es sogar möglich, während der Programmentwicklung den Source-Code, den Compiler und das Object-Programm gleichzeitig im Speicher zu haben. Die Effizienz und der geringe Platzbedarf von PASCAL machen dies möglich. Lediglich beim Schreiben von Programmen muß man sich mit einem etwas komplizierteren Arbeitsablauf abfinden.

Jedes System verfügt über einen individuellen Satz an Befehlen zur Kontrolle des Editors und des Compilers. Oftmals ist nur ein einfaches E für Editieren, ein C für Compilieren und ein R für RUN (Programmstart) alles, was Sie wissen müssen. Zunächst befassen wir uns mit der korrekten Syntax, die für jedes Programm, egal wie einfach oder komplex es sein mag, eingegeben werden muß. Glücklicherweise ist PASCAL so gut standardisiert, daß es kaum Unterschiede zwischen den einzelnen Dialekten gibt (im Gegensatz zu den verschiedenen BASIC-Versionen). Lediglich gegen Ende des Kurses werden wir uns mit einigen wenigen Ausnahmen befassen. Lassen Sie uns nun unser erstes vollständiges PASCAL-Programm betrachten:

```
Program First (output);
Const
   Message='Pascal-Programmierung';
Begin
   write (Message)
End.
```

Geben Sie zunächst die Zeilen ein. Anschließend compilieren und starten Sie es. Sollten Sie irgendwelche Beanstandungen vom Compiler erhalten, lesen Sie die Fehlermeldung sorgsam durch, und versuchen Sie, den Fehler zu erkennen. Jedes Zeichen muß exakt eingegeben werden. Beachten Sie besonders das Semikolon am Ende der ersten und dritten Zeile sowie den Punkt am Ende des Programms. Wenn Sie das Programm korrekt eingegeben haben, werden Sie die folgende Meldung auf dem Bildschirm sehen:

```
Pascal-Programmierung
```

Das Programm mag etwas trivial erscheinen, doch es demonstriert die allgemeine Form, die jedem PASCAL-Modul (bzw. Programm, Unterroutine oder Funktion) zu eigen ist. Es gibt jeweils drei separate Teile:

1. Die Kopfzeile, in diesem Fall die Programm-Überschrift.
2. Deklarationen und Definitionen, in diesem Fall eine einzige konstante Definition.
3. Der „Körper", der alle ausführbaren Anweisungen enthält.

Die erforderliche Syntax von PASCAL, zumindest für die grundlegenden Funktionen der Sprache, kann am besten durch sogenannte „Syntax-Diagramme" definiert werden. Sie sind vergleichbar mit der Straßenkarte

Paket: Hisoft PASCAL für:
Spectrum, Schneider und MSX Computer

Beschreibung:
Eine vom Standard unbedeutend abweichende Version, aber trotzdem ein guter Kauf. Das Paket wird mit einem eigenen Editor und einer „Turtle Graphics Library" (Bibliothek) geliefert.

Paket: Acorn Iso PASCAL für:
Acorn B, Electron

Beschreibung:
Sehr hochwertig. Es gibt zwei Compiler: einer im ROM mit einem semi-intelligenten Editor und Compilierung im Speicher, der andere als diskettenorientiertes System für Besitzer eines Rechners mit 6502-CPU.

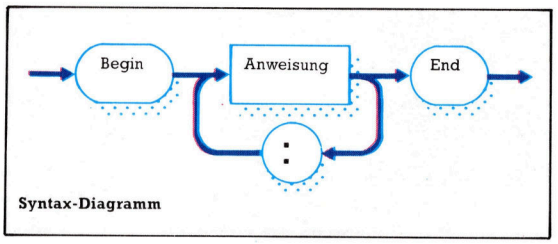
Paket: Oxford PASCAL für:
Commodore 64

Beschreibung:
Ein ebenfalls guter Kauf, produziert von Limbic Systems.

35 reservierte PASCAL-Wörter

And	Nil
Array	Not
Begin	Of
Case	Or
Const	Packed
Div	Procedure
Do	Program
Downto	Record
Else	Repeat
End	Set
File	Then
For	To
Function	Type
Goto	Until
If	Var
In	While
Label	With
Mod	

eines Einbahnstraßen-Systems. Ein legaler Weg durch das Diagramm führt von links oben nach rechts unten. Jeder Kasten, durch den wir auf diesem Weg kommen, ist entweder eine „syntaktische Einheit" (das bedeutet, sie repräsentiert sich selbst), dargestellt durch eine runde Box, oder ein anderes Objekt, das an einer anderen Stelle durch ein separates Syntax-Diagramm definiert wird. Diese werden in einem rechteckigen Kasten dargestellt.



Syntax-Diagramm

Bei der Betrachtung des Gesamtdiagramms eines Programms kann man sehen, daß die Worte „Begin" und „End" als Teile des PASCAL-Vokabulars definiert sind und somit keine weiteren Diagramme zu Erklärung ihrer Bedeutung benötigen. In der Tat haben bei PASCAL nur 35 Wörter eine festgelegte Bedeutung. Der Vollständigkeit halber haben wir sie alle am Ende dieses Kursteils aufgelistet. Unser erstes Programm verwendet nur vier dieser Wörter: Program, Const (die Abkürzung für constant = konstant), Begin und End. Das auf „Program" folgende Wort ist ein sogenannter „Identifikator", der den Programm-Namen angibt. Es kann aber auch ein anderes „legales" Wort verwendet werden.

Wenn man mit einem Buchstaben beginnt, und nur Buchstaben oder Zahlen verwendet, gibt es eine unbestimmbar große Anzahl von Wörtern, die Sie benutzen können. Trotzdem ist es nicht möglich, ein reserviertes Wort zu verwenden. Die folgenden Wörter sind beispielsweise alle erlaubt:

Name
PASCAL
ProgramEins
N
XYZ123
Adresse12
KnutRaber
EinsehrlangerIdentifikator

Die jetzt folgenden Wörter sind nicht legal:

Prog-1
ZEHN°
Feuerzeug.Ofen
and
Zeit$Ecke
Wiegeht's
1001Dalmatiner
Es regnet

Diese Wörter sind nicht erlaubt, da sie Zeichen enthalten, die nicht alphanumerisch sind, mit einer Zahl beginnen oder (im Fall and) eines der von PASCAL reservierten Wörter darstellen. Das letzte Beispiel ist illegal, da eine Leerstelle zum Trennen der Wörter verwendet wurde, wobei die Wörter an sich („Es" und „regnet") separat legal wären. Bei PASCAL besteht im allgemeinen, ähnlich wie in der englischen Sprache, kein Unterschied zwischen Klein- und Großbuchstaben, obwohl es einige Versionen gibt, bei denen reservierte Wörter in Großbuchstaben einzugeben sind.

Neben Leerstellen und dem Ende einer Zeile gibt es in der PASCAL-Syntax eine weitere Möglichkeit, die als Trennung verwendet werden kann – eine Anmerkung. Sie kann an einer beliebigen Stelle erscheinen, ausgenommen mitten in einem Wort. Anmerkungen werden durch geschwungene Klammern abgegrenzt.

Lassen Sie uns ein etwas komplexeres PASCAL-Programm betrachten:

```
Program ProgramZwei (input, output);
  {Quadrat einer Zahl}
Const
  Prompt = 'Gib Zahl ein:';
Var
    zahl : integer;
Begin
  WriteLn;
  WriteLn;
  write (Prompt);
  read (zahl);
  WriteLn (zahl, 'Quadrat ist', zahl*zahl)
End.
```

Beachten Sie, daß wir den Identifikator „input" in die Kopfzeile integriert haben. „Input" und „output" werden von PASCAL benötigt. Sie identifizieren die externen Dateien, mit denen das Programm kommuniziert. Bei einem Heimcomputer sind das normalerweise die Tastatur und der Bildschirm.

Der Speicher zum Ablegen dieser Parameter wird durch die Var-Deklaration reserviert, wobei dies in unserem Beispiel eine einzige ganze Zahl ist. Im Gegensatz zu BASIC, das normalerweise nur zwischen Zahlen und Strings unterscheiden kann (durch Anhängen eines Dollar-Zeichens an den Identifikator), verfügt PASCAL über einen nahezu unbeschränkten Bereich an verfügbaren Datentypen. Daher ist es wichtig, dem Compiler mitzuteilen, wieviel Speicherplatz er zum Speichern der jeweiligen Datensätze reservieren muß.

Der Cursor bleibt direkt an der Position nach Darstellung der Meldung stehen, so, als hätten wir eine BASIC-PRINT-Anweisung mit einem Semikolon verwendet. Dies ist kein Fehler, sondern ein Merkmal der write-Funktion. Wann immer man eine neue Zeile benötigt, muß man die alternative Anweisung WriteLn verwenden. Ln steht für Line (Zeile). Es ist recht sinnvoll, wenn man ein großes W und L verwendet, um den Anfang der zusammengesetzten Wörter besser zu kennzeichnen. Eine einfache WriteLn-Anweisung ohne weitere Parameter erzeugt eine neue Zeile.

Kleiner Schlucker

Ursprünglich für die Spielhallen entwickelt, sind zwischenzeitlich unzählige „Pacman"-Varianten für Heimcomputersysteme auf dem Markt. Atarisoft hat zwar intensiv gegen Copyrightverstöße vorzugehen versucht, doch das Ergebnis waren lediglich optisch abweichende Spiele.

Basierend auf dem Original-Spielhallen-Labyrinthspiel „Pacman" wurden zahlreiche Heimcomputerprogramme geschrieben, von „Atic Atac" bis „Jet Set Willy". In allen diesen Spielen muß der Hauptdarsteller durch viele Gänge oder Räume gesteuert werden und unterwegs Gegenstände sammeln, gleichzeitig aber vielfältigen Gefahren ausweichen. Unter gewissen Voraussetzungen hat der Spieler die Möglichkeit, den Spieß umzudrehen und die angreifenden Monster zu bedrängen.

Beim „Pacman" befindet sich die Hauptfigur im Zentrum eines Labyrinths, und der Spieler hat die Aufgabe, sie über den Schirm zu steuern und dabei die auf dem Weg liegenden Punkte zu verschlucken. Dabei ist auf die Geister zu achten, die einen bestimmten Kurs verfolgen, um „Pacman" in den verschiedenen Sackgassen des Labyrinths in die Falle zu lokken. Das Fressen von „Energiepillen" erlaubt dem Spieler, umgekehrt die Geister jagen zu können. Das bringt Extrapunkte. Zufällig erscheinende Früchte können ebenfalls vertilgt werden und erhöhen den Punktestand.

