Einsteigen - Verstehen - Beherrschen

DM 3,80 öS 30 sfr 3,80

computer kurs

Ein wöchentliches Sammelwerk

Heft 61

Marktübersicht Heimcomputer

Moderne Sprachen

Robotarm unter Spannung

CP/M: Befehlsgewalt

computer kurs

Heft 61

Inhalt

WIE SIE JEDE WOCHE IHR HEFT BEKOMMEN

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

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

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

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

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

Abonnement

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

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

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

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

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

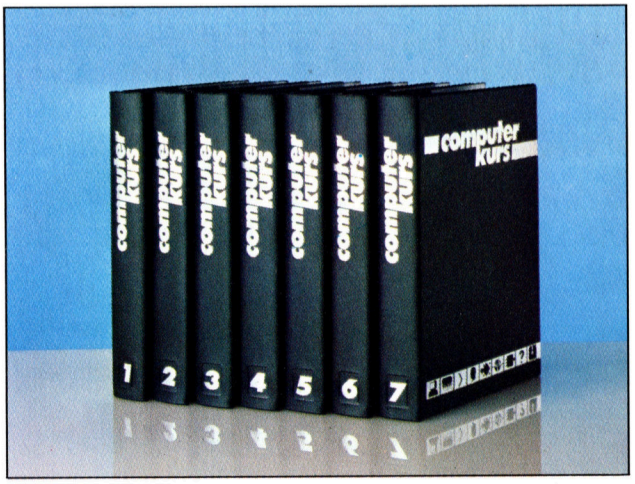
Schweiz: Der Sammelordner kostet sfr 15. Bitte wenden Sie sich an Ihren Kiosk; dort werden Sie jederzeit die gewünschten Exemplare erhalten.

INHALTSVERZEICHNIS

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

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

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



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

Einkaufsbummel

Weil der Markt bei sinkenden Preisen ständig neue und bessere Rechner anbietet, ersetzen viele Leute ihren Heimcomputer schon nach kurzer Zeit durch ein Nachfolgemodell. Wir möchten hier ein paar konkrete Hinweise zu den Stärken und Schwächen der gängigen Maschinen geben.

Die Anschaffung eines Heimcomputers sollte eigentlich unproblematisch sein, aber das weitgefächerte Angebot und die schwierige Bewertung der technischen Information läßt die Sache für manchen doch mehr zum Glücksspiel werden. Jedenfalls empfiehlt sich eine Bestandsaufnahme Ihrer Bedürfnisse, bevor Sie sich zum Kauf entschließen. Versuchen Sie, schon vorher die erforderlichen Spezifikationen, vielleicht sogar den Rechnertyp festzulegen, und lassen Sie sich nicht vom äußeren Eindruck leiten.

Vor der Anschaffung sollten Sie sich unbedingt genau über Ihre Ansprüche im klaren sein: Wollen Sie einen ausbaufähigen Rechner haben, der später mit mehr Speicherplatz und Peripherie ausgerüstet werden soll, oder suchen Sie nur eine kurzfristige Lösung, um sie später durch etwas Neues zu ersetzen?

Bei den meisten aktuellen Modellen wird die Überlegenheit den Vorläufern gegenüber stark herausgestrichen – mehr Speicherplatz, besseres BASIC, höhere Grafikauflösung und eingebaute ROM-Programme. Die älteren Rechner bieten aber, vor allem wenn sie schon in größeren Stückzahlen verkauft sind, einen entscheidenden Vorteil: Es gibt dafür reichlich Software. Viele Käufer von Neuentwicklungen müssen oft monatelang warten, bis die Programme und die Peripherie angeboten werden.

Anfangsschwächen

Besonders intensiv haben sich die Softwarehäuser um den Sinclair Spectrum, den Commodore 64 und den Atari XL gekümmert. Der ZX Spectrum ist geradezu ein Paradebeispiel dafür, wie mit ausgefeilter, allerdings oft nur kompliziert über England zu beziehender Software die angeborenen Schwächen eines Rechners ausgeglichen werden können. Trotzdem hätte wohl keiner der drei erwähnten Rechner in der ursprünglichen Form heute noch einen erfolgreichen Start zu verzeichnen: Beim Spectrum ist die Tastatur dürftig, und das Commodore-BASIC läßt Befehle vermissen, die zur vollen Nutzung seiner Möglichkeiten unumgänglich sind.

Bei den meisten neuen Micros ist es umgekehrt – sie haben eindrucksvolle Spezifikationen, aber nicht viel Software zu bieten. Der Käufer eines solchen Computers muß darauf hoffen, daß die Maschine sich durchsetzt und damit auch die Softwareproduzenten für sich gewinnt. Hardwaremäßig gibt es heute sehr viel mehr fürs Geld: vollwertige Tastatur, großer Arbeitsspeicher (64 oder 12" KByte) und gute Grafikmöglichkeiten gelten als selbstverständlich bei allen angebotenen Microcomputern.

Ein anderer aktueller Trend sind kostenlose Software-Beigaben. Zum Sinclair QL gehören vier solcher Pakete – Textverarbeitung, Tabellenkalkulation, Datenbank und Geschäftsgrafik. Der Commodore Plus/4 wurde mit entsprechender ROM-Software ausgestattet, die allerdings weniger durchdacht und ohne Diskettenlaufwerk auch kaum richtig anzuwenden ist. Bei anderen Rechnern liegt das Gewicht auf Spielen. Beim Commodore 16 werden vier Programme mitgeliefert, und selbst der betagte Sinclair Spectrum wird stellenweise mit einem Sechserpack von Spielen angeboten.

Beim Kauf eines neuen Rechners ist aber noch mehr zu berücksichtigen: Die Maschinen unterscheiden sich erheblich in der Ausbaufähigkeit, soweit es Diskettenlaufwerke, Drucker, Modems und die sonstige Peripherie betrifft. Bei einigen Modellen paßt standardisiertes Zubehör, während bei anderen Spezialschnittstellen erforderlich sind, was die Auswahl für den Benutzer natürlich stark einschränkt. Unerläßlich ist auch eine gute Anleitung – in manchen Fällen ist sie mangelhaft und bietet dem Anwender keinerlei Hilfe. Der Käufer sollte auch darauf achten, wo der Schwerpunkt bei der Software liegt; beim Acorn B überwiegen zum Beispiel die Lernprogramme, während es für den ZX Spectrum mehr Spiele gibt.

Schneider CPC 464

Der CPC 464 stammt von dem britischen Hersteller Amstrad, dessen große Erfahrung im aktuellen HiFi-Geschäft die kompakte Gestaltung dieses Heimcomputer-Erstlings mitbestimmten. Das preisgünstige Komplettangebot mit Monitor und integriertem Cassettenrecorder spricht nicht nur den Anfänger an. Joystick- und Centronics-Schnittstellen sind eingebaut. Statt Cassette ist beim 664 und 612" ein Diskettenlaufwerk eingebaut, das als Paket mit einem CP/M-Betriebssystem und der Sprache LOGO ausgeliefert wird. Der CPC 464 ist ein in Fachkreisen anerkannter Allround-Rechner.

Sinclair QL

Der QL wird standardmäßig mit 12" KByte Speicher, eingebautem Doppel-Microdrive und vier kommerziellen Programmen geliefert. Er bietet rein äußerlich viel fürs Geld, hat aber seine Eigenheiten. Trotz Verbesserungen gegenüber der früheren Ausführung fehlt der Tastatur immer noch die Professionalität, und in dem strukturierten BASIC sind zwar anfängliche Fehler beseitigt, aber es arbeitet angesichts des schnellen 32-Bit-Prozessors erstaunlich langsam. Die Langzeit-Zuverlässigkeit der Microdrives ist noch zweifelhaft, und die Editiermöglichkeiten sind nicht zufriedenstellend.

Bei Grafikbetrieb beträgt die Auflösung vierfarbig 512 × 256 oder achtfarbig 256 × 256 Punkte. Die Farbwahl kann für jedes Pixel unabhängig erfolgen, so daß der Bildschirm allein 32 KByte belegt. Ein Disketten- oder Cassetten-Interface fehlt, dagegen sind Joystick-, Monitor-, Netzwerk- und zwei RS232-Schnittstellen vorhanden. Als wesentliches Argument für die Anschaffung kann durchaus die beigepackte Software (von Psion) gelten; die Textverarbeitungs-, Kalkulations-, Dateiverwaltungs- und Geschäftsgrafik-Programme sind, verglichen mit der üblichen Heimcomputer-Software, äußerst aufwendig. Der gute Eindruck wird jedoch hardwaremäßig, vor allem durch die langsamen Microdrives, getrübt.

Sinclair Spectrum

Trotz seiner begrenzten technischen Möglichkeiten hat sich der ZX Spectrum sehr erfolgreich entwickelt. Die dürftige Tastatur des Erstmodells wurde beim Spectrum Plus entscheidend verbessert. Die Eingabe der BASIC-Befehle erfolgt über vordefinierte Tasten, was Anfängern die Sache erleichtert, für Fortgeschrittene aber doch Probleme mit sich bringt. Die Grafikauflösung beträgt 256 × 192 Pixel bei acht Farben (jeweils zwei pro Zeichenfeld). Die Tonerzeugung ist monophon und etwas leise. Das BASIC ist ordentlich, wenn auch etwas langsam. Genormte Schnittstellen fehlen beim Spectrum, jedoch gibt es von Drittfirmen eine Menge Peripherie, die sich an den User Port hängen läßt. Sinclair selbst offeriert ein „Interface 1" für Microdrives, Vernetzung und RS232-Zubehör, außerdem ein „Interface 2", das die Verwendung von Cartridge-Software gestattet. Vor allem in der Plus-Version ist der Spectrum aufgrund der breiten, allerdings nicht immer leicht erhältlichen, Softwarebasis unverändert interessant – er wird heute durchweg mit 4" K RAM angeboten.

Schwierige Entscheidung
In den Diagrammen sind die oben erläuterten Rechner nach einheitlichen Kriterien bewertet, um vergleichbare Profile zu erhalten.

Acorn B

In England wird der Einsatz des Acorn B in Schulen seitens der Regierung gefördert: daher gibt es dazu eine ganze Menge Lernprogramme. Der Acorn B hat ausgezeichnete technische Daten aufzuweisen: schnelles strukturiertes BASIC, hervorragende Grafikauflösung und guten Sound sowie ungewöhnlich viele Schnittstellen – Centronics, RS423, RGB, Composite Video, vier A/D-Kanäle, User Port, 1-MHz-Buserweiterung und die „Tube", über die ein zweiter Prozessor angekoppelt werden kann. Außerdem sind Diskettenbetrieb und Netzwerkverbund vorgesehen. Die 32 K RAM der Grundausstattung werden je nach Bildschirmmodus bis auf neun KByte von der Grafik belegt; deshalb sind mindestens 64 K zu empfehlen. Nachrüstbar ist auch ein zweiter Prozessor. Damit wird CP/M-Software verwendbar.

Acorn Electron

Als „kleiner Bruder" des Acorn B ist der Electron mit dem gleichen hervorragend strukturierten BASIC, aber nicht mit einem so vielseitigen Schnittstellenangebot ausgestattet. Das Electron-BASIC läuft langsamer als die B-Version, und der Electron unterstützt nicht den Acorn B-Bildschirmmodus „Teletext". Einiges von der Acorn B-Software ist Electron-kompatibel. Der Electron verfügt mit einer Auflösung von 640 × 256 Pixeln über eindrucksvolle Grafik-Fähigkeiten: das Textformat beträgt bis zu 32 Zeilen mit 80 Zeichen. Leider wird für den Bildschirmaufbau viel Speicherplatz benötigt, so daß von den 32 K RAM bei maximaler Auflösung nur neun KByte für den Benutzer frei bleiben.