Für Atari, Commodore, Spectrum

Der Vergleich verschiedener Computer-Versionen ein- und desselben Spiels mag ein wenig unfair sein. Schließlich holt ein Programm, das für ein bestimmtes System entwickelt wurde, das beste aus dem Rechner heraus. Das gilt besonders für Programme, die im Original für die Atari-Computer geschrieben wurden. Diese sind ja für hochwertige, wenngleich teure, spielhallenähnliche Software bekannt.

„Pacman" auf den Atari-Computern – das ist ein großes Labyrinth mit ausgezeichnetem Ton und guter Grafik, glatter Sprite-Bewegung und verschiedenen Schwierigkeitsgraden. Beim VC 20 hat das Labyrinth nur ein Viertel der Größe des Atari-Originals, wogegen die Sprites doppelt so groß sind. Der geringe Freiraum macht es äußerst schwer, den Geistern auszuweichen. Die Grafiken sind allerdings gut definiert, die Bewegung verläuft gleichmäßig, und der Sound entspricht dem der Atari-Version. Bei der Spectrum-Variante hat der Spieler überdies die Möglichkeit, wahlweise über Tastatur oder Joysticks zu steuern. An den begrenzten Soundmöglichkeiten des Spectrum konnten die Atarisoft-Programmierer natürlich nicht vorbei; enttäuschend ist aber, wie wenig man grafisch gemacht hat. Das Labyrinth entspricht zwar dem Original, die ruckartigen Bewegungen aber erwecken den Eindruck, als betrachte man einen alten Stummfilm. Die Schwächen bei der VC 20-Version sind hardwarebedingt, aber die Schwächen des Spectrum-Programms sind unerklärlich.

Gleich nach seiner Erstveröffentlichung 1980 wurde „Pacman" ein Superhit. Atarisoft hatte seinerzeit sehr schnell Vorsichtsmaßnahmen gestartet und konkurrierende Softwareunternehmen an dem Vertrieb von „Pacman"-Plagiaten gerichtlich zu hindern versucht. Doch es verstrich zu viel Zeit, bevor man sich entschloß, eine „offizielle" Version für andere Rechner zu veröffentlichen. Vor einigen Jahren noch hätte Atari Zigtausend „Pacman"-Programme verkaufen können. Heute gibt es weitaus bessere Programme in Arcadequalität auf dem Markt.

Pacman: Für Atari-Computer, Commodore VC 20 und Spectrum

Hersteller: Atari Corp. (Deutschland) GmbH

Autor: Atari

Joysticks: Erforderlich

Format: Atari und VC 20: Steckmodul; Spectrum: Cassette

Motorsteuerung

Der letzte Teil unseres Selbstbau-Kurses enthielt die Anleitung zum Bau eines Interface, mit dem Niedervolt-Geräte gesteuert werden können. Auch das Prinzip einer Motor-Steuerung wurde schon beschrieben. Nun geht es an die Software, wozu auch die Entwicklung einer Regelung mit Rückkopplung gehört.

In der Industrie ist die Steuerung externer Maschinen durch Microprozessoren längst eingeführt, wobei die Einsatzmöglichkeiten vom Flaschenzählen bis zum Karosserieschweißen reichen. Dazu müssen Daten meist erst in eine für den Prozessor verständliche Form übersetzt werden. In weiteren Schritten werden diese Daten analysiert und aus der Analyse die nächsten notwendigen Handlungen abgeleitet. Wiederholt sich ein solcher Ablauf in einem stetigen Kreislauf, so ist dies eine „rückgekoppelte Steuerung".

Das Prinzip läßt sich an einem Suppentopf auf dem Herd demonstrieren: Zum Garwerden muß die Suppe zwar sehr heiß sein, soll aber andererseits nicht überkochen. Das kann erreicht werden, indem man soviel Wärme zuführt, bis die Suppe brodelt – dann wird die Wärmezufuhr reduziert, bis es im Topf nur noch leicht köchelt. Wann immer die Suppe überzukochen droht, wird die Kochplatte kleiner gestellt. In diesem Fall überwachen wir den Zustand der Suppe mit den Augen, analysieren das Gesehene und führen danach die jeweils notwendige Handlung aus. Damit fahren wir fort, bis die Suppe fertig ist. Microprozessoren würden diese Aufgabe ähnlich, wenn auch nicht genau gleich bewerkstelligen – insbesondere den augenblicklichen Zustand der Suppe müßte der Prozessor anders als wir überwachen. Für uns genügt einfaches Hinschauen – zusammen mit unserer Erfahrung erkennen wir, ob die Suppe kocht oder nicht. Der Microprozessor kann jedoch nur mit physikalisch Messbarem umgehen, etwa mit einer Temperaturangabe. Um das Kochen der Suppe per Computer zu steuern, wären noch Experimente zur höchstmöglichen Temperatur nötig, bei der die Suppe noch nicht überkocht. Ist dieser Wert bekannt, könnte der Prozessor uns die Arbeit abnehmen – vorausgesetzt, daß geeignete Apparaturen angeschlossen sind.

Mit dem Ausgangsbuffer und dem Niedervolt-Ausgang können Motoren auf verschiedene Weise gesteuert werden. Zum Ausprobieren der neuen Software eignet sich ein Lego- oder Spielzeugmotor besonders gut. Wichtig ist nur, daß die angeschlossene Spannung die zulässige Eingangsspannung des Motors keinesfalls überschreitet. Wenn Sie die Motoranschlüsse auf Kanal 0 des Niedervolt-Ausganges legen, läßt sich der Motor mit der „Z"-Taste ein- und mit „X" ausschalten.

Acorn B

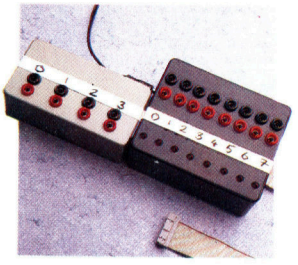
```
10 REM BBC SIMPLE MOTOR
20 DDR=&FE62:DATREG=&FE60
30 ?DDR=255:REM ALL LINES OUTPUT
40 ?DATREG=0:REM MOTOR OFF
50 REPEAT
60 A$=INKEY$(1):REM KEYPRESS?
70 IF A$="Z" THEN ?DATREG=1:REM TURN ON
80 UNTIL A$="X"
90 ?DATREG=0:REM TURN OFF
```

Commodore 64

```
10 REM CBM64 SIMPLE MOTOR
20 DDR=56579:DATREG=56577
30 POKEDDR,255: REM ALL LINES OUTPUT
40 POKEDATREG,0:REM MOTOR OFF
50 GETA$:IFA$<>"Z" ANDA$<>"X" THEN
   50:REM AWAIT KEY
60 IF A$="Z" THEN POKEDATREG,1:GOTO50
70 IF A$="X" THEN POKEDATREG,0:END
```

Übungen

Durch die Ein- und Ausgabemöglichkeiten unserer beiden Zusatzgeräte können sie Steuerungsaufgaben sehr elegant lösen. Für die im folgenden Experiment erwähnten Sensoren können Sie entweder Druckschalter oder Reed-Relais verwenden, die durch kleine Magnete betätigt werden. Sie sind in jedem Elektronik-Geschäft preiswert zu haben. Das gilt auch für wärmeempfindliche Schalter (Thermostate), die einen inneren Kontakt beim Erreichen einer festgelegten Temperatur schließen.
Schreiben Sie ein Programm, das ...
1) ein Fahrzeug zwischen zwei in seinem Weg liegenden Sensoren hin- und herfahren läßt,
2) ein Fahrzeug direkt auf einem Sensor anhalten läßt, wobei es auch zurücksetzt, falls nötig,
3) eine Tasse Wasser auf der Temperatur zwischen 40°C und 70° C hält (12-V-Tauchsieder und 2 Thermoschalter),
4) die Geschwindigkeit eines Fahrzeuges zwischen zwei Punkten in Metern/Sekunde ausgibt. (Sie müssen dazu die Streckenlänge und die Fahrzeit messen.)



Der Anschlußbuffer – das erste Werk unseres Selbstbau-Kurses – schützt die Schaltung des Computers vor zu großen Ein- und Ausgangsströmen. Seine unabhängige Stromversorgung speist auch die Buffer-Erweiterung. Damit lassen sich 12-Volt-Geräte über Software kontrollieren und schalten.

Wenn Sie den Motor wie unten im Bild an zwei benachbarte Buchsen des Niedervolt-Ausganges anschließen, kann auch seine Drehrichtung umgeschaltet werden. Dazu wird ein einfacher Ein/Ausschalter mit Anschluß 7 des Ausgangsbuffers verbunden.

Mit den folgenden Programmen fließt bei einer Eins im Datenregister der Strom in der einen, bei einer Zwei in der entgegengesetzten Richtung durch den Motor. Dabei wird wiederholt geprüft, ob Anschluß 7 durch Schließen des Schalters auf „Low" liegt – nur dann wird die Eins im Register durch eine Zwei ersetzt. Die Schaltstellung wirkt sich also auf die Drehrichtung aus – ein einfaches Beispiel für eine rückgekoppelte Steuerung.