Schneider CPC 464

Mangelhaft			Ausgezeichnet
			Preis
			Arbeitsspeicher
			Massenspeicher
			Tastatur
			BASIC-Grafik
			BASIC-Sound
			BASIC-Editor
			BASIC-Komfort
			Softwarequalität
			Softwareumfang
			Schnittstellen
			Monitorausgang

Sinclair QL

Mangelhaft			Ausgezeichnet
			Preis
			Arbeitsspeicher
			Massenspeicher
			Tastatur
			BASIC-Grafik
			BASIC-Sound
			BASIC-Editor
			BASIC-Komfort
			Softwarequalität
			Softwareumfang
			Schnittstellen
			Monitorausgang

Sinclair Spectrum

Mangelhaft			Ausgezeichnet
			Preis
			Arbeitsspeicher
			Massenspeicher
			Tastatur
			BASIC-Grafik
			BASIC-Sound
			BASIC-Editor
			BASIC-Komfort
			Softwarequalität
			Softwareumfang
			Schnittstellen
			Monitorausgang

Acorn B

Mangelhaft			Ausgezeichnet
			Preis
			Arbeitsspeicher
			Massenspeicher
			Tastatur
			BASIC-Grafik
			BASIC-Sound
			BASIC-Editor
			BASIC-Komfort
			Softwarequalität
			Softwareumfang
			Schnittstellen
			Monitorausgang

Acorn Electron

Mangelhaft			Ausgezeichnet
			Preis
			Arbeitsspeicher
			Massenspeicher
			Tastatur
			BASIC-Grafik
			BASIC-Sound
			BASIC-Editor
			BASIC-Komfort
			Softwarequalität
			Softwareumfang
			Schnittstellen
			Monitorausgang

Commodore Plus/4

Dieser Rechner war als Nachfolger für den Commodore 64 gedacht. Er hat eine Grafikauflösung von 320 × 200 Pixeln und 64 KByte RAM, verfügt über 121 Farbtöne und ein verbessertes BASIC, das dem Benutzer eine weitergehende Beeinflussung der Bildschirmdarstellung ermöglicht. Bei höchster Grafikauflösung sind für jedes Zeichenfeld zwei, bei 160 × 200 Pixeln vier Farben vorgesehen. Die Tonerzeugung (max. zweistimmig) erreicht nicht das Niveau des C 64, aber ihre Programmierung ist durch das verbesserte BASIC merklich erleichtert. Auf die spezielle Sprite-Grafik des C 64 wurde hier verzichtet, dafür gibt es einen Maschinensprache-Monitor und vier kleine ROM-residente Anwenderprogramme (deshalb „Plus/4") für Textverarbeitung, Tabellenkalkulation, Dateiverwaltung und Grafik, die ohne Diskettenlaufwerk allerdings nur begrenzt nutzbar sind. Der Plus/4 braucht einen speziellen Cassettenrecorder und spezielle Joysticks; dagegen passen die Drucker und das Diskettenlaufwerk vom C 64 auch hier.

MSX-Standard-Computer

MSX ist ein „Minimalstandard", und die meisten Hersteller bieten eine wesentliche darüber hinausgehende Ausstattung an. Abgebildet ist hier der HX-10 von Toshiba. MSX spezifiziert ein besonders wirkungsvolles BASIC mit komfortabler Grafik und Tonerzeugung, speziellen Interrupt-Routinen für die Behandlung von „Ereignissen" und einem guten Editor. Die MSX-Rechner haben Funktionstasten, die mit zehn verschiedenen Kommandos belegbar sind. Die Grafik-Auflösung beträgt 256 × 192 Pixel bei 16 Farben, außerdem sind 32 Sprites vorgesehen.

Commodore 64

Für diesen populären Rechner gibt es eine Fülle von Software. Leider ist er nur mit einem mäßigen BASIC ausgestattet, das komfortable Befehle für die volle Nutzung der ausgezeichneten Sound- und Grafik-Möglichkeiten vermissen läßt. Die Maximalauflösung beträgt 320 × 200 Pixel bei insgesamt 16 Farben, und es lassen sich Sprites programmieren. Von den 64 K RAM stehen dem Benutzer 39 K zur Verfügung. Sie brauchen einen speziellen Cassettenrecorder und die Commodore-eigenen Drucker und Diskettenlaufwerke – alles nicht teuer, aber die Floppy ist als langsam und störanfällig bekannt.

Atari 600/800 XL

Diese Rechner sind verbesserte Versionen der 400/800er Serie, daher gibt es eine Menge Software. Der Atari 600 XL ist ein Auslaufmodell (Speicherkapazität 16 K); mehr zu empfehlen ist der im übrigen gleich gebaute 800 XL mit 64 K RAM. Die maximale Grafikauflösung beträgt 320 × 192 Pixel, allerdings nur zweifarbig; bei geringerer Auflösung sind 16 Farben in 16 Helligkeitsstufen darstellbar. Außerordentlich vielseitig sind die Klangerzeugung und die Sprite-Grafik, während das übrige BASIC nicht ganz dem letzten Stand entspricht. Die Atari-Rechner brauchen einen speziellen Cassettenrecorder. Störender ist, daß ein ausführliches Handbuch nicht zur Maschine gehört, sondern eigens dazugekauft werden muß. Die Atari-Peripherie ist nicht normgerecht, aber preiswert und überall erhältlich.

Commodore 16

Der Commodore 16 sollte den VC-20 ersetzen und wird daher auch als „Anfängerpaket" mit Cassettenrecorder und BASIC-Kurs auf Cassette angeboten. Das Gehäuse erinnert zwar an den VC-20 und den C 64, aber das Innenleben der Maschine entspricht eher dem teureren Plus/4, so daß auch dessen Maschinensprache-Monitor und das gleiche BASIC zum Einsatz kommt. Im Normalbetrieb bleiben von den 16 K RAM noch 12 K für BASIC-Programme, bei höchster Grafikauflösung allerdings nur zwei KByte. Der Rechner wäre vor allem für Neulinge durchaus interessant, wurde aber, vor allem wegen der mangelhaften Softwareunterstützung, kein Markterfolg.

Preissenkung
In diesem Artikel haben wir auf Preisangaben verzichtet. Die Preise sind wieder sehr in Bewegung geraten. Sie sollten daher möglichst selbst genaue Informationen einholen.

Commodore Plus/4

Mangelhaft			Ausgezeichnet
			Preis
			Arbeitsspeicher
			Massenspeicher
			Tastatur
			BASIC-Grafik
			BASIC-Sound
			BASIC-Editor
			BASIC-Komfort
			Softwarequalität
			Softwareumfang
			Schnittstellen
			Monitorausgang

Commodore 64

Mangelhaft			Ausgezeichnet
			Preis
			Arbeitsspeicher
			Massenspeicher
			Tastatur
			BASIC-Grafik
			BASIC-Sound
			BASIC-Editor
			BASIC-Komfort
			Softwarequalität
			Softwareumfang
			Schnittstellen
			Monitorausgang

Commodore 16

Mangelhaft			Ausgezeichnet
			Preis
			Arbeitsspeicher
			Massenspeicher
			Tastatur
			BASIC-Grafik
			BASIC-Sound
			BASIC-Editor
			BASIC-Komfort
			Softwarequalität
			Softwareumfang
			Schnittstellen
			Monitorausgang

Atari 600/800 XL

Mangelhaft			Ausgezeichnet
			Preis
			Arbeitsspeicher
			Massenspeicher
			Tastatur
			BASIC-Grafik
			BASIC-Sound
			BASIC-Editor
			BASIC-Komfort
			Softwarequalität
			Softwareumfang
			Schnittstellen
			Monitorausgang

MSX-Standard-Rechner

Mangelhaft			Ausgezeichnet
			Preis
			Arbeitsspeicher
			Massenspeicher
			Tastatur
			BASIC-Grafik
			BASIC-Sound
			BASIC-Editor
			BASIC-Komfort
			Softwarequalität
			Softwareumfang
			Schnittstellen
			Monitorausgang

CP/M-Befehle

In unserer CP/M-Serie untersuchen wir die Diskettenbefehle, mit denen sich auf flexible Weise Peripheriegeräte ansprechen und Dateien organisieren lassen. Wir zeigen Schritt für Schritt, wie CP/M vom Einschalten des Computers an funktioniert.

In der ersten Folge haben wir uns die Entwicklung und den Aufbau von CP/M angesehen. Dabei stellten wir fest, daß sich einige CP/M-Befehle ständig im RAM befinden, während andere auf Diskette gespeichert sind und nur für ihren Einsatz in den Arbeitsspeicher geladen und danach wieder gelöscht werden. Diese „flüchtigen" Diskettenbefehle geben dem Anwender die Möglichkeit, Diskettenstationen anzusprechen, Dateien zu organisieren und weitere Peripheriegeräte zu steuern.

Gleich beim Einschalten des Systems führt der Computer Tests durch, mit denen er sicherstellt, daß alle erforderlichen Systemkomponenten angeschlossen sind. Der Prozessor aktiviert dann die Kanäle zu den Peripheriegeräten. Er sendet Codes, die ihre Funktionsfähigkeit prüfen und stellt fest, ob sie bereit sind, Daten zu empfangen. Dieser Vorgang heißt „Initialisierung".

Bei einigen Peripheriegeräten muß der Computer auf eine Rückmeldung des angesprochenen Gerätes warten, bevor er weitere Aufgaben erledigen kann.

Erweiterung	Kommentar	Beispiel
ASM	Quelldateien der Assemblersprache	CODEPROG.ASM
BAK	Sicherheitskopie einer Textdatei	MEMO.BAK
BAS	Quelldatei in BASIC	PROG.BAS
COM	Bezeichnung einer Befehlsdatei	PIP.COM
HEX	Hexadezimaldatei auf Maschinenebene	GRAPHIC.HEX
INT	Compiliertes BASIC-Programm	GAMES.INT
PRN	Datei für Programmlistings des Assemblers	CODEPROG.PRN
SUB	Befehlsdatei für BATCH-Abläufe	SPOOLER.SUB
$$$	Zwischendatei des Editors	PHONE.$$$

Die Initialisierung veranlaßt den Schreib-/Lesekopf der Diskettenstation, die erste Spur der CP/M Systemdiskette zu lesen. Ist keine Diskette vorhanden, dreht sich das Laufwerk bis eine Diskette eingelegt wird. Auf Spur Null ist der „Urlader" (Bootstrap Loader) gespeichert, der dem Computer mitteilt, wie er das CP/M-Programm laden soll. Existiert auf Spur Null kein Ladeprogramm, erscheint eine Fehlermeldung.

Nach dem Laden von CP/M erscheint auf dem Bildschirm das Symbol A> (auf manchen Maschinen OA>) und der Cursor. A> wird „Prompt" genannt und zeigt an, daß das System betriebsbereit ist. Der Prompt teilt uns weiterhin mit, welche Diskettenstation CP/M gerade anspricht: das aktuelle Laufwerk. Da CP/M gerade erst geladen wurde, sind wir auf Laufwerk A angemeldet.

Wenn nur ein Diskettenlaufwerk angeschlossen ist oder das zweite nicht gebraucht wird, reicht dies völlig aus. Wenn Sie aber Laufwerk B ansprechen wollen, müssen Sie CP/M dies mit dem Befehl B: und RETURN mitteilen. CP/M prüft dann, ob das Laufwerk angeschlossen ist und eine Diskette enthält. Ist dies der Fall, zeigt auf dem Bildschirm der Prompt B> an, daß nun Laufwerk B betriebsbereit ist. Mit CP/M lassen sich auf die gleiche Weise auch die Laufwerke C und D ansprechen.

Es werden zwar nur selten vier Laufwerke an ein System angeschlossen, doch arbeiten viele Computer mit doppelseitigen Laufwerken. Die Vorderseiten der Disketten werden als A und B bezeichnet und die Rückseiten als C und D.

Nach der Anmeldung in CP/M kann mit dem Befehl DIR oder dir eine Liste dargestellt werden, die anzeigt, welche Dateien auf der Diskette vorhanden sind. Dabei erscheinen auch die Befehle des CP/M-Betriebssystems.

Auf den ersten Blick erscheint es umständlich, erst eine Befehlsdatei laden und aktivieren zu müssen, um simple Statusinformation erhalten zu können. Warum werden die entsprechenden Befehle nicht schon beim Einschalten des Gerätes ins RAM geladen? Der Grund ist einfach: Abgesehen von der Platzeinsparung im RAM wird durch diese Methode die Kompatibilität zwischen unterschiedlichen Systemen erst möglich.

CP/M enthält eine Anzahl Befehle, mit denen sich Diskettendateien bearbeiten lassen. Obwohl CP/M-Anwendungen einen bestimmten Standard haben müssen, um auf unterschiedlichen Maschinentypen laufen zu können, haben viele Hersteller das System den Eigenschaften ihrer Computer angepaßt. Da es in CP/M keinen Standardbefehl zur Formatierung von Disketten gibt, fügt jeder Hersteller eine eigene Version an das System an.

In der letzten Folge erwähnten wir, daß der Name einer CP/M-Datei aus zwei Teilen besteht: Ein Punkt trennt den Hauptnamen von der Erweiterung, die bestimmt, nach welcher Methode die Datei in den Computer geladen wird. Die Befehlsdateien der CP/M-Systemdiskette haben beispielsweise die Erweiterung „.COM".

Dateien mit dieser Erweiterung werden automatisch gestartet.

Für das Laden und Ausführen von Befehlsdateien muß nur der Hauptname eingegeben und RETURN gedrückt werden.

Der Befehl STAT hat ähnliche Funktionen wie DIR. Er gibt ausführliche Informationen über die Diskette und die darauf gespeicherten Dateien. Bei der Eingabe von STAT und Return wird angezeigt, wieviel Platz auf der Diskette für neue Dateien frei ist, wieviel Platz die bestehenden Dateien belegen sowie die Schreib-/Lesemöglichkeiten. So bedeutet der Code R/W, daß die Diskette gelesen und beschrieben werden kann, R/A dagegen, daß sie schreibgeschützt ist – sie läßt sich also nur lesen, nicht aber beschreiben.

Dateien übermitteln

STAT selbst kann den Schreibschutz aktivieren. Nach Eingabe von STAT D:=R/O (D bedas Laufwerk) läßt sich die Diskette nur noch lesen. Jeder Versuch, jetzt Daten darauf zu speichern, erzeugt einen BDOS-Fehler. Mit STAT lassen sich auch die Namenserweiterungen und die Anzahl der logischen Diskettensektoren pro Datei anzeigen. Wird hinter STAT ein Dateiname angegeben, so erscheinen nur die Daten dieser Datei. Weiterhin kann STAT den Status der angeschlossenen Peripheriegeräte anzeigen und ändern. STAT DEV stellt eine Liste aller angeschlossenen Ein- und Ausgabegeräte dar.

Eine der wichtigsten Eigenschaften von Betriebssystemen ist ihre Fähigkeit, Dateien von einer Diskette zur anderen übertragen zu können. In CP/M führt PIP (Peripheral Interchange Program) diese Aufgabe aus. Das Programm kann jedoch weit mehr, als nur Dateien zu kopieren – mit ihm lassen sich Dateien auch an Drucker oder andere Ein- und Ausgabegeräte senden.

PIP wird durch die Eingabe von PIP und RETURN aufgerufen. Statt der Prompts erscheint nun ein Stern, der anzeigt, daß das Programm den nächsten Befehl erwartet. PIP hat folgendes Eingabeformat: D:ZIELDATEI =D: QUELLDATEI. Nehmen Sie an, Sie möchten die Datei HCAC.TXT von einer Diskette auf eine andere kopieren und dabei gleich den Namen in ARBEIT.TXT ändern. Zuerst tauschen Sie die Systemdiskette in Laufwerk A gegen die Diskette mit HCAC.TXT aus, die zweite Diskette legen Sie in Laufwerk B ein. Die Eingabe „B:ARBEIT.TXT=A:HCAC.TXT" kopiert die HCAC.TXT-Datei von Laufwerk A auf B und gibt: ihr die Bezeichnung ARBEIT.TXT.

Beachten Sie, daß PIP zuerst die Zieldatei erwartet und dann erst die Ursprungsdatei. Nach Abschluß des Kopiervorgangs können Sie mit DIR überprüfen, ob die Datei erfolgreich und richtig kopiert wurde.

Auch wenn Sie nur über eine Diskettenstation verfügen, können Sie mit PIP arbeiten – Ursprungs- und Ziellaufwerk ist dann A. Bei diesem Ablauf lädt CP/M eine Anzahl Sektoren der Ursprungsdatei und informiert Sie, wann Sie die Diskette wechseln sollen, damit die Zieldiskette beschrieben werden kann.

Mit PIP können Sie auch Dateien an den Drucker senden. Der Befehl lautet: PIP LPT:=B:HCAC.TXT. Dabei fällt auf, daß hinter PIP gleich der restliche Befehl eingegeben wurde. CP/M bietet diese Möglichkeit, damit Sie nicht den Ladevorgang abwarten müssen, bevor Sie weitere Befehle eingeben können.

In dem Befehl steht statt der Zieldatei der Code LPT (Line PrinTer). Er bezeichnet den Drucker. Die Laufwerksbezeichnung vor HCAC.TXT gibt an, wo die Datei zu finden ist. Wenn B das angemeldete Laufwerk ist, braucht diese Information allerdings nicht im Befehl zu erscheinen. Es gibt sogar CP/M-Versionen, in denen alle Laufwerksangaben weggelassen werden können, da CP/M alle verfügbaren Diskettenstationen durchsucht, bevor es die Fehlermeldung „Datei nicht gefunden" ausgibt.

In unserem Kopiervorgang hatten wir den Namen verändert. Dies ist bei PIP jedoch nicht nötig, da Dateien auch unter dem gleichen Namen kopiert werden können. Wenn Sie Dateien nur umbenennen, nicht aber kopieren wollen, können Sie den Befehl REN (RENAME) nehmen. Die Umbenennung von HCAC.TXT in ARBEIT.TXT hat folgendes Eingabeformat: REN HCAC.TXT=ARBEIT.TXT. Bedenken Sie, daß auch hier der Zielname zuerst eingegeben werden muß. Der Befehl REN erzeugt im Inhaltsverzeichnis der Diskette einen neuen Eintrag und löscht den alten Namen.

Auf ähnliche Weise lassen sich auch Dateien von einer Diskette löschen: ERA Dateiname und RETURN. ERA sollte – wie alle Löschbefehle – mit Umsicht eingesetzt werden, da bei fehlender oder falscher Laufwerksangabe schnell die falsche Datei gelöscht wird. Sie sollten es sich zur Gewohnheit machen, in Befehlen immer die Laufwerksbezeichnung anzugeben, selbst wenn es von CP/M nicht gefordert wird (etwa ERA B:HCAC.TXT).

In der nächsten Folge untersuchen wir weitere CP/M-Befehle, sehen uns die Steuerzeichen des Systems an und beschäftigen uns mit den Namenserweiterungen.

ASM – Der CP/M-Assembler

Die meisten CP/M-Inplementationen verfügen über ein eigenes Assemblermodul namens ASM. Obwohl von vielen Programmierern ignoriert, ist ASM ein außergewöhnlich flexibles Werkzeug, das nicht nur über die normalen Assemblerdirektiven verfügt, sondern auch die bedingte Assemblierung ermöglicht.

ASM wurde ursprünglich für den 8080-Prozessor entwickelt. Da diese CPU später durch den Z80 ersetzt wurde, akzeptiert der ASM nicht alle Standardbefehle des Z80. Das Buch „Programmierung des Z80" enthält eine Liste der Abweichungen. Wenn Sie den ASM ständig auf einer Z80-Maschine einsetzen wollen, empfehlen wir Ihnen jedoch, mit ASM Macro-Routinen einzusetzen, die mit den vertrauten mnemotischen Kürzeln Z80-Code erzeugt.

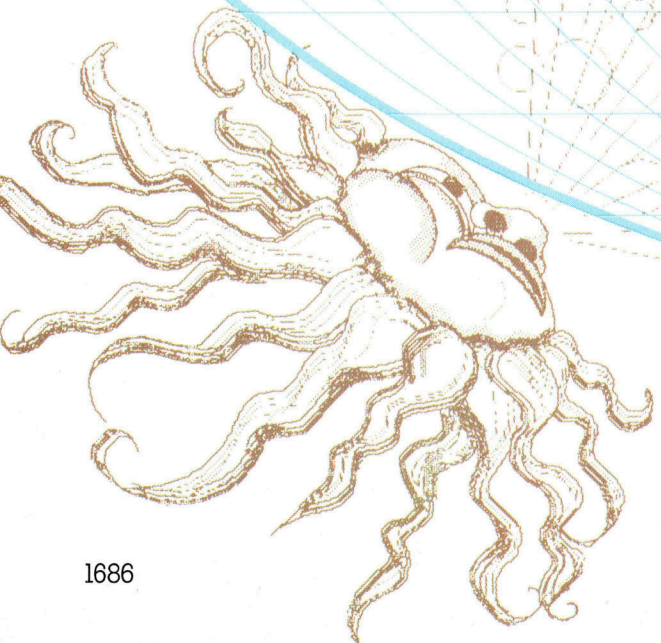
Mann über Bord

Während einer Fahrt in die Neue Welt können unvorhersehbare Ereignisse eintreten. Welche Auswirkungen diese Vorkommnisse auf die Lebensmittelvorräte und die Anzahl und Stärke der Mannschaft haben, erfahren Sie in diesem Artikel.

Um die großen Schwierigkeiten einer derart langen Seereise im 15. Jahrhundert zu simulieren, müssen wir im Programm die verschiedensten Zwischenfälle vorsehen. Einige können günstig sein, etwa ein stetiger Wind von achtern, andere dagegen die Fahrt beeinträchtigen, wie beispielsweise ein umfangreicher Verlust von Proviant.

Insgesamt gibt es 16 mögliche Ereignisse, von denen jedoch nur zwei pro Woche eintreten können. Wir schreiben eine Unterroutine, die zufällig eines der Ereignisse auswählt, sowie den Programmcode für die ersten fünf Möglichkeiten. Im letzten Artikel haben wir einen Programmteil eingebaut, der die Stärke der Mannschaft mit dem Zufallsgenerator bestimmte. Entfernen Sie die Zeilen 601 bis 604, bevor Sie Modul 6 eingeben. Dieses Modul besteht aus mehreren Teilen: einem Initialisierungsabschnitt, zwei Zeilen in der Hauptschleife und einer Unterroutine zur Auswahl eines Ereignisses sowie zur Handhabung von fünf dieser Ereignisse.

Es ist wichtig, daß jedes Ereignis nur einmal im Verlauf der Fahrt eintritt. Um das zu erreichen, wird in Zeile 42 ein Array RR() DIMensioniert, das 16 Elemente für jedes der 16 möglichen Ereignisse enthält. Jedes Element ist auf 0 gesetzt. Tritt ein Ereignis ein, wird das Element auf 1 gesetzt und somit nicht mehr aufgerufen. Es ist wichtig, die eingetretenen Ereignisse zu zählen, damit das Programm nicht endlos das Array durchsucht, ohne ein Ereignis zu finden, wenn bereits alle eingetreten sind. Diese Zahl wird in der Variablen RC verzeichnet, die bei jedem Ereignis um 1 erhöht wird. Die Anzahl der Möglichkeiten wird in RM gespeichert. Beachten Sie, daß nach Einfügen weiterer Ereignisse (im nächsten Artikel) der Wert von RM geändert werden muß.



Ereignisreiche Fahrt

In Zeile 860 der Hauptschleife wird die Unterroutine bei Zeile 5500 aufgerufen. Dadurch tritt das Ereignis nach der Angabe der geschätzten Reisedauer (die im Logbuch festgelegt ist) ein. Die Unterroutine bei 5500 wählt aus, welches Ereignis jede Woche eintreten soll. Gibt es keinen neuen Zwischenfall mehr, so wird direkt zum Hauptprogramm zurückverzweigt.

In Zeile 5510 wird eine Zufallszahl zwischen 1 und der Anzahl möglicher Ereignisse generiert (RM). Zeile 5520 überprüft das Array RR(), ob das entsprechende Element auf 1 gesetzt ist. Wenn ja, erfolgt Rücksprung zum Hauptprogramm. Wurde ein neues Ereignis ausgewählt, wird das zugehörige Flag in RR() durch Zeile 5523 auf 1 gesetzt, damit es nicht immer wieder eintreten kann.