Acorn B

```
10 REM BBC DIRECTED MOTORS
20 DDR=&FE62:DATREG=&FE60
30 ?DDR=127:REM LINE 7 INPUT
40 ?DATREG=0:REM TURN OFF
50 A$=GET$:REM AWAIT KEYPRESS
60 REPEAT
70 A$=INKEY$(1)
80 IF (?DATREG AND 128)=0 THEN DIR=2 ELSE DIR=1
90 ?DATREG=DIR
100 UNTIL A$="X":REM PRESS X TO END
110 ?DATREG=0:REM TURN OFF
```

Commodore 64

```
10 REM CBM64 DIRECTED MOTORS
20 DDR=56579:DATREG=56577
30 POKEDDR,127:REM LINE 7 INPUT
40 POKEDATREG,0:REM ALL OFF
50 GETA$:IFA$=" "THEN50:REM AWAIT KEYPRESS
60 GETA$
70 IF (PEEK(DATREG)AND128)=0THENPOKEDATREG,
   2:GOTO90
80 POKEDATREG,1
90 IFA$<>"X"THEN60
100 POKEDATREG,0:REM TURN OFF
```

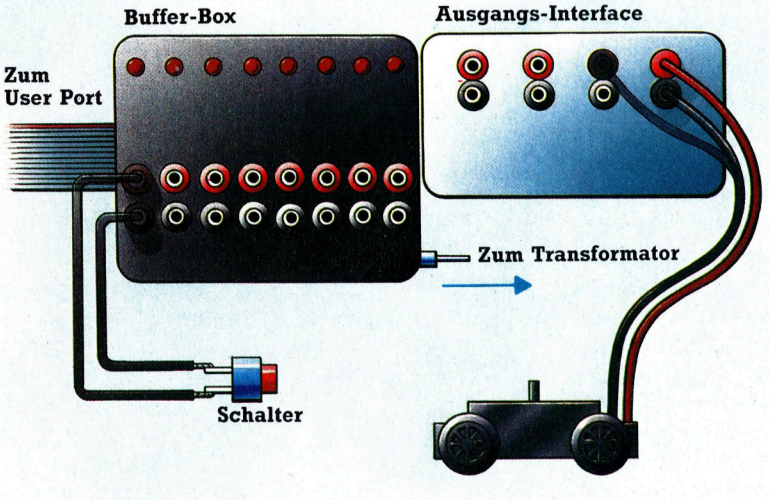
Sie können jedoch nicht nur die Drehrichtung, sondern auch die Geschwindigkeit des Motors über den Niedervolt-Ausgang steuern. Dazu brauchen Sie keine so komplizierten Geräte wie einen Digital/Analogwandler. – Einfacher geht es durch eine gepulste Spannung, die den Motor in schneller Folge ein- und ausschaltet. Der Motor wird sich dabei nahezu kontinuierlich drehen, wenn der Schaltvorgang schnell genug vor sich geht. Die Länge der Pausen zwischen den einzelnen Stromstößen bestimmt dann die Drehgeschwindigkeit. Das Programm für die gepulste Steuerung wiederholt dazu eine Struktur, in die eine Verzögerungsschleife eingegliedert ist.

Acorn B

```
10 REM BBC VARIABLE MOTOR CONTROL
20 DDR=&FE62:DATREG=&FE60:SPEED=30
30 ?DDR=255:REM ALL OUTPUT
40 ?DATREG=0:REM ALL OFF
50 A$=GET$;REM AWAIT KEYPRESS
60 REPEAT
70 A$=INKEY$(1)
80 ?DATREG=0:REM TURN OFF
90 FORI=1TO(100-SPEED):NEXT:REM DELAY1
100 ?DATREG=1:REM TURN ON
110 FORI=1TO SPEED:NEXT:REM DELAY 2
120 IF A$="D"THEN SPEED=SPEED-5
130 IF A$="Z"THEN SPEED=SPEED+5
140 UNTIL A$="X"
150 ?DATREG=0:REM TURN OFF
```

Commodore 64

```
10 REM CBM64 VARIABLE MOTOR CONTROL
20 DDR=56579:DATREG=56577:SPEED=30
30 POKEDDR,255:REM ALL LINES OUTPUT
40 POKEDATREG,0:REM TURN OFF
50 GETA$:IFA$=" "THEN50:REM AWAIT KEY
60 GETA$
70 POKEDATREG=0:REM TURN OFF
80 FORI=1TO(100-SPEED):NEXT:REM DELAY1
```

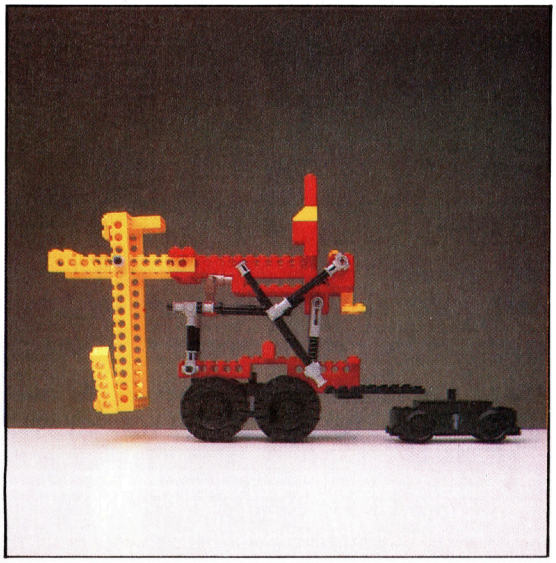


Allradantrieb
Die beiden Zusatzgeräte haben einen gemeinsamen Datenbus und teilen über den Minicon-Anschluß auch die Stromversorgung. Verfügt jedes Gerät über eine Buchse und einen Stekker, lassen sie sich im „Huckepack-Verfahren" zusammenschließen.

Der 12-Volt-Gleichstrommotor des Autos wird von der Spannung der Ausgangsbuchsen angetrieben. Der Schalter auf Bit 7 des Ausgangsbuffers steuert gemeinsam mit dem Programm, ob das Fahrzeug vor- oder rückwärts fährt.

```
90 POKEDATREG=1:REM TURN ON
100 FORI=1TOSPEED:NEXT:REM DELAY2
110 IFA$="D"THENSPEED=SPEED-5
120 IFA$="Z"THENSPEED=SPEED+5
130 IFA$<>"X"THEN60
140 POKEDATREG,0:REM TURN OFF
```

In diesem Programm steuert die Variable SPEED die Länge der Verzögerungsschleifen. Dabei wird die eine Verzögerungsschleife länger, wenn die andere kürzer ist und umgekehrt. DELAY 1 gibt die Zeit an, in welcher der Motor keinen Strom erhält. Während Delay 2 ist die Spannung eingeschaltet. Hat SPEED einen großen Wert, sind DELAY 1 kurz und DELAY 2 lang, wodurch sich der Motor schneller dreht. Den pulsierenden Strom können Sie am Flakkern der LEDs von Anschluß 1 gut erkennen.



Mit kleinen Autos auf dem Fußboden herumfahren – das mag Ihnen vielleicht als ein etwas zu geringer Lohn für große Mühen und Kosten erscheinen. Aber gerade der Bau einfacher Geräte und ihre Programmierung geben einen guten Einstieg in die Welt der Elektronik.

Lösung der Übungen

1) Die Sensoren liegen auf Anschluß 6 und 7, und der Motor ist an die positiven Kontakte 0 und 1 angeschlossen:

```
10 REM BBC VERSION 3.1
20 DDR=&FE62:DATREG=&FE60
30 ?DDR=63:REM LINES 6&7 INPUT
40 forward=1:reverse=2
50 ?DATREG=forward
60 FORI=1TO2000:NEXT:REM DELAY
70 REPEAT UNTIL(?DATREG AND192)<>192
80 ?DATREG=reverse
90 FORI=1TO2000:NEXT REM DELAY
100 IF(?DATREG AND192)<>192 THEN50
    ELSE GOTO100
```

```
10 REM CBM 64 VERSION 3.1
20 DDR=56579:DATREG=56577
30 POKEDDR,63:REM LINES 6&7 INPUT
40 FW=1:RV=2
50 POKE DATREG,FW
60 FORI=1TO1000:NEXT:REM DELAY
70 IF(PEEK(DATREG)AND192)=192THEN70
80 POKE DATREG,RV
90 FORI=1TO1000:NEXT:REM DELAY
100 IF(PEEK(DATREG)AND192)<>192THEN50
110 GOTO100
```

3) Der 40-Grad-Sensor liegt auf Anschluß 6, der 70-Grad-Sensor auf Anschluß 7 und der Tauchsieder auf Anschluß 0:

```
10 REM BBC VERSION 3.3
20 DDR=&FE62:DATREG=&FE60
30 ?DDR=63:REM LINES 6&7 INPUT
40 REPEAT
50 A$=INKEY$(1)
60 ?DATREG=1:REM TURN ON HEATER
70 REPEAT
80 UNTIL(?DATREG AND192)=0:REM 70 DEG
90 ?DATREG=0:REM HEATER OFF
100 REPEAT UNTIL(?DATREG AND192)=192
110 UNTIL A$()"":REM KEYPRESS
```

```
10 REM CBM 64 VERSION 3.3
20 DDR=56579:DATREG=56577
30 POKE DRR,63:REM LINES 6&7 INPUT
40 GET A$
50 ?DATREG=1:REM TURN HEATER ON
60 IF(PEEK(DATREG)AND192)<>0THEN60
70 POKE DATREG,0:REM HEATER OFF
80 IF(PEEK(DATREG)AND192)<>192THEN80
90 IF A$=""THEN40
```

2) Der Sensor liegt auf Anschluß 7, der Motor ist zwischen Anschluß 0 und 1 geschaltet:

```
10 REM BBC VERSION 3.2
20 DDR=&FE62:DATREG=&FE60
30 ?DDR=127:REM LINE 7 INPUT
40 speed=30:forward=1:reverse=2
50 direction=forward
60 REPEAT
70 ?DATREG=0:REM OFF
80 FORI=1TO(100-speed):NEXT
90 ?DATREG=direction
100 FORI=1TOspeed:NEXT
110 UNTIL(?DATREG AND128)=0:REM SWITCH
120 FORI=1TO1000:NEXT:REM DELAY
130 REM TEST FOR OVER PAD
140 IF(?DATREG AND128)=0THEN?DATREG=0:
    END
150 REM REVERSE SLOWLY
160 speed=2:direction=reverse:GOTO60
```

```
10 REM CBM 64 VERSION 3.2
20 DDR=56579:DATREG=56577
30 POKE DDR,127:REM LINE 7 INPUT
40 SP=30:FW=1:RV=2
50 DR=FW
60 POKE DATREG,0: REM OFF
70 FORI=1TO(100-SP):NEXT
80 POKE DATREG,DR
90 FORI=1TOSP:NEXT
100 IF(PEEK(DATREG)AND128)<>0THEN60
110 FORI=1TO1000:NEXT:REM DELAY
120 IF(PEEK(DATREG)AND128)=0THENPOKE
    DATREG,0:END
130 REM REVERSE BACK SLOWLY
140 SP=2:DR=RV:GOTO60
```

4) Schalter 1 liegt auf Anschluß 6, Schalter 2 auf Anschluß 7:

```
10 REM BBC VERSION 3.4
20 DDR=&FE62:DATREG=&FE60:DISTANCE=1
30 ?DDR=63
40 REPEAT UNTIL(?DATREG AND64)=0
50 ?DATREG=1
60 TIME=0:REM START TIMER
70 REPEAT UNTIL(?DATREG AND128)=0
80 PRINT"SPEED="DISTANCE/(TIME/100)
90 ?DATREG=0
```

```
10 REM CBM 64 VERSION 3.4
20 DDR=56579:DATREG=56577:DS=1
30 POKE DDR,63
40 IF(PEEK(DATREG)AND64)<>0THEN40
50 POKE DATREG,1
60 T=TI:REM START TIMER
70 IF(PEEK(DATREG)AND128)<>0THEN60
80 PRINT"SPEED="DS/((TI-T)/60)
90 POKE DATREG,0
```

Drachenfutter

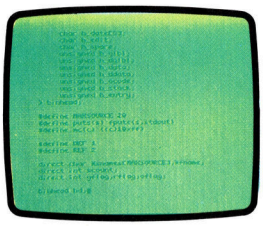
Bei dem Diskettensystem für den Dragon 32/64 setzte die Firma Dragon Data das bewährte Konzept benutzerfreundlicher Software in „Microsoft Advanced Color BASIC“ fort. Die Laufwerke werden über eine Parallelschnittstelle in Form eines Steckmoduls angeschlossen, in dem auch das ROM mit dem Betriebssystem untergebracht ist.