In Zeile 5525 wird die ON X GOTO-Anweisung verwendet, um das Programm zu den zugehörigen Programmzeilen des gewählten Ereignisses zu verzweigen. Mit ON. . .GOTO und ON. . .GOSUB können einige der IF. . .THEN-Anweisungen ersetzt werden. Jedem Befehl folgt eine Liste von Zeilennummern. Ist die Va-

riable 1, so wird die erste Nummer dieser Liste verwendet. Ist der Wert =2, wird die zweite Nummer gewählt usw.

Wir befassen uns hier mit fünf Ereignissen. Der Programmcode für das erste (Mann über Bord) befindet sich bei Zeile 5540. Die letzten vier Nummern der ON. . .GOTO-Liste sind identisch, so daß das Programm bei den Zufallswerten 2, 3, 4, und 5 zum gleichen Programmteil verzweigt. Die vier Ereignisse der Elemente 2 bis 5 handhaben über Bord gegangene Vorräte und verwenden hierfür denselben Programmcode ab Zeile 5570.

Mann über Bord

Wird die erste Zeilennummer ausgewählt, erscheint eine Meldung, daß ein Mann über Bord gegangen ist. Zeile 5554 druckt die neue Gesamtanzahl der Mannschaft, indem 1 von CN (= Anzahl der Seemänner) subtrahiert wird, ohne jedoch den Wert von CN zu ändern. Das geschieht erst am Ende des Wochenberichtes durch Überprüfung, ob der Stärkewert eines Seemannes auf −999 gesetzt ist.

Natürlich muß der Rechner auch einen Seemann aussuchen, den das Schicksal trifft. Hier kommt nun der Ausdruck „Seemanns Los" so richtig zum Tragen. Was wäre ein Seefahrtsspiel ohne solche Ereignisse! Diese Auswahl erfolgt in der Schleife in Zeile 5558, die das Stärke/Typ-Array durchsucht, dabei auf 0 oder −999 gesetzte Elemente ignoriert und das erste lebende Mitglied als Opfer wählt. Zeile 5560 „wirft" den Mann ins Meer, sein Stärkewert wird auf −999 gesetzt. Der Wert von T wird dann auf 16 gesetzt, um die Schleife zu verlassen.

Die restlichen Ereignisse, mit denen wir uns in diesem Programm-Modul befassen, beziehen sich auf die vier Vorratsarten, die über Bord gehen können. Um das Spiel interessanter zu machen, wird eine zufällige Menge jedes Vorrats gewählt. Die Menge an Obst, Gemüse, Fleisch und Wasser wird durch PA(1), PA(2), PA(3) und PA(4) repräsentiert.

Inventur der Vorräte

In Zeile 5572 wird der Wert 1 von der durch Zeile 5500 generierten Zufallszahl X subtrahiert, da ein Ereignis bereits eingetreten ist. Die Zufallszahl wird jetzt zwischen 1 und 4 liegen, entsprechend einem der vier Vorräte. Zeile 5574 überprüft, ob der Vorrat ausgegangen ist. Wenn dies zutrifft, wird der Spieler informiert, daß ein Teil des Vorrats über Bord gegangen ist.

Die Menge an verlorener Nahrung oder Wasser beträgt entweder 1/3, 1/4 oder 1/5 der vorhandenen Gesamtmenge. Dies wird in Zeile 5592 bestimmt, indem die Menge des Vorrates durch 3, 4 oder 5 dividiert und dieser Wert von der Gesamtmenge subtrahiert wird. Das Programm berechnet dann die Anzahl an Wochenrationen für die übrige Vorratsmenge.

Modul sechs: Zufällige Ereignisse

Initialisiere Variablen

```
42 DIMRR(16)
43 REM INDICATORS TO SHOW IF RANDOM EVENT(N) ALREA
DY OCCURED
44 RC=0
45 REM COUNT OF RANDOM EVENTS SO FAR
46 RM=5
47 REM NO. OF RANDOM EVENTS IN PROGRAM
```

Ergänzung an Hauptprogramm

```
860 GOSUB5500
861 REM GO TO GENERATE RANDOM EVENTS
```

Zufallsereignis-Unterroutine

```
5500 REM RANDOM EVENT GENERATOR
5502 IFRC=RM THEN RETURN
5503 REM EXIT IF ALL RANDOM EVENTS DONE
5504 PRINTCHR$(147)
5505 GOSUB9200
5510 X=INT(RND(1)*RM)+1
5515 REM GENERATE RANDOM NO. BETWEEN 1 AND RM
5520 IFRR(X)=1 THEN RETURN
5522 REM RETURN IF THIS EVENT ALREADY DONE
5523 RR(X)=1:RC=RC+1
5524 REM SET IND. TO SHOW THIS EVENT DONE AND INCR
EMENT EVENT COUNT
5525 ON X GOTO 5540,5570,5570,5570,5570
5528 REM GO TO APPROPRIATE CODE FOR THIS EVENT
5530 PRINT:S$=K$:GOSUB9100
5535 GET1$:IF1$=""THEN5535
5539 RETURN: REM RETURN TO MAIN JOURNEY LOOP
5540 REM EVENT 1 - MAN OVERBOARD!
5542 PRINT
5543 S$="  DURING THE WEEK*":GOSUB9100
5545 PRINT:GOSUB9200
5546 S$=" 1 PERSON WAS LOST OVERBOARD*":GOSUB9100
5548 S$=" IN A STORM*":GOSUB9100
5550 PRINT:GOSUB9200
5552 S$="YOUR CREW IS NOW REDUCED TO*":GOSUB9100
5554 PRINTCN-1;"MEMBERS"
5558 FORT=1TO16
5559 REM SEARCH FOR CREW MEMBER TO LOSE
5560 IFTS(T,2)=0ORTS(T,2)=-999THEN5566
5562 TS(T,2)=-999:REM DEAD
5564 T=16
5566 NEXT
5568 GOTO5530
5570 REM EVENTS 2 TO 5 - PROVISIONS LOST
5572 X=X-1:REM X NOW POINTS TO PROVISION(1-4)
5574 IFPA(X)=0 OR PA(X)=-999 THEN5530
5576 REM NO ACTION IF THIS PROVISION ALREADY EXHAU
STED
5578 PRINT
5580 S$="  DURING THE WEEK*":GOSUB9100
5582 PRINT:GOSUB9200
5584 PRINT" SOME OF YOUR ";P$(X)
5586 S$=" WAS WASHED OVERBOARD*":GOSUB9100
5588 PRINT:GOSUB9200
5590 S$="YOU NOW HAVE APPROXIMATELY*":GOSUB9100
5592 PA(X)=PA(X)-INT(PA(X)/(INT(RND(1)*3)+3))
5593 REM REDUCE PROV AMOUNT BY 1/2 1/3 OR1/4
5594 PRINTINT(PA(X)/(CN*PN(X)));
5595 PRINT"WEEKS ";P$(X);" LEFT"
5599 GOTO5530
```

BASIC-Dialekte

Spectrum:
Führen Sie die folgenden Änderungen aus:

```
5504 CLS
5525 IF X=1 THEN GOTO 5540
5526 IF X>1 AND X<6 THEN GOTO 5570
5535 LET I$ = INKEY$ : IF I$ ="" THEN GOTO
     5535
```

Acorn B:
Ändern Sie das Programm wie folgt:

```
5504 CLS
5535 I$ = GET$
```

Der FX-Effekt

In dieser Folge untersuchen wir die OSBYTE-Aufrufe. OSBYTE ist eine einfache Methode, unterschiedliche Betriebssystemfunktionen des Acorn B einzusetzen.

Bei der ersten Untersuchung des Acorn B-Betriebssystems hatten wir kurz die OS-Aufrufe (OSBYTE) gestreift, die das Verhalten von OS-Funktionen verändern können. Mit *FX4,1 läßt sich beispielsweise die Reaktion des Acorn B auf die Betätigung der Cursortasten neu definieren.

In der 1.2-Version des OS gibt es über 100 dieser Aufrufe, die entsprechend viele OS-Funktionen des Acorn B steuern. Wir haben bereits festgestellt, daß durch den indirekten Aufruf von OS-Routinen die Programmfunktionen auch bei Änderungen der Hard- und Softwarekonfiguration nicht beeinflußt werden. OSBYTE ist eine praktische Methode, indirekt auf die OS-Routinen zuzugreifen.

Wenn Ihre Maschine mit der alten OS-Version 0.1 arbeitet, funktionieren einige Aufrufe von OSBYTE nicht, die im Anwenderhandbuch aufgeführt sind und die auch dieser Kurs behandelt. Wenn Sie *HELP <RETURN> eingeben, wird die Versionsnummer des Betriebssystems dargestellt.

Sehen wir uns zunächst an, wie OSBYTE von BASIC und vom Maschinencode aus eingesetzt wird. Wie die meisten OS-Aufrufe des Acorn B arbeitet auch OSBYTE mit Vektoren. Der OSBYTE-Vektor liegt bei den Adressen &20A und &20B. Die Tabelle zeigt, wie der OSBYTE-Aufruf *FX4,1 ausgegeben wird:

Der Einsatz von *FX von BASIC aus	Der Einsatz von USR von BASIC aus	Maschinencode
*FX4,1	A%=4:X%=1:D%=USR(&FFF4)	LDA #4 LDX #1 JSR &FFF4

Die Beispiele zeigen, daß alle OSBYTE-Routinen über die Adresse &FFF4 aufgerufen werden. Parameter lassen sich mit den Registern A, X und Y des Prozessors 6502 an OSBYTE übergeben. Wir können von BASIC aus A%, X% und Y% mit Werten belegen und dann über USR oder CALL eine Maschinencoderoutine aufrufen, die diese Werte in die Prozessorregister A, X und Y lädt. Mit USR (zweites Beispiel) läßt sich das Ergebnis eines Aufrufs auch wieder in eine BASIC-Variable übertragen. Dies ist besonders interessant, wenn die Daten in BASIC noch gebraucht werden. Wir kommen später noch darauf zurück.

Das dritte Beispiel ruft OSBYTE über ein kurzes Maschinencodeprogramm auf. Dabei werden die entsprechenden Register einfach mit Werten geladen und OSBYTE über &FFF4 angesprochen. Die Beispiele zeigen ebenfalls, wie sich die Parameter (4 und 1) auf drei verschiedene Weisen einfach an das OS übergeben lassen.

Unabhängig von der Art des OSBYTE-Aufrufs gibt das Register A immer an, welche der vielen OSBYTE-Routinen derzeit eingesetzt wird. X und Y übergeben die Parameter. Einige OSBYTE-Routinen brauchen keine Parameter, andere lediglich nur einen (Register X) und einige zwei (X und Y).

Hier ein paar Beispiele der drei Formate:

Keine Parameter	Ein Parameter	Zwei Parameter
*FX0	*FX4,1	*FX151,96,200

Beim Aufruf von OSBYTE-Routinen per *FX sind einige Punkte zu beachten. Wie alle *-Befehle kennt OSBYTE keine BASIC-Variablen. So löst die Ausführung des folgenden Codes die Fehlermeldung „Bad Command" aus:

```
a=4:b=1
*FX a,b
```

Das Problem läßt sich jedoch umgehen, wenn wir den FX-Befehl per OSCLI an das OS übergeben. So sieht es aus:

```
10 DIM C 100
20 a = 4
30 b = 1
40 $C = "*FX" + STR$a + "," +STR$b
50 X% = C MOD 256
60 Y% = C DIV 256
70 CALL &FFF7
```

Der FX-Befehl darf innerhalb einer Programmzeile nur als jeweils letzte Anweisung eingegeben werden:

```
*FX 4,1:PRINT "Hallo!"
```

erzeugt die Fehlermeldung „Bad Command", da das OS den Doppelpunkt und PRINT als Teil des FX-Befehls interpretiert.

Beide Probleme werden dadurch verursacht, daß *FX-Aufrufe nicht vom BASIC-Interpreter bearbeitet werden, sondern vom Befehlszeileninterpreter (CLI), der BASIC-Zeilen mit mehreren Befehlen nicht verarbeiten kann.