Die Dragon-Laufwerke arbeiten mit $5^1/_4$-Zoll-Disketten und einfacher Aufzeichnungsdichte. Bei der Formatierung werden 40 Spuren mit 18 Sektoren zu je 256 Byte angelegt, so daß sich eine Kapazität von 40x18x256=184320 Byte= 180 KByte ergibt. Nach Abzug des Platzbedarfs für Dateikennsätze, Dateiverzeichnis und Organisation bleiben etwa 175 KByte pro Diskettenseite nutzbar.

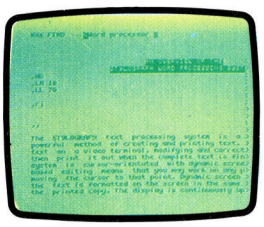
Das Dateiverzeichnis (Directory) enthält alle auf der Diskette stehenden Dateien (Files). Die Dateinamen (mit Typangabe) und der freie Speicherplatz werden auf Wunsch dargestellt. Es gibt vier Datei-Typen (BASIC-Programm, Daten, Binärcode, Duplikat), die durch ein .BAS, .DAT, .BIN bzw. .BAK hinter dem Dateinamen gekennzeichnet sind. Das Disketten-Betriebssystem (DOS) gibt keine Auskunft über die Anordnung der Dateien auf der Diskette.

Merkwürdigerweise fehlt beim Dragon-DOS eine Routine für den Direktzugriff auf einzelne Bytes. So etwas ist nützlich, wenn wie bei einem Datenbanksystem häufig kleine Datenmengen angefordert werden. Stattdessen bietet das DOS einen simulierten Direktzugriff („Simulated Random Access“), der mit ein paar zusätzlichen BASIC-Zeilen genau dasselbe leistet. Das Dragon-DOS ergänzt durch seinen Aufbau die BASIC-Programmierung in hervorragender Weise.

Wegen des hohen Anschaffungspreises und der geringen Diskettenkapazität ergibt sich nur ein mäßiges Preis/Leistungs-Verhältnis. Das hat der Hersteller wohl selbst gemerkt, denn inzwischen sind Laufwerke für doppelseitige Double-Density-Disketten mit 80 Spuren zu einem vernünftigen Preis vorgesehen. Für den Dragon 64 ist auf Wunsch ein Betriebssystem erhältlich, das von Diskette in das RAM geladen wird und dort allerdings 16 KByte belegt. Dieses „OS9“ (Operating System für den 6809-Prozessor) beinhaltet ein sehr leistungsfähiges DOS. Die Sprachen PASCAL, C, COBOL und BASIC sind auf Disketten verfügbar.

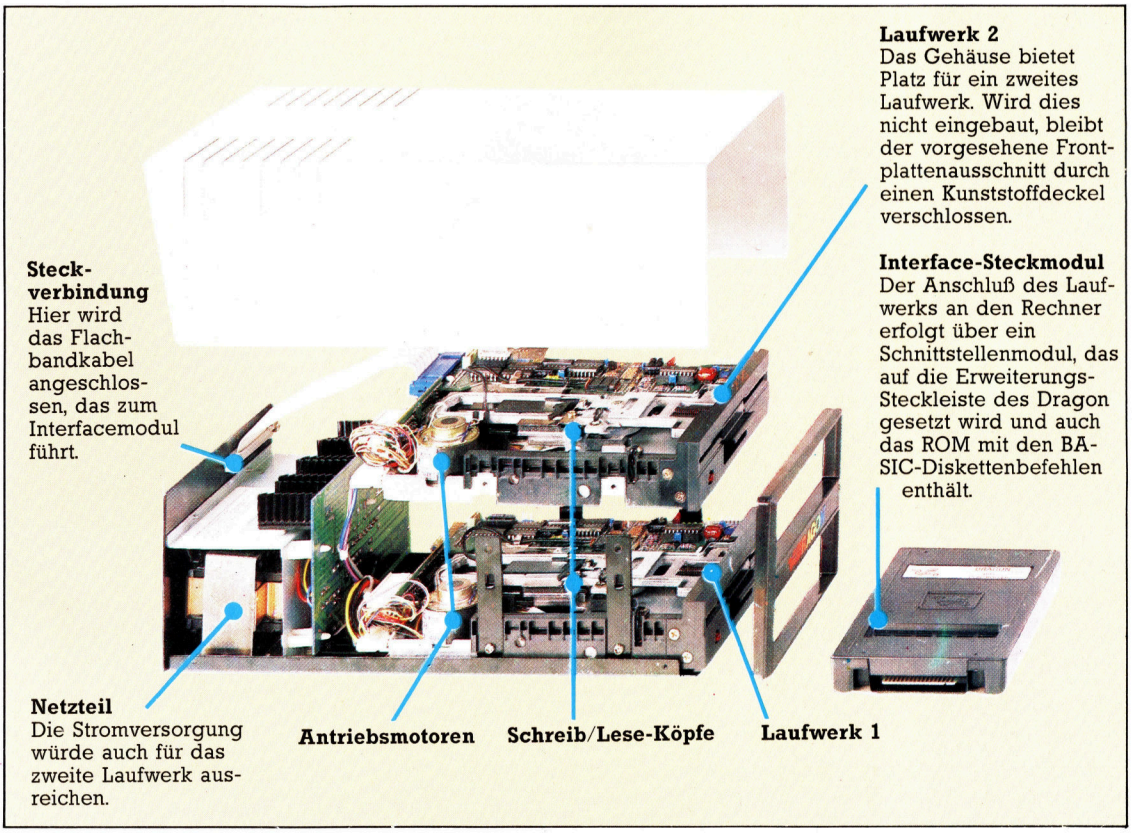


Dieses Diskettensoftware-Paket arbeitet mit dem Betriebssystem OS9. Es beinhaltet unter anderem einen Pascal- und einen C-Compiler.



Unter OS9 läuft auch kommerzielle Software. Dieses Textverarbeitungssystem kann anhand eines Verzeichnisses von 42000 Wörtern die Rechtschreibung überprüfen und bietet die Möglichkeit, Serienbriefe abzufassen.

Das Gehäuse der Dragon-Diskettenstation hat ein eigenes Netzteil und Platz genug für den Einbau eines zweiten Laufwerks. Über ein Flachbandkabel wird das Gerät mit dem Interface-Steckmodul am Rechner verbunden. Das Laufwerk arbeitet mit einem von Dragon Data entwickelten DOS oder mit der kommerziellen (und entsprechend teureren) OS9-Software. Jede Diskette bietet eine Datenkapazität von 175 KByte.



Dragon-Diskettenbefehle

Bei vielen Befehlen werden immer die gleichen Datei-Parameter benötigt, meist in der Form

BEFEHL"D:DATEINAME.TYP"

Hier wird durch D (Drive) das Laufwerk (1–4) festgelegt und die Standardeinstellung aufgehoben. Der DATEINAME darf bis zu acht Buchstaben enthalten. Mit .TYP kann die Art der Daten definiert werden. Wird darauf verzichtet, setzt das DOS selbst .BAS ein. Der obige Parametersatz wird im folgenden als FSP („File Specification Parameter") bezeichnet.

DRIVE N
definiert die Standardeinstellung. N kann Werte von 1-4 annehmen.

DSKINIT
bewirkt die Formatierung einer Diskette, allgemein in der Form:

DSKINIT D,S,T

D bezeichnet wieder das Laufwerk, S die zu formatierende Seite der Diskette (1 oder 2, ohne Angabe automatisch 1), T (Tracks) ist die gewünschte Anzahl von Spuren (40 oder 80, standardmäßig 40). Wenn nur ein Laufwerk vorhanden ist, genügt Eingabe von DSKINIT (ENTER) zum ordnungsgemäßen Formatieren der Diskette.

DIR D
listet die Dateinamen (mit Typ und Länge) auf der Diskette im Laufwerk D beispielsweise wie folgt:

```
DIR
PROGNAME .BAS 1654
CODEPROG .BIN 1389
PROGDATA .DAT 2581
PROGNAME .BAK 1654
167322 FREE BYTES
```

Programm-Dateien

SAVE
dient zur Übertragung von BASIC- oder Maschinenprogrammen auf Diskette. SAVE FSP speichert ein BASIC-Programm (ohne .BAS, da hier die Standardeinstellung verwendet wird). Der Befehl

SAVE FSP SSSS,EEEE,XXXX

speichert eine Binärdatei auf Diskette. Dabei ist SSSS die dezimale RAM-Anfangsadresse, EEEE die Endadresse und XXXX die Startadresse des Maschinenprogramms.

LOAD
Das Einlesen von BASIC- oder Binärdateien erfolgt mit dem Befehl

LOAD FSP

Wenn es sich um ein Maschinenprogramm handelt, kann FSP noch durch ,SSSS ergänzt werden, um die neue Anfangsadresse im Arbeitsspeicher festzulegen.

RUN FSP
bewirkt das Laden und sofortige Starten des angegebenen BASIC-Programms.

CHAIN FSP
dient zum Laden und Starten eines Programms ohne Veränderung der vorher abgelegten Variablen. Das Hinzufügen von ,SSSS hat dieselbe Wirkung wie beim LOAD-Befehl.

FREE D
gibt die beim Laufwerk D noch verfügbare Speicherkapazität an.

COPY
erzeugt von der Datei OLDFSP ein Duplikat namens NEWFSP, Befehlsform:

COPY OLDFSP TO NEWFSP

Wenn keine Drive-Nummern angegeben werden, wird auf dem Standard-Laufwerk kopiert.

RENAME
dient zum Austausch von Dateinamen beim gleichen Laufwerk (Standardeinstellung, wenn Angaben fehlen).

MERGE FSP
verbindet das spezifizierte BASIC-Programm auf Diskette mit einem anderen im Arbeitsspeicher. Belegen beide Programme dieselbe Zeilennummer, wird der Arbeitsspeicher überschrieben.

KILL FSP
löscht die angegebene Datei auf der Diskette.