Ebenso wie sich mit X und Y Parameter an

OSBYTE übergeben lassen, so können auch die vom OS eingesetzten Variablen zurückgeben werden. Dieser Vorgang wird ebenfalls über USR oder eine Maschinencoderoutine gesteuert. Die Maschinencoderoutinen sind leichter zu bedienen, da die per USR gelieferten Werte erst decodiert (in Bits zerlegt) werden müssen. Die Register X und Y enthalten die rückgemeldeten Werte, wobei das Übertragsflag bei einigen Modulen gegebenenfalls auch vorhandene Fehler anzeigt.

CALLs von OSBYTE

Die zurückgegebenen Ergebnisse hängen natürlich von der eingesetzten Routine ab – das heißt von dem Wert, der sich bei Aufruf im A-Register befand. Nicht alle OSBYTE-Routinen geben Ergebnisse zurück, doch liefern viele Module wichtige Daten über das OS.

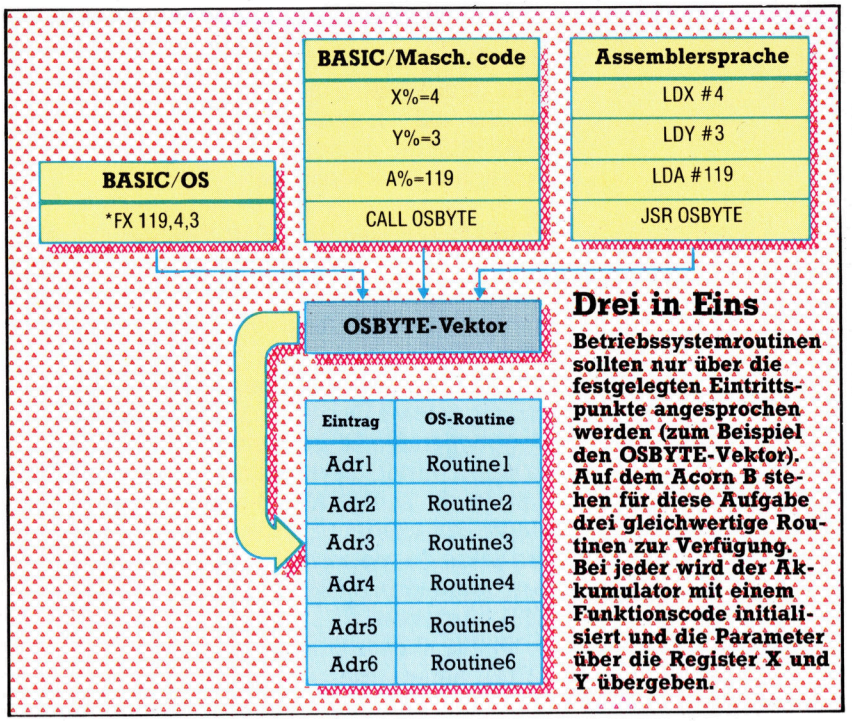
Zwei Arten von Informationen werden von OSBYTE zurückgemeldet: zunächst die Daten, die von einem Teil des Systems kommen – einer Schnittstelle oder einer Systemvariable. CALLs von OSBYTE, die diese Art von Information liefern, werden „read only"-CALLs genannt. Ein typisches Beispiel ist der OSBYTE-Aufruf mit A=129, den BASIC für die Funktion INKEY() einsetzt.

Mit den Registern X und Y läßt sich auch die vom OS verlangte Zeitverzögerung einstellen. Das X-Register speichert dabei das niederwertige Byte der Verzögerung (in Hundertstelsekunden) und Y das höherwertige. Für den Aufruf dieser Funktion können Sie die unten abgedruckte Maschinencodeversion einsetzen. Sie veranlaßt das System, bis zu einer Sekunde auf das Drücken einer Taste zu warten. Die Register X und Y geben die Werte zurück. Wenn das Übertragsflag auf 0 steht und das Y-Register ebenfalls 0 enthält, wurde der CALL mit einem Tastendruck beendet. Der ASCII-Wert der Taste ist dann in X gespeichert. Wenn Y den Wert 255 enthält und C auf 1 steht, wurde während der Verzögerungszeit keine Taste betätigt. Steht C auf 1 und Y auf 27, wurde die Escape-Taste gedrückt.

Das folgende Maschinencodebeispiel zeigt diesen OSBYTE-Aufruf. Wenn bei der Rückkehr das Übertragsflag gesetzt ist, verzweigt sie in eine weitere Subroutine, die dann über den Wert des Y-Registers prüfen könnte, ob eine alphanumerische Taste oder Escape gedrückt wurde.

```
1000 LDA    #129   /OSBYTE setzen
1010 LDX    #100   /Parameter
1020 LDY    #0
1030 JSR    &FFF4  /OSBYTE-Aufruf
                   ausführen
1040 BCS    Fehler
1050 RTS
1060 .Fehler       /Fehlerbehandlung
```

Es ist wichtig, sich diesen Aufbau einzuprägen, da er in ähnlicher Form auch für andere OS-BYTE-Aufrufe Verwendung finden kann. Besonders die Prüfroutinen sichern ein zuverlässiges Arbeiten und sollten stets zur Absicherung Verwendung finden.



Andere CALLs von OSBYTE, speziell die Aufrufe mit einem Wert zwischen 166 und 255, können sowohl Variablen aus dem OS lesen als auch dorthin schreiben. Wie kann nun OSBYTE überhaupt wissen, ob ein Lese- oder Schreibvorgang ausgelöst wurde?

Die Antwort ist einfach: Wenn an OSBYTE ein Parameter übergeben werden soll, muß das X-Register diesen Wert enthalten und Y auf 0 stehen. Für das Lesen einer der Systemvariablen wird X auf 0 und Y auf 255 gesetzt und dann der Aufruf durchgeführt. Die Ergebniswerte befinden sich danach in den Registern X und Y.

Wichtige Funktionen

OSBYTE-Aufrufe erledigen die „Verwaltung" des Betriebssystems und werden für viele Aufgaben eingesetzt – Diskettensystem, Tastatursteuerung, Econet, die Tasten Escape und Break etc. Es ist unmöglich, hier alle OSBYTE-Aufrufe aufzuführen. Wir werden uns daher auf einige wichtige Funktionen beschränken, die im Anwenderhandbuch des Acorn B nicht ausreichend erklärt sind.

Funktionstasten: *FX18 hat keine Parameter. Der Befehl löscht im Speicher die aktuellen Definitionen der Funktionstasten und macht so eine Neubelegung möglich.

*FX225 bis *FX228: Wenn die Funktionstasten nicht mit Strings programmiert sind, bewirkt dieser Aufruf, daß die roten Funktionstasten ASCII-Werte ausgeben. Zum Beispiel veranlaßt *FX225,n, daß die Funktionstaste f0 den ASCII-

Code für n liefert, f1 den ASCII-Code für (n+1) etc. *FS226 führt die gleiche Aufgabe für die mit Shift gedrückten Funktionstasten aus, *FX227 für die mit Control gedrückten Tasten und *FX228 für die gemeinsam mit Shift und Control betätigten Funktionstasten.

Video-Funktionen und VDU-Treiber: Der Bildschirm des Acorn B wird hauptsächlich mit Werten gesteuert, die an VDU-Treiber übermittelt werden. Etliche Aufrufe von OSBYTE sind dabei sehr hilfreich.

*FX19: Der Befehl arbeitet ohne Parameter und eignet sich für die Programmierung von Bewegungsgrafik. Er bewirkt, daß der Computer so lange wartet, bis das nächste Bild vollständig dargestellt ist. Der Aufruf sollte in jeder Pause eingesetzt werden. Da die Anzeige fünfzigmal pro Sekunde wiederholt wird, lassen sich damit Verzögerungen einsetzen oder Interupts an die CPU ausgeben.

*FX218: Dieser Schreib-/Leseaufruf von OSBYTE liefert die Länge der „VDU-Warteschlange". Wir hatten bereits erwähnt, daß auf einige VDU-Codes andere Codes folgen müssen. Die Anzahl der Bytes, auf die die VDU-Treiber zu einem gegebenen Zeitpunkt warten, ist die Zahl der Bytes in der VDU-Warteschlange. Der Befehl wird hauptsächlich als Lesevorgang eingesetzt. Das Ergebnis befindet sich dann erwartungsgemäß im X-Register.

*FX20: Mit diesem Aufruf können Sie alle ASCII-Zeichen zwischen 32 und 255 umdefinieren (und nicht nur eine begrenzte Anzahl). Im Modus 0 bis 6 sind dabei auch die normalen Tastaturzeichen zugänglich. Für die Umdefinierung von Zeichen, deren Codes zwischen 32 und 128 liegen, muß im Speicher Platz reserviert werden (der BASIC-Bereich wird mit PAGE auf einen höheren Wert gesetzt). Die Umdefinierung arbeitet mit VDU23 und Blöcken von 32 Zeichen, wobei jeder Block 256 Bytes benötigt. Der einzige Parameter, der benötigt wird, wird in das X-Register übergeben. Im Handbuch ist diese Technik ausführlich beschrieben.

X-Parameter	Angesprochener Buffer
0	Tastatur
3	Drucker
4	Tonkanal 0
5	Tonkanal 1
6	Tonkanal 2
7	Tonkanal 3

Tonerzeugung: *FX210,n schaltet sämtliche Soundeffekte ab, und *FX210,0 stellt sie wieder an. Setzt man einen anderen Parameter als X ein, so wird die Klangausgabe deaktiviert.

*FX211 bis *FX214: Diese als Schreib-/Leseaufrufe ausgelegten Routinen steuern den von CTRL-G oder VDU7 erzeugten Klang. *FX211,n bestimmt den Kanal des Tonerzeugungschips, auf dem der VDU7-Klang generiert wird. *FX212,n legt die Lautstärke der fertigen Töne fest oder definiert die zur Tonerzeugung notwendige Hüllkurve. Die Codierung der Lautstärke hat das gleiche Format wie der Amplitudenparameter im BASIC-Befehl SOUND (−15 maximale und 0 minimale Lautstärke, positive Werte bezeichnen die Nummer der Hüllkurve). Der per X-Register an den FX CALL übergebene Wert wird nach der Formel (n−1)*8 berechnet, wobei n der Parameter ist. *FX213,n steuert die Höhe des von VDU7 erzeugten Tones und *FX214,n die Dauer.

Escape-Taste: Mit *FX229,n läßt sich die Escape-Taste „abschalten". *FX229,1 schaltet sie aus, während *FX229,1 ihre normale Funktion wiederherstellt.

Mit *FX220,n kann der Anwender eine Taste definieren, die den Escape-Vorgang auslöst; die normale Escape-Taste wird dabei abgeschaltet. n bestimmt den ASCII-Wert der Taste. So veranlaßt *FX220,65, daß die Taste A eine Escape-Funktion auslöst.

Buffer: *FX21,n löscht die im Buffer vorhandenen Bytes. So beseitigt das Löschen des Tastaturbuffers alle Zeichen, die sich dort während eines Programmablaufs angesammelt haben. Wenn sich bei Ausführung eines GET-Befehls noch Zeichen im Tastaturbuffer befinden, holt GET sich ein Zeichen aus dem Buffer und nicht von der Tastatur. Das Löschen des Klangbuffers beendet die Tonerzeugung dieses Kanals, selbst wenn sich darin noch Befehle befinden, auf die der Klangchip wartet. X definiert den Buffer, der gelöscht werden soll (siehe Tabelle).

*FX138,n,m setzt den Wert m in den Buffer n ein. Die Buffernummern sind die gleichen wie bei *FX21. So legt *FX138,0,65 das Zeichen A im Tastaturbuffer ab.

Bei all diesen Aufrufen von OSBYTE befinden sich die entsprechenden Parameter in den Registern A, X und Y.

Diese Aufstellung erhebt keinen Anspruch auf Vollständigkeit.

Wir haben nur einige wesentliche Funktionen von OSBYTE behandelt. Es gibt zwar noch weitaus mehr, doch verwenden alle die Techniken, die in diesem Artikel beschrieben sind.