PROTECT
bewirkt eine Software-Sperre gegen Löschen oder Überschreiben einer Datei (außer bei Disketten-Neuformatierung mit DSKINIT) durch

PROTECT ON FSP

Bei Ausgabe der Directory erscheint mit dem Dateinamen dann ein inverses P-Feld. Der Befehl PROTECT OFF FSP hebt die Sperre auf.

BACKUP
kopiert die Diskette im Laufwerk (DA) vollständig auf eine andere im Laufwerk (DB), wenn Sie schreiben:

BACKUP DA TO DB,S,T

S und T sind wie bei DSKINIT definiert, so daß Duplikate bei verschiedenen Disketten- und Laufwerkformaten möglich sind. Bei nur einem Laufwerk genügt die Eingabe BACKUP (ENTER), daraufhin wird auf dem Bildschirm angezeigt, wie Quell- und Zieldiskette zu wechseln sind.

VERIFY
wird durch ON und OFF in bzw. außer Kraft gesetzt und prüft automatisch diskettengespeicherte Dateien auf Übereinstimmung mit dem Original.

Datenfiles

FWRITE
dient zum Eröffnen und Auffüllen einer Datei mit Variablenlisten. Bei jedem FWRITE wird die angesprochene Datei über einen logischen Kanal erreicht, der die weitere Auffüllung der Datei zuläßt. Die Schreibweise dieses Befehls:

FWRITE"DATEINAME";VAR

DATEINAME bezeichnet die zu beschreibende (bzw. zu eröffnende) Datei und VAR die Liste, deren Elemente gespeichert werden sollen. Strings gelten durch Komma oder Semikolon als innerhalb eines Datenfiles getrennt, wenn sie nicht mit FLREAD gelesen werden. Durch

FWRITE"DATEINAME",FROM SB,FOR TB;VAR

wird die Liste VAR unter DATEINAME (beginnend mit dem Start-Byte SB) eingetragen, wobei TB ihre Länge (in „Total Bytes") angibt. Maximal können über FWRITE zehn Kanäle gleichzeitig offengehalten werden.

CLOSE D
schließt beim Laufwerk D wieder die mit FWRITE, FREAD oder FLREAD eröffneten Kanäle.

CREATE"DATEINAME",FL
legt unter DATEINAME ein Datenfile mit FL (File Length) Bytes an.

FREAD
ist wie FWRITE aufgebaut. Die Liste VAR wird ganz bzw. (wie im zweiten Beispiel bei FWRITE) ab Byte SB eingelesen, der Lesezeiger wird um (SB + TB) Bytes weitergestellt.

FLREAD
entspricht FREAD, Komma und Semikolon gelten jedoch nicht als Trennsymbol.

EOF
wird beim Lesen zum Kenntlichmachen des letzten Elements verwendet, zum Beispiel:

EP=EOF("DATEINAME")

Hier ist EP=0, solange der Lesezeiger noch nicht auf das letzte Datenfeld zeigt, und springt erst beim Ende auf Eins.

LOC"DATEINAME"
gibt die Einstellung des Lesezeigers als Nummer des nächsten Bytes an, das zu lesen ist.

SWRITE
speichert Daten auf Spur T im Sektor S, etwa die Strings A$ und B$ mit höchstens je 128 Byte. Der Befehl lautet hier:

SWRITE D,T,S,A$,B$

SREAD
ermöglicht es, die abgespeicherten Daten wiederzufinden. A$ und B$ können dabei neu benannt werden.

Alles Schiebung

In dieser Folge sehen wir uns die Probleme der Subtraktion und Multiplikation genauer an und stellen eine neue Klasse von Befehlen vor: die Op-codes für Shift und Rotate.

Der Z80 und der 6502 unterstützen beide den SBC-Befehl (Subtrahieren mit Übertrag), führen ihn jedoch völlig verschieden aus. Der 6502 benutzt das Übertragsflag, um ein „Borgen" anzuzeigen, das dem Übertrag der Addition entspricht. In der Assemblersprache des Z80 funktioniert SBC genau wie der Befehl ADC, wobei das Übertragsflag jedoch je nach Ergebnis gesetzt oder auf Null gestellt wird.

Wenn Sie mit ADC $E4 auf $5F addieren (nachdem das Übertragsflag zuvor auf Null gesetzt wurde), erscheint im Akkumulator die Summe $43 mit gesetztem Flag; das vollständige Ergebnis ist daher $0143. Das Übertragsflag wurde gesetzt, da der Akkumulator allein das vollständige Ergebnis nicht speichern kann.

Wenn Sie auf dem Z80 nun wiederum das Übertragsflag löschen und dann $E4 von $5F subtrahieren, steht im Akkumulator als Ergebnis $7B mit gesetztem Flag. Bei einer erneuten Addition von $7B auf $E4 (wiederum nachdem das Übertragsflag gelöscht wurde) enthält der Akkumulator als Ergebnis $5F mit gesetztem Flag. Die folgenden Zeilen zeigen, wie der Vorgang abläuft:

$5F — $E4 = $7B Übertragsflag gesetzt
$5F = $E4 + $7B Übertragsflag gesetzt

Wenn Sie das Flag als Indikator eines negativen Ergebnisses nehmen, läßt sich $7B als Zweierkomplement verstehen:

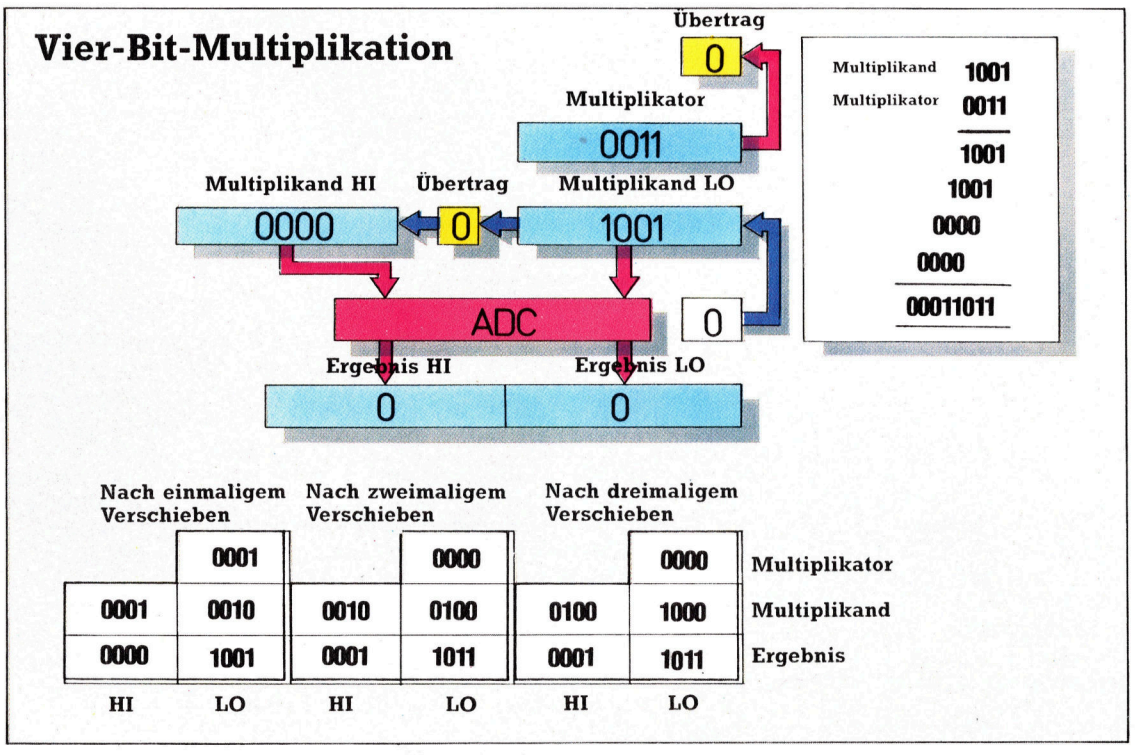
$7B binär	= 01111011
minus Eins	—1
ergibt das	
Einerkomplement Umkehrung	01111011
ergibt das Zweierkomplement	10000100 = $85

$5F — $E4 sollte daher die negative Zahl —$85 ergeben. Überprüfen Sie das Ergebnis im Dezimalsystem:

$5F =	95 dezimal
—$E4 =	228 dezimal
$85 =	—133 dezimal

So weit, so gut. Nehmen Sie nun an, daß die Subtraktion eigentlich eine Zwei-Byte-Zahl betraf: $375F — $21E4.

HI	LO		
$37	5F	=	14175 dezimal
—$21	E4	= —	8676 dezimal
$15	7B	=	5499 dezimal



Das Beispiel zeigt aus Gründen der Einfachheit eine Vier-Bit-Multiplikation – die Anzahl der Bits hat keinen Einfluß auf den Algorithmus. Dabei wird deutlich, wie der Inhalt der einzelnen Bits des Multiplikanden – Eins oder Null – das Ergebnis aus der Addition von Nullen oder den verschobenen Versionen des Multiplikanden entstehen lassen. Die Bits des Multiplikators werden nach rechts über das Übertragsflag verschoben, während die Multiplikandenbits durch eine Linksrotation vom niederwertigen Byte durch das Übertragsflag in das höherwertige Byte wandern.

Bei der Subtraktion des niederwertigen Bytes ist das Ergebnis $7B mit gesetztem Übertragsflag. Dieser Übertrag wird dann von dem SBC-Befehl auf die Zahl $21 addiert, die dadurch $22 wird. Von $37 subtrahiert ergibt sich $15.

Die Zwei-Byte-Arithmetik des Z80 besteht daher aus diesem einfachen Ablauf:
1) Übertragsflag löschen.
2) Die niederwertigen Bytes mit Übertrag subtrahieren.
3) Die höherwertigen Bytes mit Übertrag subtrahieren.

Die Version des 6502 arbeitet schon im ersten Teilvorgang anders: Das Übertragsflag muß gesetzt sein, damit die niederwertigen Bytes von den höherwertigen Bytes „borgen" können. Ist kein „Borgen" notwendig, läuft die Subtraktion normal ab, wobei das Übertragsflag für die Subtraktion der höherwertigen Bytes bestehen bleibt. Tritt in der Subtraktion des niederwertigen Bytes jedoch ein Unterlauf auf, übernimmt das Übertragsflag die Funktion des „neunten" Akkumulatorbytes. Nachdem auf diese Weise sichergestellt wurde, daß das richtige Ergebnis auftritt, wird das Flag gelöscht. Auf dem 6502 erzeugt die Subtraktion höherwertiger Bytes mit gelöschtem Übertragsflag das gleiche Ergebnis wie eine Z80-Subtraktion mit gesetztem Übertragsbyte, da die Zahl, die subtrahiert werden soll, vor dem Subtraktionsvorgang dekrementiert wird. Beide Methoden, einen Unterlauf abzufangen, gehen auf die alten mathematischen Methoden des „Borgens" und „Übertragens" zurück. Doch sehen wir uns die Version des 6502 einmal genauer an.