Korrektur
Im Programm (Seite 1634) für anwenderdefinierte Funktionen (USERV) fehlte ein Teil. Hier das vollständige Listing.

```
5  DIM C 100
10 USERV=&200
20 ?USERV=&00
25 ?(USERV+1)=&0C
30 FOR I%=0 TO 2
   STEP 2
40 P%=&C00
50 [ OPT I%
60     CMP #0
70     BNE notcode
80     TXA
90    .loop
95     JSR &FFE3
100    DEY
110    CPY #0
120    BNE loop
130    RTS
140   .notcode
145    RTS
150 ]: NEXT I%
160 .
200 FOR rep=1 to 10
210 FOR asc=33 TO 48
220 asc$=STR$(asc)
230 rep$=STR$(rep)
240 code$="*CODE
      "+asc$+","+rep$
250 $C=code$
260 X%=C MOD 256
270 Y%=C DIU 256
280 CALL &FFF7
290 NEXT: NEXT
```

Die gedruckte Seite

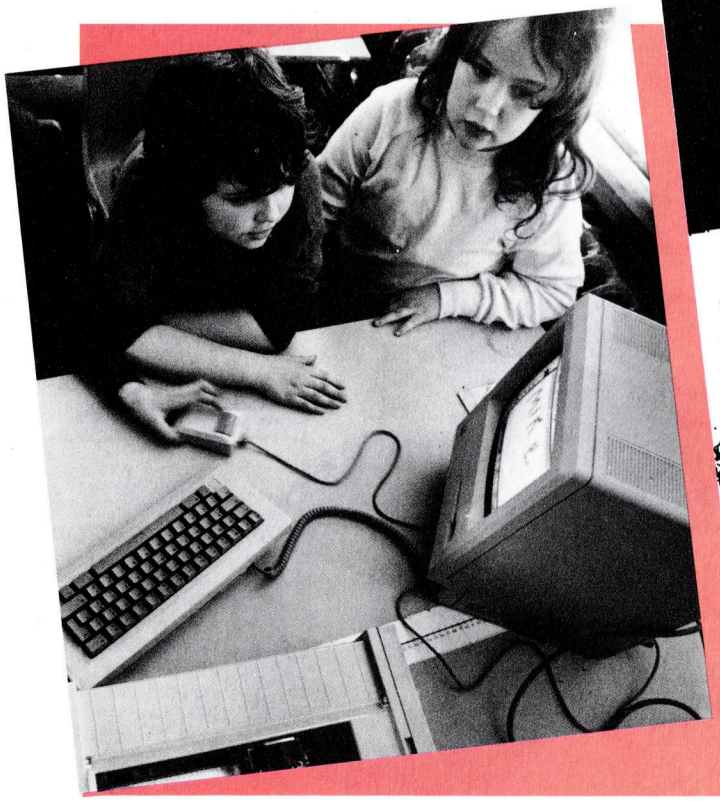
Wir beenden unsere kurze Serie über die Einführung neuer Technologien im grafischen Gewerbe und zeigen Möglichkeiten, wie sich Produktionskosten reduzieren lassen.

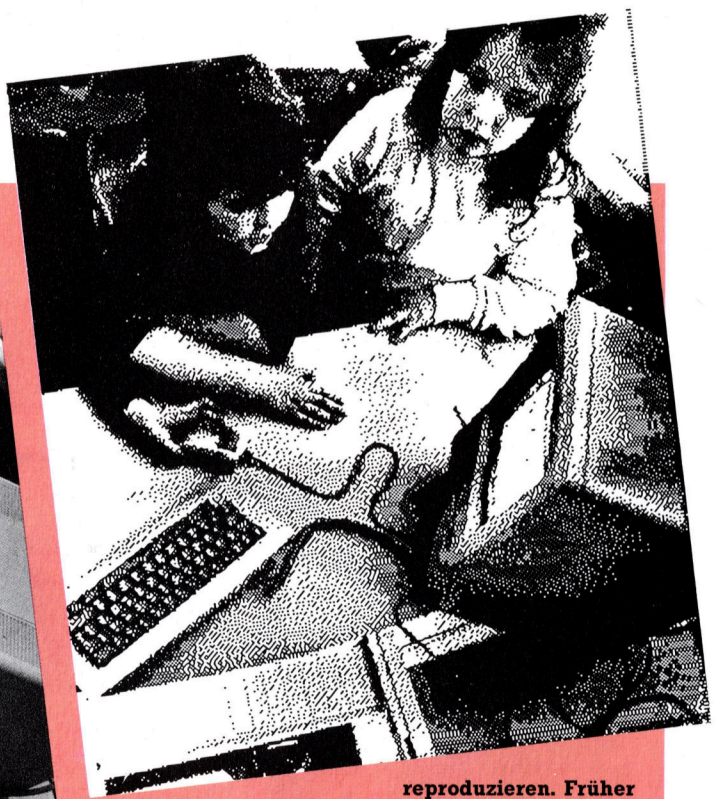
Die Einführung des Buchdrucks mit beweglichen Lettern durch Johannes Gutenberg war ein bedeutendes Ereignis, weil damit „das geschriebene Wort" mehr Menschen zugänglich wurde. Vor dieser Erfindung wurden Bücher mühsam von Hand hergestellt – entweder handgeschrieben oder aber Buchstabe für Buchstabe in Holzplatten geschnitten. Da der Arbeitsaufwand und somit die Produktionskosten sehr hoch waren, gab es nur wenig Exemplare. Johannes Gutenbergs Technik der Einzelbuchstaben reduzierte die Herstellungskosten drastisch. Es wurde möglich, mehr Exemplare eines Werkes kostengünstig zu drucken.

Natürlich sind die jüngeren Fortschritte in der Druckindustrie mit der Erfindung der Druckerpresse nicht vergleichbar. Dennoch lassen sich einige Vergleiche ziehen. Mit den neuen, auf Computer-Technologie basierenden Herstellungsmethoden können Vorlagen erstellt werden, die qualitativ traditionellen Produkten entsprechen, aber zu niedrigeren Kosten herstellbar sind. Während durch Gutenbergs Arbeit Bücher einem größeren Publikum zugänglich wurden, hat der computerisierte Herstellungsprozeß eine andere, aber nicht weniger wichtige Auswirkung.

Die Verkaufspreise für Zeitschriften und Bücher sind bereits recht niedrig, und es ist kaum anzunehmen, daß niedrigere Herstellungskosten und somit ein geringerer Endverbraucherpreis den Verkauf entscheidend fördern. Für Verlagshäuser aber bedeutet die Produktionskostenreduzierung, daß weniger Exemplare verkauft werden müssen, um kostendeckend arbeiten zu können. Dadurch wiederum können Spezialpublikationen, die zuvor aus ökonomischen Gründen nicht realisierbar waren, auch endlich für kleinere Zielgruppen veröffentlicht werden.

In der letzten Ausgabe zeigten wir den Herstellungsprozeß bei der Zeitschriftenproduktion auf und führten derzeit stattfindende Veränderungen an. Verfolgen wir den Gedanken einer optimierten Produktion weiter, so kom-





Mit dem Digitiser kann der Anwender Bilder speichern und dann im Seitenlayout reproduzieren. Früher waren diese Geräte teuer. Das gezeigte, digitalisierte Bild wurde mit dem „Thundercan"-System hergestellt.

men wir zum „Stand-Alone"-Microcomputer, der sowohl zur Textverarbeitung wie zur Layout-Gestaltung als auch als Steuereinheit für einen Hochleistungsdrucker ohne weiteres verwendet werden kann.

Der mit 512 KByte ausgerüstete Apple Macintosh (bekannt als „Fat Mac") verfügt über hochentwickelte Software, die speziell für diesen Zweck geeignet ist. Das Programmpaket enthält den „PageMaker". Benutzt man einen Hochleistungsdrucker in Verbindung mit diesem Programm, so kann man Publikationen produzieren, die fast Satzqualität haben.

Der Macintosh mit seiner Fenstertechnik, Piktogrammen und der Maus-Steuerung ist ideal für eine Anwendung wie den PageMaker. Die vertrauten „Pulldown"-Menüs und Bildschirmfenster werden sofort nach Laden des PageMaker angezeigt. Der Bildschirm des Mac wird dann zu einem Layouttisch, auf dem sämtliche Werkzeuge für die Seitengestaltung zur Verfügung stehen.

Layout am Schirm

Normalerweise beginnt der Layouter mit einem Spaltenbogen, der Standardseite jeder Publikation, und zieht Linien, um Textläufe, Überschriften, Paginierung und weiteres unterzubringen. Kopien dieses Rasters werden dann benutzt, um der Publikation ein einheitliches Gesicht zu geben. Traditionell werden Layoutbögen für die Nullnummer mit der Hand gezeichnet und bleiben dann solange Standard, bis eine Layoutänderung erfolgt.

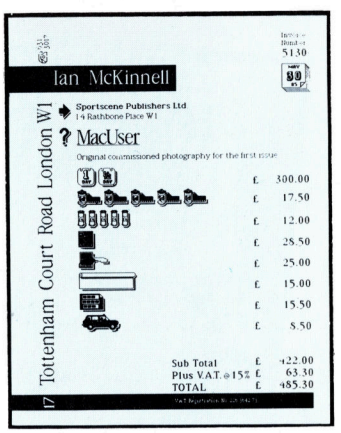
Diese Methode wird durch den PageMaker ersetzt, mit dem die Erstellung von „Master"-Seiten möglich ist. Damit wird zunächst das Seitenformat (z. B. DIN A4) festgelegt, die Kolumnen, Schnittkanten und Positionen der Paginierung, der Logos und Kolumnentitel bestimmt. Ist eine „Master"-Seite für ein bestimmtes Objekt erstellt, kann sie jederzeit wieder abgerufen werden und als Layoutbogen benutzt werden.

Jetzt lassen sich die individuellen Seiten gestalten. Normalerweise benutzt der Layouter ein Federmesser, um Texte und Bilder auszuschneiden und sie auf dem Raster zu positionieren. Beim PageMaker können Text und Fotos von Diskette geladen werden. Da diese Programme mit anderen Macintosh-Programmen kompatibel sind, läßt sich ein mit MacWrite geschriebener Artikel direkt in den PageMaker laden und weiterverarbeiten. Die Gestaltung erfolgt mit PacPaint und MacDraw.

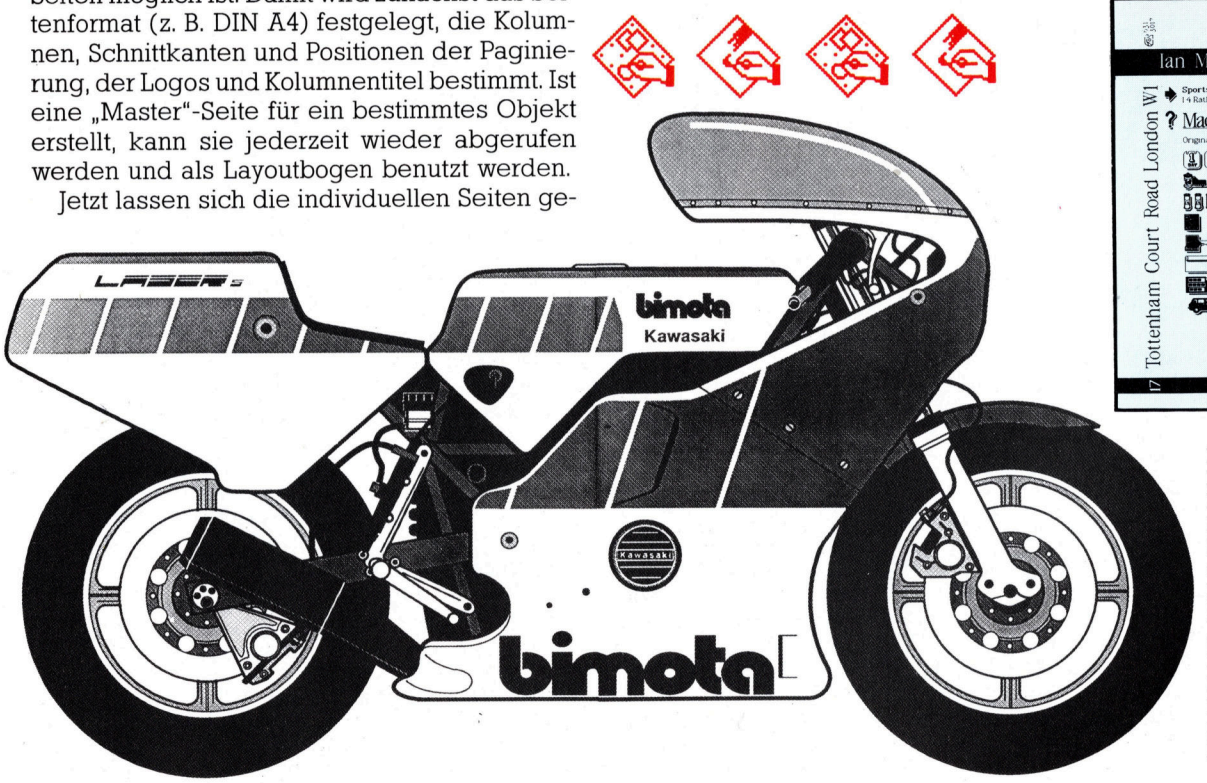
Grafiken verkleinern

Nachdem die verschiedenen Komponenten geladen worden sind, bedient man sich der „Toolbox" auf dem Bildschirm. Ihre „Werkzeuge" erlauben einem zum Beispiel, Text auf dem Bildschirm hinzuzufügen oder zu editieren – indem man das Piktogramm A wählt oder mit der „Repro" Grafiken verkleinert, vergrößert oder beschneidet, um sie dem vorhandenen Raum anzupassen. Mittels der Maus können Text und Grafik fixiert und auf dem Layoutbogen bewegt werden.