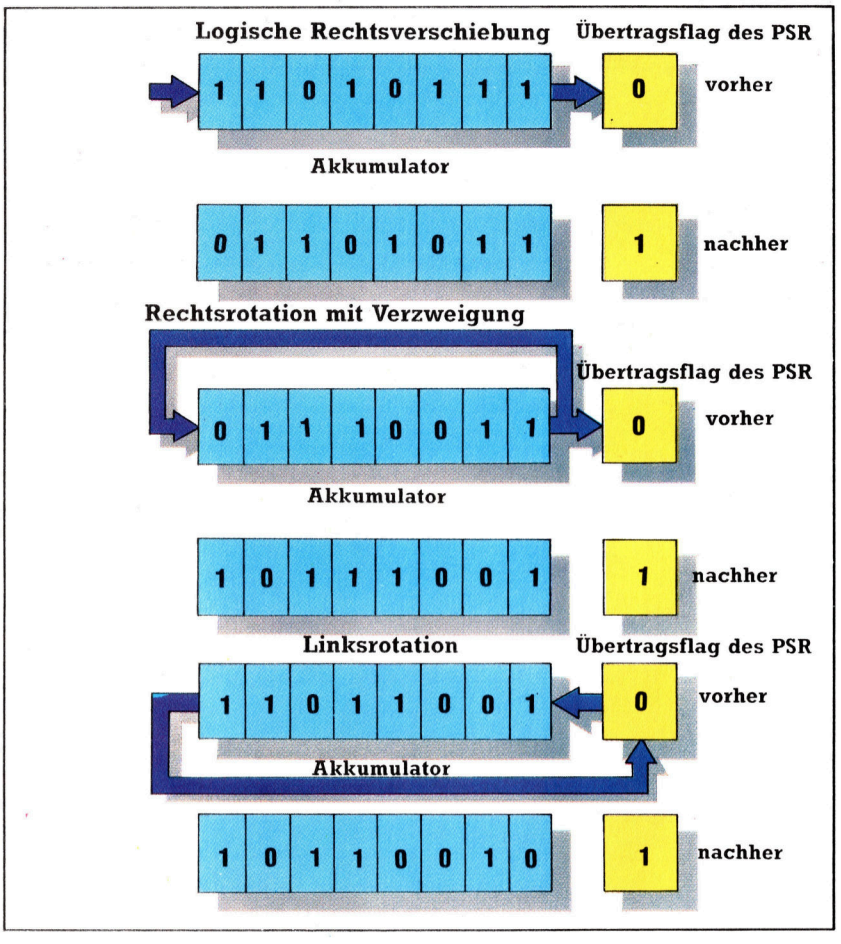
Wenn das Übertragsflag gelöscht ist und wir $E4 von $5F subtrahiert haben, steht im Akkumulator als Ergebnis $7A, wobei das Flag nicht gesetzt ist. Das Beispiel des Z80 zeigte, daß die Berechnung $7B ergab, wobei das gesetzte Übertragsflag eine negative Zahl anzeigte. $7B ist das Zweierkomplement des „echten" Ergebnisses (-$85). $7A muß daher das Einerkomplement dieser Zahl sein und der Status des Übertragsflags eine Art Schalter des Akkumulators. Bei gesetztem Flag ergibt sich das Zweierkomplement, ist es nicht gesetzt, das Einerkomplement.

Wird die Subtraktion auf dem 6502 mit gesetztem Übertragsflag durchgeführt, enthält der Akkumulator $7B, und das Flag ist dadurch gelöscht.

Multiplikationen

Untersuchen wir eine normale dezimale Multiplikation:

```
 174    Multiplikand
*209    Multiplikator
 348    Erstes Teilergebnis
 000    Zweites Teilergebnis
1566    Drittes Teilergebnis
36366   Endergebnis
```



Der Ablauf läßt sich leicht nachvollziehen, ohne daß Sie verstehen müssen, auf welchen mathematischen Prinzipien die Schreibweise mit unterschiedlichen Positionen beruht. Die Methode besteht darin, jedes Teilergebnis gegenüber dem vorigen Teilergebnis um genau eine Stelle nach rechts zu rücken (damit der Vorgang anschaulicher wird, wurden die leeren Stellen nicht mit Nullen aufgefüllt).

Viele Menschen haben Schwierigkeiten mit dem Zusammenspiel der versetzten Teilergebnisse und dem Einmaleins. Die binäre Multiplikation ist einfacher. Sie kennt nur einen Rechenvorgang: Eins mal Eins. Alle anderen Ein-Byte-Ergebnisse sind Null. Sehen Sie sich folgende binäre Multiplikation an:

```
1101     = dezimal 13
 *1001   = dezimal 9
1101     Erstes Teilergebnis
0000     Zweites Teilergebnis
0000     Drittes Teilergebnis
1101     Viertes Teilergebnis
1110101  = 117 dezimal
```

In diesem Beispiel wird nicht nur die Verschiebung der Teilergebnisse deutlich, sondern auch die bestechende Einfachheit der binären Multiplikation. Ein Teilergebnis kann nur Null oder der verschobene Multiplikand sein, je nachdem ob das entsprechende Bit des Multiplikators Eins oder Null ist. Diese Struktur ist

Die Befehle für Verschieben und Rotation werden hauptsächlich für die bitweise Untersuchung eines Registers eingesetzt. Bei jeder Rotation wird das höchste oder das niedrigste Bit in das Übertragsflag des PSR gestellt. Der Status des Übertragsflags kann dann von einem Verzweigungsbefehl abgefragt werden und so den Programmablauf steuern. Bei den Rotationsbefehlen bleibt der Inhalt des Registers erhalten, während die logische Verschiebung die leeren Stellen mit Nullen füllt. Eine Linksverschiebung multipliziert die Register daher mit zwei, während die Rechtsverschiebung sie durch zwei dividiert.

Acht-Bit-Multiplikation

6502			Z80		
	ORG	$C100		ORG	$D000
START	LDA	#$00	START	LD	BC,(MPR)
	STA	PRODLO		LD	B,$08
	STA	PRODHI		LD	DE,(MPDLO)
	STA	MPDHI		LD	D,$00
	LDX	#8		LD	HL,$00
	CLC		LOOP0	SRL	C
LOOP0	ROR	MPR		JR	NC,CONT0
	BCC	CONT0		CALL	ADDIT
	JSR	ADDIT	CONT0	SLA	E
CONT0	ASL	MPDLO	ENDLP0	DJNZ	LOOP0
	ROL	MPDHI		LD	PRODLO
	DEX			RTS	
ENDLP0	BNE	LOOP0	MPR	DB	$E2
	RTS		MPDLO	DB	$7A
MPR	DB	$E2	MPDHI	DB	$00
MPDLO	DB	$7A	PRODLO	DW	$0000
MPDHI	DB	$00	ADDIT	ADD	HL,DE
PRODLO	DB	$00		RET	
PRODHI	DB	$00			
ADDIT	CLC				
	LDA	PRODLO			
	ADC	MPDLO			
	STA	PRODLO			
	LDA	PRODHI			
	ADC	MPDHI			
	STA	PRODHI			
	RTS				

den Tests ähnlich, denen wir in den Steuermechanismen der Assemblersprache schon begegnet sind. In der binären Multiplikation muß nacheinander jedes Bit des Multiplikators untersucht werden – ist es Null, wird Null addiert; ist es Eins, wird der versetzte Multiplikand auf die Gesamtsumme addiert. Wie können nun die einzelnen Bits des Multiplikators untersucht werden, und wie läßt sich der Multiplikand versetzen?

Der Zustand eines bestimmten Bits innerhalb eines Bytes läßt sich mit dem Befehl BIT feststellen, den es im Instruktionssatz beider Prozessoren gibt. Auf dem Z80 enthält dieser Befehl eine Adresse und eine Bitzahl als Operand. Dabei wird das Nullflag gesetzt, wenn das entsprechende Bit Null ist, und gelöscht, wenn das Bit Eins ist. Der BIT-Befehl des 6502 enthält nur die Adresse. Der Adresseninhalt wird über AND mit dem Akkumulator verknüpft, und das Nullflag wird gesetzt oder nicht gesetzt, je nachdem ob das Ergebnis wahr oder falsch ist.

Die Instruktionssätze des Z80 und des 6502 besitzen dafür eine Reihe von Verschiebungs- und Rotationsbefehlen, wobei die des Z80 komplexer sind. Generell verschieben sie jedes Registerbit um eine Position nach rechts oder links. Die Unterschiede der Prozessoren liegen in der Behandlung des letzten Registerbits – das Bit, das sich aus dem Register heraus bewegt, während am anderen Ende ein Bit eingefügt wird. Die Verschiebung von Bit 7 aus dem Register und das Einsetzen in Bit 0 nennt man Linksrotation. Wird Bit 0 auf Bit 7 verschoben, spricht man von einer Rechtsrotation. Der übrige Inhalt des Registers wird dabei entsprechend verändert, wobei jedoch keine neuen Werte eingefügt werden. Nach acht Rotationen ist der ursprüngliche Zustand des Registers wiederhergestellt.

Ohne Rotation sind eine Bestimmungsadresse für das herausgeschobene Bit und eine Quelle für das eingesetzte Bit notwendig. Beide Funktionen werden oft von den Flags des Prozessor-Status-Registers und speziell von dem Übertragsflag übernommen. Für den Aufbau einer Multiplikationsroutine für zwei Ein-Byte-Zahlen muß der Multiplikand nach links und der Multiplikator nach rechts verschoben werden. Dabei werden die Bits des Multiplikanden in das höherwertige Multiplikandenbyte verschoben und die unbelegten Bits mit Nullen gefüllt. Die Multiplikatorbits müssen dann zum Testen über die PSR-Flags geleitet werden. Den Zustand dieser und der neu eingeschobenen Multiplikatorbits braucht man nur dann zu beachten, wenn sie nochmals gebraucht werden. Bei dem Multiplikator ist für uns nur wichtig, ob das ausgeschobene Bit Eins oder Null ist.

In der linksstehenden Abbildung wurde der Multiplikator in der Adresse MPR untergebracht, der Multiplikand in MPDLO und das Ergebnis in PRODLO und PRODHI.