PageMaker bietet aber auch Möglichkeiten, die Seite mit speziellen Elementen auszustatten, etwa Kästen um bestimmte Textbereiche. Es enthält Piktogramme, mit denen man Linien, Kreise, Ellipsen und Rechtecke mit oder ohne „runde Ecken" ziehen kann. Selbst die Stärke der Linien läßt sich bestimmen, indem man das entsprechende Menü anwählt. Sind die Konturen einmal gezogen, können sie schwarz oder weiß, mit unterschiedlichen Grauwerten oder Rastern gefüllt werden. PageMaker bietet dem Layouter unschätzbare Vorteile, ist aber so einfach in der Handhabung, daß er auch von uner-



Diese Rechnung wurde direkt mit einem Macintosh entworfen und gedruckt. Die Markenzeichen von Unternehmen und Warenzeichen werden ebenfalls auf Rechnungen und Briefbögen gedruckt.



Mit Programmen wie MacDraw und MacPaint geschaffene Files können auf Diskette gespeichert und dann in den PageMaker geladen werden. Anschließend lassen sie sich für den Druck bearbeiten. Sowohl MacDraw als auch MacPaint bieten dem Layouter optimale Möglichkeiten.

fahrenen Anwendern erfolgreich und ohne Schwierigkeiten benutzt werden kann.

Der „Fat Mac", ein Laser Writer und die erforderliche Software kosten eine schöne Stange Geld, doch die Einführung der Rechner auf WIMP-Basis wie etwa des Atari 520ST mit dem GEM-Betriebssystem könnten zu drastischen Kostenreduzierungen führen.

Kostenreduzierung

Wieviel Zeit und Geld kann man mit einem elektronischen Seitengestaltungs-System wie dem Macintosh, Laser Drucker und PageMaker tatsächlich sparen? Nehmen wir an, der Macintosh würde für die Herstellung einer 16seitigen Zeitung in Profiqualität benutzt werden. Vorausgesetzt, diese Publikation würde von einem kleinen Redaktions- und Layout-Team erstellt, und sämtliche Arbeiten bis auf den Satz würden hausintern durchgeführt, ergäbe sich folgende Kostenrechnung:

Mit dem PageMaker könnte eine vergleichbare Publikation von einer Person in etwa fünf Stunden zu einem Stundenpreis von etwa 40 Mark hergestellt werden. Die 16 Seiten ließen sich also mit einem Produktionskostenaufwand von 200 Mark erstellen. Doch es gibt noch andere Vorteile. Bei der herkömmlichen Methode ist ein sorgfältig geplanter Arbeitsablauf Voraussetzung, der in unserem Beispiel rund zwei Wochen beanspruchte. Mit dem PageMaker entfallen viele organisatorische Probleme, da Textkorrekturen, Kürzungen und Ergänzungen.

Arbeit	Aufwand	Kosten DM
Manuskriptauszeichnung	5 Stunden à DM 20,–	100,–
Satz und Korrekturen	pauschal	1000,–
Herstellung	5 Stunden à DM 20,–	100,–
Korrekturlesen	8 Stunden à DM 20,–	160,–
Layout und Reinzeichnung	6 Stunden à DM 20,–	120,–
Reinzeichnung/Endmontage	16 Stunden à DM 20,–	320,–
	Gesamt:	1800,–

Das Foto zeigt ein modernes Fotosatzsystem auf Laserbasis. Es ist ein typisches computergesteuertes System und kann bis zu 368 000 Zeichen pro Stunde erzeugen.

Das dargestellte System besteht aus drei Hauptkomponenten. Links steht eine Linotronic 300 – ein hochwertiges Lasersatzsystem mit einer Auflösung von bis zu einer Million Punkte pro Quadratzentimeter. Die Schrift kann erweitert oder komprimiert, kursiv gestellt und negativ gesetzt werden. Bis zu 32 Schrifttypen pro Zeile sind darstellbar.

Das Terminal in der Mitte erlaubt die manuelle Texteingabe von Diskette oder über ein angeschlossenes Modem.

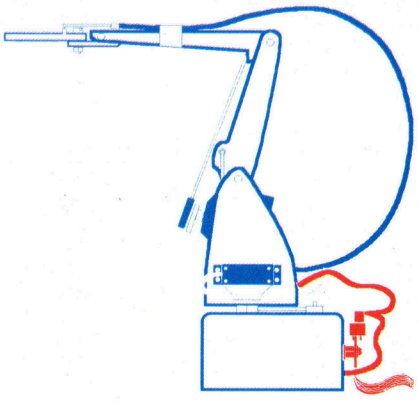
Die Darstellungseinheit rechts, eine Linotype-Paul Typeview 300, stellt den Text in Schriftart, Schriftgrad und Position so dar, wie er gesetzt aus dem Gerät kommt. Mit derartigen Systemen lassen sich nicht nur fertige Aufsichtsvorlagen erstellen, sie können auch an größere Netzwerke angeschlossen werden, zu denen verschiedene Verlage Zugriff haben. Linotype zum Beispiel bietet ein System, das bis zu 40 interaktive Terminals gleichzeitig steuern kann und dabei On-Line speichert und gleichzeitig Zugriff zu Peripheriegeräten erlaubt. Ein einziges von einer Zeitung betriebenes System wäre so in der Lage, Eingaben von Redakteuren im Haus und von Außenredaktionen zu verarbeiten, Archivmaterial aufzunehmen, es an die Redaktionen weiterzuleiten, es mit Illustrationen zu kombinieren und das Gesamtmaterial an das Layout zu geben, bevor die Ausgabe des Endprodukts erfolgt.

Schrittmacher

Bei modernen Litho-Techniken bedient man sich der Fotografie, um Bilder auf Papier zu übertragen. Druckfertige Seiten müssen deshalb als „Aufsichtsvorlagen" geliefert werden. Der Text muß sauber gesetzt vorliegen. Das ist anders als früher, als noch Lettern in Metall hergestellt wurden.

Fotosatzmaschinen liefern die benötigten Vorlagen. Der Text wird über Tastatur oder von Diskette eingegeben und auf langen Fahnen – auf Fotopapier – ausgegeben, das den sauber gesetzten Text enthält und in die Reinzeichnung geklebt wird.

Ältere Fotosatzgeräte arbeiteten mit einer runden Scheibe, die alle Satzzeichen in einem bestimmten Schnitt trug. Lichtempfindliches Papier wurde dann an einer Seite der Scheibe vorbeigeführt. War der gewünschte Buchstabe in der richtigen Position, erfolgte die Belichtung, und damit wurde der Buchstabe aufs Papier übertragen.



Es ist soweit!

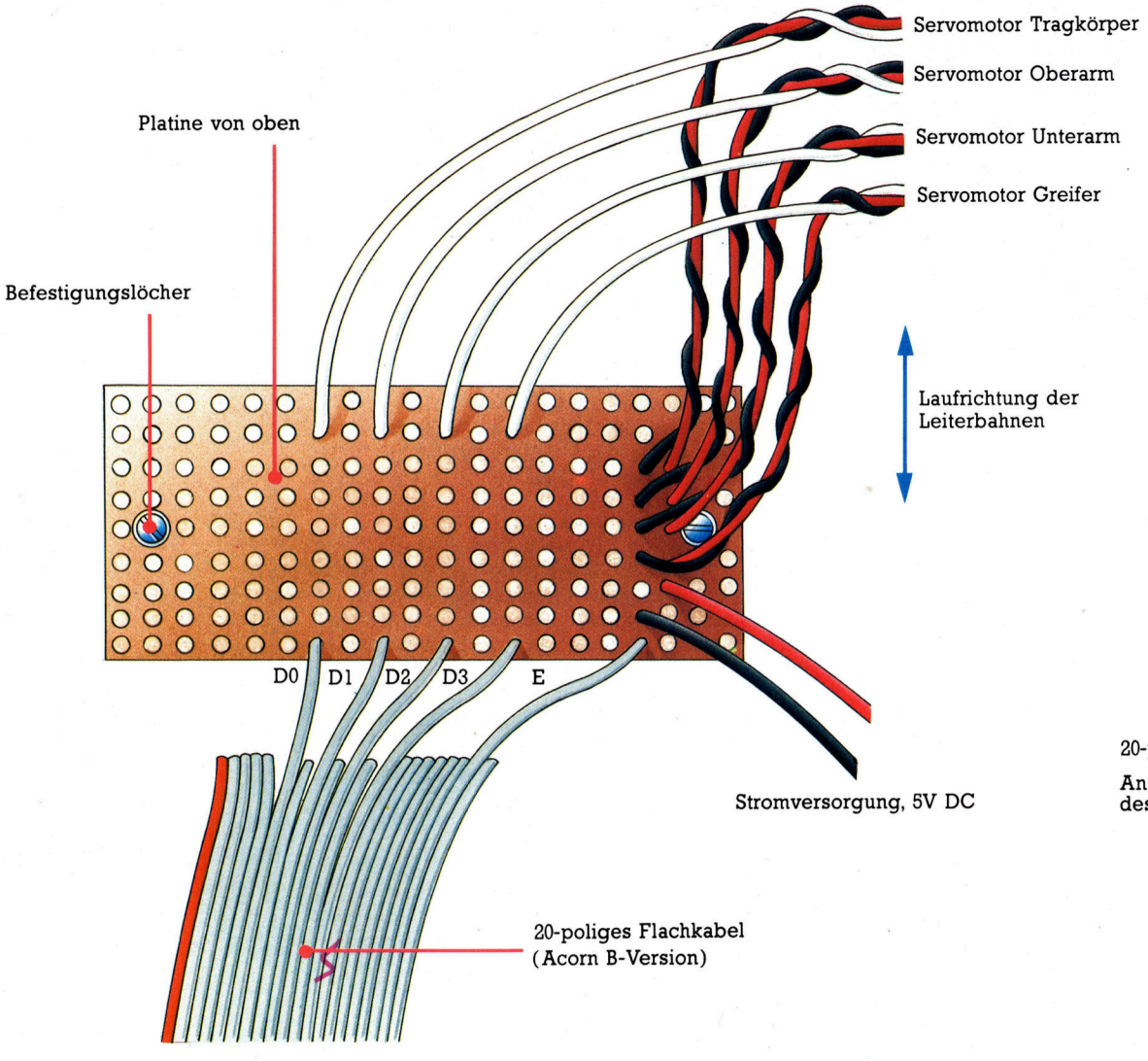
Der mechanische Aufbau unseres Robot-Armes ist abgeschlossen; was fehlt, sind nur noch die elektrischen Verbindungen. Zu jedem der vier Servomotoren müssen Stromversorgungs- und Steuerleitungen geführt werden.