In diesem Beispiel wird deutlich, wieviel einfacher die 16-Bit-Register und der zugehörige Befehlssatz die Programmierung des Z80 gestalten. Sehen Sie sich dafür die ADDIT-Unterroutine beider Programme an. Der 6502 rotiert den Multiplikator mit ROR bitweise nach rechts in den Übertrag und benutzt ASL und ROL, um den Multiplikanden nach links aus MPDLO nach MPDHI und in den Übertrag zu schieben. Die Schleife wird von dem X-Register gesteuert, das als Zähler dient.

Die Z80-Version verschiebt den Multiplikator mit SRL nach rechts in den Übertrag und den Multiplikanden nach links über den Übertrag in das Registerpaar DE. Das Register B ist der Schleifenzähler. Interessant ist hierbei, daß der ADD-Befehl nicht nur eine 16-Bit-Arithmetik unterstützt, sondern anders als ADC nicht von dem Übertragsflag beeinflußt wird.

In der nächsten Folge untersuchen wir die Division und die Techniken, mit der die Bildschirmdarstellung gesteuert werden kann.

Übung

1) Schreiben Sie eine Unterroutine für die Multiplikation eines 16-Bit-Multiplikanden mit einem Acht-Bit-Multiplikator.

2) Die Multiplikation ist nur eine wiederholte Addition. Schreiben Sie eine Routine für eine Acht-Bit-mal-acht-Bit-Multiplikation, die keine Befehle für Verschiebung/Rotation enthält.

Branchenwechsel

Die Entwicklung der Tandy Corporation begann bereits vor über fünfzig Jahren – ein weiter Weg vom Lederlieferanten zum größten Computerhändler der Welt.

Die Tandy Corporation hat mit ihren Filialunternehmen Tandy und American Radio Shack eine Ladenkette von fast 400 Computerzentren und über 5500 Niederlassungen in 76 Ländern. Das Unternehmen besitzt 29 Fabriken, in denen die Produkte hergestellt werden, die unter den Markennamen Tandy und Radio Shack im Angebot sind.

Tandy wurde jedoch nicht als Elektronikfachgeschäft gegründet. Norton Hinckley und David Tandy begannen 1927 mit der Hinckley-Tandy Leather Company, einer Firma, die als Lederlieferant für Schuster in Beaumont, Texas, tätig war. Der erste geschäftliche Schritt in Richtung Elektronik-Gigant erfolgte 1963, als David Tandys Sohn Charles sich entschloß, das Unternehmen zu erweitern. Er erwarb Anteile an einer in Boston ansässigen Firma namens Radio Shack, die beträchtliche Probleme hatte.

Übernahme von Radio Shack

Seit Anfang der zwanziger Jahre war dieses Unternehmen auf den Vertrieb von elektronischen Bauteilen spezialisiert. Obwohl das Geschäft überwiegend auf dem Versandwege abgewickelt wurde und im Raum Boston insgesamt neun Niederlassungen existierten, machte Radio Shack große Verluste. 1967 war es Charles Tandy bereits gelungen, den Verlust von vier Millionen Dollar in einen Gewinn von 20 Millionen Dollar umzuwandeln.

Der nächste entscheidende Schritt erfolgte 1970 mit der Übernahme einer Ladenkette namens „Leonards", die Produkte der Unterhaltungselektronik verkaufte und Tandy den Einstieg in diesen Marktbereich ermöglichte. Im Tandy-Katalog von 1984 sind 2625 Artikel aufgeführt, die nichts mit Computern zu tun haben, vom Widerstand über HiFi-Bausteine bis hin zu Synthesizern. Dazu kommen 396 Computer-Artikel.

Die erste englische Tandy-Niederlassung wurde 1973 eröffnet. Das Haus erarbeitete sich rasch einen guten Ruf als Elektronikspezialist. 1978, als der „TRS-80 Modell I" eingeführt wurde, verfügte Tandy in England bereits über 120 Niederlassungen. 1983 war die Zahl der Filialen bereits auf 227 gewachsen.

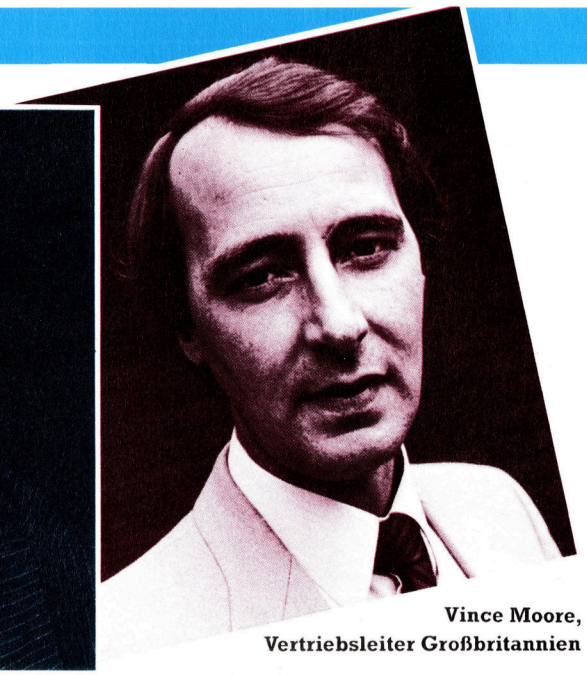
Mit dem „Modell I" verschaffte sich Tandy den Ruf eines führenden Computerherstellers. Dabei handelt es sich um einen Einplatinenrechner mit einem Z80-Microprozessor, vier K RAM-Speicher und einem Schwarzweißbildschirm mit niedrig auflösender Grafik.

Der Name Tandy ist längst zum Markenzeichen geworden. Damit sind drei Qualitätsbegriffe verbunden: Angebot, Service und Beratung. Unter dem Namen Radio Shack ist das Unternehmen auch in den USA bekannt.

Seitdem ist die Tandy-Computerproduktlinie ständig aktualisiert worden, wenngleich es nicht gelang, mit den Nachfolgemodellen an den Erfolg des „Modell I" anzuschließen. Das Unternehmen bietet inzwischen auch Geschäftscomputer an, so beispielsweise „Modell II" sowie „Modell 12" und „Modell 16". Bei den beiden letzteren handelt es sich um 16-Bit-Rechner. Das gilt auch für das neue, IBM PC-kompatible „Modell 2000". Inzwischen bietet Tandy das „Modell 4" an (und dazu eine tragbare Version, das „Modell 4P"). Beide sind besonders für den Geschäftsbereich geeignet und – das ist bei Microcomputern ungewöhnlich – kompatibel zu den Modellen I und III.

Mit dem „Tandy Color Computer", basierend auf dem 6809-Microprozessor, versuchte das Unternehmen, seine Position im Markt wieder-

John Sayers, Marketing-Direktor Großbritannien

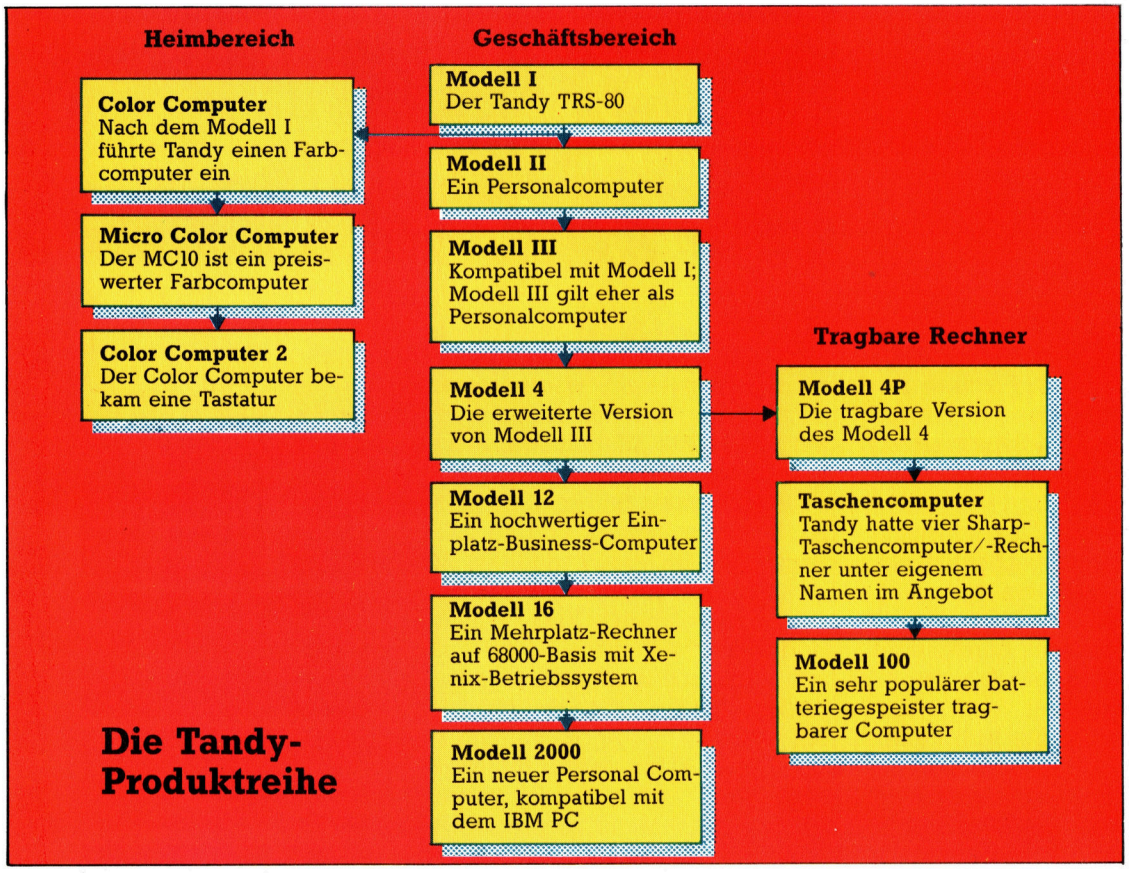


Vince Moore, Vertriebsleiter Großbritannien

zuerlangen. Dieser Rechner ist in vielen Teilen dem „Dragon" ähnlich. Eine weiterentwickelte, zugleich verkleinerte Version mit der Bezeichnung „Micro Color Computer" ist Indiz dafür, daß Tandy in diesem Bereich weiter arbeitet.

Die vielleicht interessanteste Produktlinie ist die der tragbaren Computer. Tandy verkaufte zunächst Taschencomputer, die den Sharp-Rechnern ähnlich waren. Das neue „Modell 100" resultierte aus der Zusammenarbeit mit dem japanischen Anbieter Kyocera und dem amerikanischen Software-Haus Microsoft. Das Ergebnis: Ein buchgroßer, batteriegespeister tragbarer Rechner mit integriertem BASIC, Kommunikations-Software und „Tagebuch".

Viele Computerunternehmen mußten 1984 große Verluste hinnehmen. Tandy dagegen arbeitet außerordentlich erfolgreich und bringt ständig neue Produkte heraus, die sowohl in Fort Worth, Texas, dem neuen Sitz der Gesellschaft, als auch in der TC Electronics Corporation, der japanischen Niederlassung von Tandy in Tokio, entwickelt werden. Die Tandy Corporation scheint auch langfristig einen festen Platz im Microcomputer-Geschäft beanspruchen zu können.



Die Übersicht zeigt die drei Geschäftsbereiche der Tandy-Rechner und ihre Entwicklung seit Beginn der Produktion.

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.

Data Processing = Datenverarbeitung
Früher hieß alles „elektronische Datenverarbeitung", was mit Computern zu tun hatte. Vor dem Einzug der Microprozessoren gab es neben Großrechnern nur sehr teure kleinere Anlagen. Diese waren für Privatleute ungeeignet, unter anderem deshalb, weil sie ein bestimmtes Raumklima und ständigen Service benötigten. Die Großrechner wurden durch geschultes Personal bedient, und ohne Genehmigung hatte niemand Zugang zur Maschine.

Die Microcomputer-Revolution fegte die ganze Exklusivität hinweg. Der Micro stellt keine besonderen Ansprüche an Aufstellungsort und Bedienung und kann als „Arbeitsplatzrechner" auf dem Schreibtisch jedes Angestellten ständig für alle möglichen Zwecke bereitstehen. In großen Firmen gibt es daneben weiterhin ein richtiges Rechenzentrum, das für übergreifende Aufgaben wie Finanzplanung, Lagerverwaltung, Buchführung, Produktionsabwicklung und den Personalbereich mit Lohn- und Gehaltsabrechnung zuständig ist.

Debugging = Fehlerbeseitigung
Debugging ist die Beseitigung von Störungen im weitesten Sinn, seien es nun Programmfehler oder Defekte in der Hardware. Im engeren Sinn wird das Wort für das systematische Testen und Korrigieren von Software verwendet.

Der Begriff ist von Grace Hopper geprägt worden, die 1945 bei der Entwicklung des Relais-Rechners „Harvard Mark II" mitarbeitete. Als dieser plötzlich seine Dienste verweigerte, suchte man lange nach der Ursache – diese zeigte sich dann auch, in Form eines Nachtfalters (bug), der sich zwischen den Relais verklemmt hatte. Seither ist das Wort „Bug" für Fehler jeglicher Art in Gebrauch, auch für Softwarefehler.

Decision Table = Entscheidungstabelle
Die Flexibilität eines Rechners hängt wesentlich davon ab, wie Entscheidungen realisiert werden können. Das einfachste Beispiel ist die IF... THEN – Anweisung in BASIC, die ein bestimmtes Vorgehen veranlaßt, wenn die definierten Bedingungen erfüllt sind.

In einer Entscheidungstabelle sind die Bedingungen und die möglichen Verfahrensweisen systematisch zusammengestellt.

In BASIC ist der Aufbau einer Entscheidungstabelle, außer in Form einer Serie von IF... THEN-Anweisungen, eine komplizierte Sache. Leichter realisierbar ist die Tabelle mit ON... GOTO-Anweisungen, wobei die Bedingungen durch Zahlen gekennzeichnet werden und die zugehörigen Vorgehensweisen über Sprungbefehle angesteuert werden.

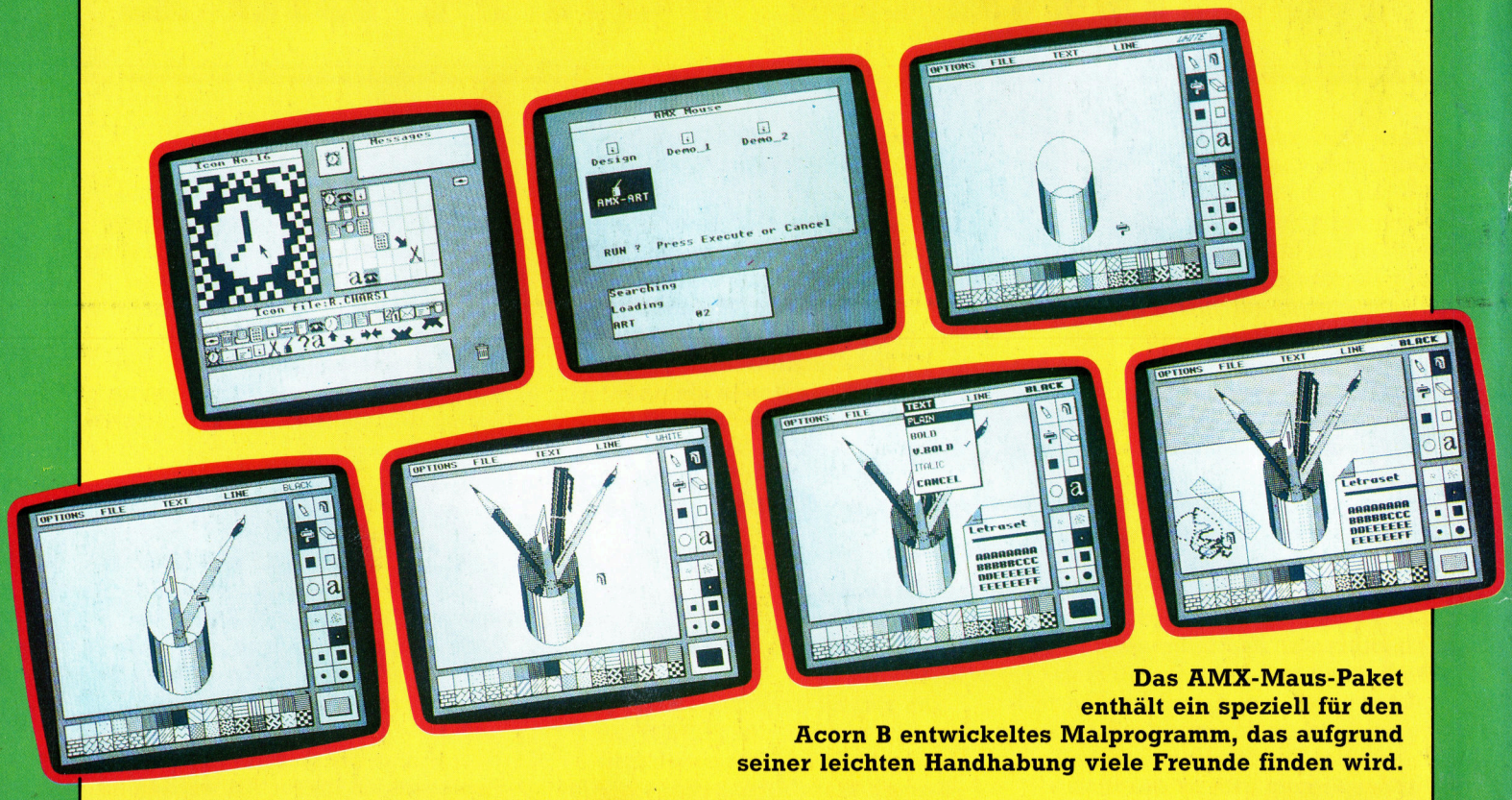
Decision Tree = Entscheidungsbaum
Bei einem Entscheidungsbaum liegen Bedingungen und zugeordnete Vorgehensweisen in anderer Struktur vor als bei der Entscheidungstabelle. Der Rechner geht nicht schrittweise die ganze Bedingungsliste durch, sondern wird von einer Eingangsliste über Verzweigungen an immer detailliertere Teillisten weiterdirigiert.

Declaration Statement = Vereinbarung
In BASIC müssen alle Variablenfelder vor Gebrauch DIMensioniert werden. Einige andere Sprachen verlangen eine ähnliche Spezifizierung für sämtliche Variablen. Der Benutzer muß definieren, ob es Integer- oder Real-, Double-Precision- oder String-Variable sein sollen. Alle Anweisungen dieser Art werden als Vereinbarungen bezeichnet.

Die Vereinbarung von Feldern dient eigentlich nur zur Verbesserung der Leistungsfähigkeit. DIM-Anweisungen veranlassen den BASIC-Interpreter, im RAM einen Bereich von der Größe der Felder zu reservieren. Gewöhnliche Variable werden im Anschluß daran gespeichert. Ohne vorherige Dimensionierung müßte das Betriebssystem beim Auftreten eines neuen, höheren Feldindex jedesmal alle folgenden Variablen im RAM weiterschieben, um Platz zu schaffen.

Auch in BASIC ist es zweckmäßig, eine Vereinbarung aller Variablen zu simulieren, indem man sie in den ersten Programmzeilen in der Reihenfolge der Häufigkeit ihrer Benutzung einmal erscheinen läßt. So können die notwendige Rechenzeit und der benötigte Speicherplatz wesentlich verringert werden.

Bildnachweise
897: Liz Heaney, Pilot Software
903, 908, 915, 918, 919, 924: Ian McKinnell
905, 912, 913, 914: Liz Dixon
907: Ian McKinnell, Microscope
909, 910, 911: Chris Stevens
916: Kevin Jones
917: Ian McKinnell, Dave Whelan



Das AMX-Maus-Paket enthält ein speziell für den Acorn B entwickeltes Malprogramm, das aufgrund seiner leichten Handhabung viele Freunde finden wird.

computer kurs

Heft 34

Raster-Fahndung

Computer machen gezielte Fahndungen in der Polizeiarbeit leichter. Doch bergen Rechnersysteme auch Gefahren für den Bürger – ist der „Überwachungsstaat" nahe?

Doppelte Kraft

Im vorherigen Kurs wurde die Steuerung eines einmotorigen Modellautos beschrieben. Wie man dagegen zwei Motoren per Rechner steuert, erklärt diese Folge.

Kompakt verpackt

Auf dem Markt der Personal-Computer dominieren die IBM-kompatiblen Systeme. Wir beschreiben den Compaq Plus.

Karnaugh-Tafeln

Zur Vereinfachung lassen sich „Karnaugh-Tafeln" einsetzen: Sie stellen logische Ausdrücke dar.

Handel im All

Das Spiel „Elite" verfügt über dreidimensionale Grafik, viel Action und Spannung. Für Commodore und Acorn.