1. Platine vorbereiten

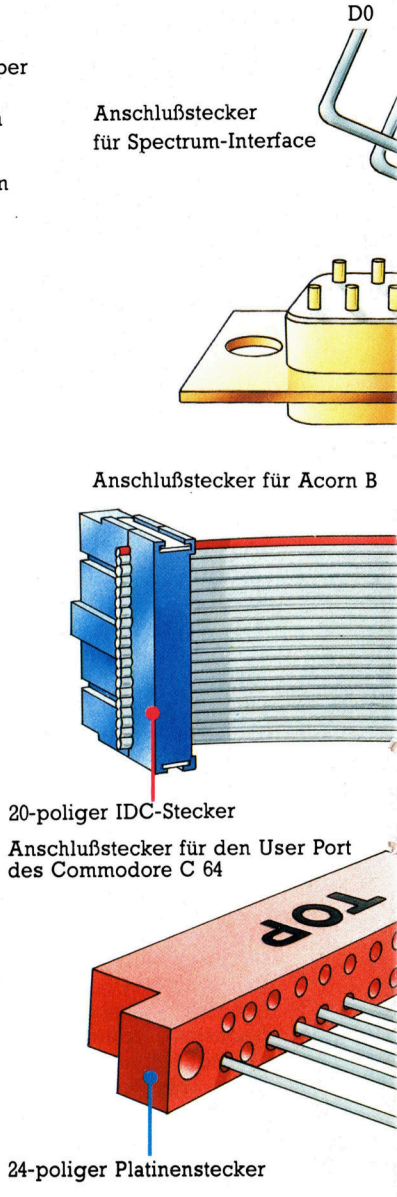
Jeder Servomotor hat drei Anschlußleitungen. Die bei unserem Projekt verwendeten Futuba-Motoren haben zur sicheren Unterscheidung verschiedenfarbige Kabel: Die Steuerleitungen sind weiß, die Plus-Leitungen der Stromversorgung rot und die Masseleitungen schwarz. Zur Verbindung der Motoranschlüsse mit dem Computer verwenden wir ein 20-poliges Flachkabel und eine Leiterplatte mit 20 Bahnen à neun Löcher, die hinten auf der Basis des Robot-Armes montiert werden soll. Zuerst die Platine (Veroboard) zuschneiden und die beiden seitlichen Befestigungslöcher bohren. Mit der Platine als Schablone werden dann in die Rückwand der Basis zwei weitere Löcher gebohrt, so daß die Platine sich dort mit Maschinenschrauben befestigen läßt. Ein weiteres Loch dient zum Durchführen der diversen Anschlußleitungen des Servomotors im Basis-Gehäuse.

Vor dem Festschrauben der Platine werden die von den vier Motoren herangeführten Leitungen nach dem Vorbild der Zeichnung mit der Platine verlötet. Die vier Steuerleitun-

1. Platine vorbereiten



2. Anschluß am C



gen und die Computer-Masseleitung liegen an der Platinen-Unterkante. Das Bild zeigt das 20-polige Kabel für den Acorn-B, für die Spectrum- und Commodore-64-Version können Sie entweder fünf Einzel-Litzen oder ein 5-poliges Flachkabel verwenden. Die weißen Steuerleitungen der Motoren sitzen jeweils direkt über den entsprechenden Datenleitungen vom Computer. Die schwarzen Kabel liegen alle auf der gleichen Leiterbahn wie der Masseanschluß des Computers und der Minuspol der Stromversorgung, der positive Anschluß wird mit roten Kabeln verbunden.

Zuletzt wird die 5-Volt-Stromversorgung mit der Platine verbunden. Die interne 5-Volt-Stromversorgung des Acorn B, Spectrum bzw. Commodore 64 reicht zum Antrieb der vier Servomotoren nicht aus. Als Alternative dient entweder eine 4,5-Volt-Flachbatterie oder ein Batteriehalter mit 3 1,5-Volt-Monozellen. Natürlich geht es auch mit einem 5-Volt-Gleichstromnetzteil entsprechender Leistung, dann muß aber anstelle der Stromversorgungs-Leitungen ein kleiner Netzteil-Stecker auf der Verbindungsplatine montiert werden. Der negative Anschluß des Steckers wird dann mit der gleichen Leiterbahn verbunden, an der die Masseleitung des Computers und die schwarzen Leitungen der Servomotoren angelötet sind. Entsprechend wird der positive Anschluß an der Leiterbahn der roten Servomotor-Kabel angeschlossen.

2. Anschluß am Computer

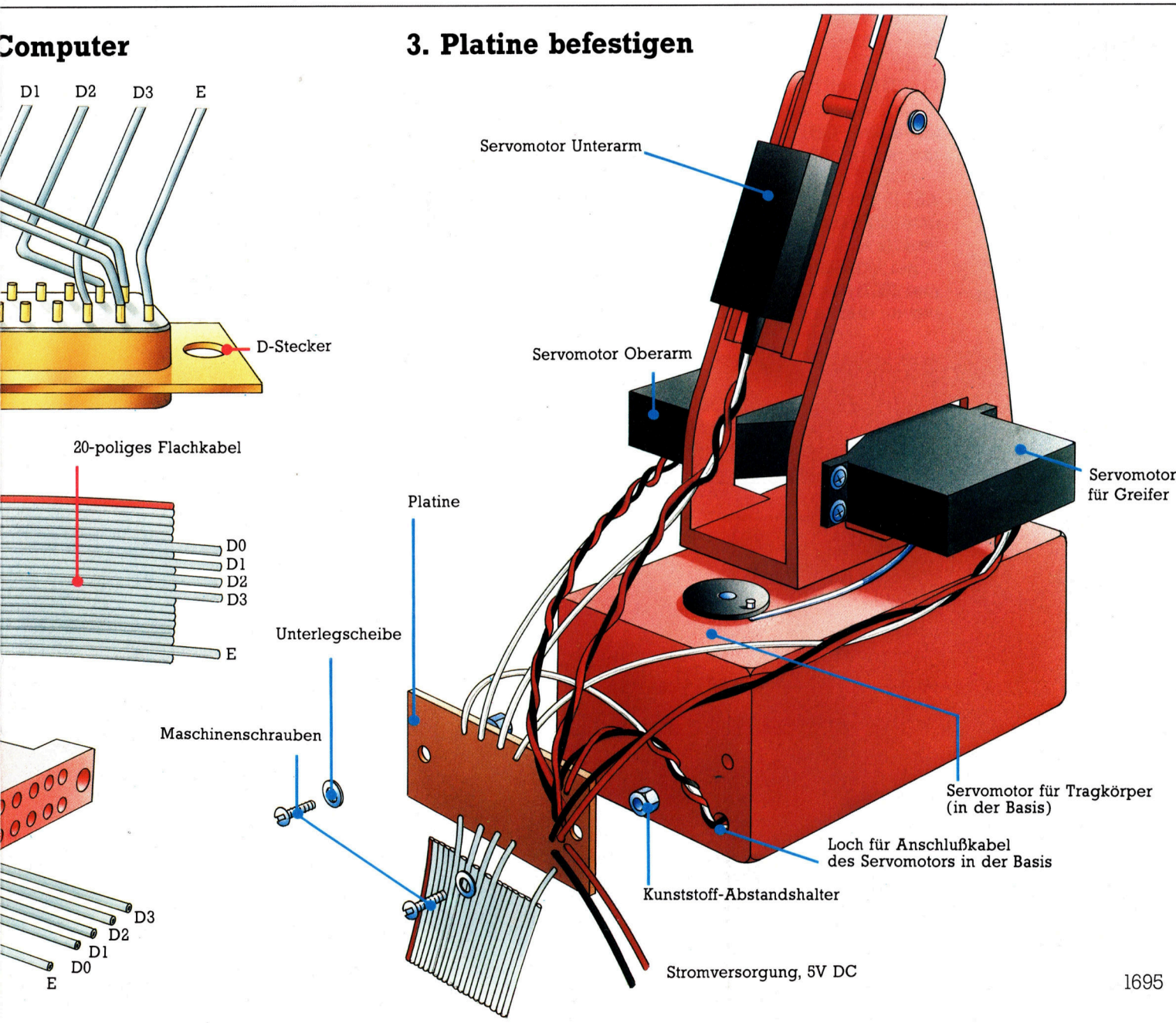
Was jetzt noch fehlt, ist die Verbindung der Platine mit den Datenleitungen D0 bis D3 und der Masse des Computers. Beim Acorn B verwenden Sie dazu ein normales 20-poliges Flachkabel mit IDC-Stecker, für den Commodore C 64 einen 24-poligen Platinenstecker (0,15 Zoll). Leitungen wie im Bild unten an der Platine festlöten.

Sowohl beim Acorn B als auch beim Commodore ist die Schaltung für Ein- und Ausgabe über den User Port bereits in die Hauptplatine integriert. Der Spectrum verfügt in der Grundausstattung nicht über die entsprechende Schaltung – er muß erst mit einem speziellen Interface ausgerüstet werden, das sich am Erweiterungsanschluß aufstecken läßt. Den Aufbau dieses Interface haben wir schon beschrieben (siehe S. 1464, 1490, 1518).

Die Daten- und Stromversorgungsanschlüsse des Spectrum-Interface sind in einem 12-poligen D-Stecker zusammengefaßt, der sich für einen direkten Anschluß am Robot-Arm nutzen läßt. Dazu brauchen Sie einen 15-poligen D-Anschluß, wie er auch am Gehäuse unseres Robot-Autos eingebaut ist. Er wird gemäß dem Bild unten mit dem freien Ende des Flachkabels von der Platine des Robot-Arms verbunden und mit einem Stekkergehäuse versehen.

3. Platine befestigen

Nach der gründlichen Prüfung aller Verbindungsleitungen und Anschlüsse können Sie die Platine mit zwei Maschinenschrauben auf der Rückseite der Basis befestigen. Datenleitung D0 (ganz links) ist mit der weißen Leitung des Servomotors in der Basis verbunden; D1 führt zum Servomotor für den „Oberarm", D2 zum Motor des „Unterarms", der neben dem Schultergelenk sitzt. Zur Steuerung des Greifermotors (rechts am Tragkörper) dient Datenleitung D3.



Wortspiele

Mit der Verarbeitung von Strings beschließen wir unsere FORTH-Serie. Dabei simulieren wir verschiedene Möglichkeiten der BASIC-Stringverarbeitung in FORTH.

FORTH kann Strings auf drei verschiedene Weisen im Speicher unterbringen. Jede Methode arbeitet mit einer Adresse, die sich leicht einsetzen läßt und auf den Speicherbereich zeigt, in dem der String untergebracht wurde. Alle drei Methoden verwenden jedoch unterschiedliche Techniken, um die Länge des Strings festzustellen. Beim „gezählten String" wird die Längeninformation im ersten Byte des Bereichs untergebracht, in dem auch der String gespeichert wurde. Der „Textstring" legt den Längenwert auf den Stapel, wo er verändert werden kann. Einige FORTH-Versionen unterstützen Strings, die mit NUL beendet sind. Dabei ist das Ende des Strings einfach durch ein Nullbyte markiert.

Wir hatten bereits gesehen, daß sich das Wort ." zwar leicht einsetzen läßt, für kompliziertere Stringverarbeitung aber nicht geeignet ist, da dabei zwei wesentliche Probleme entstehen: Wo soll der String untergebracht werden und wie läßt er sich speichern? Da Abläufe dieser Art recht komplex sind, lohnt es sich, Wörter zu definieren, die die Einzelheiten für Sie erledigen.

Strings können in zwei Bereichen untergebracht werden – in den mit ALLOT reservierten Parameterfeldern der im Vokabular eingetragenen Wörter und in einem Bereich namens „Pad". Pad ist ein Zwischenspeicher, den FORTH auch für andere Aufgaben einsetzt. Bei jedem Vokabulareintrag oder jeder Interpretation eines FORTH-Wortes per Tastatur kann Pad überschrieben oder verschoben werden.

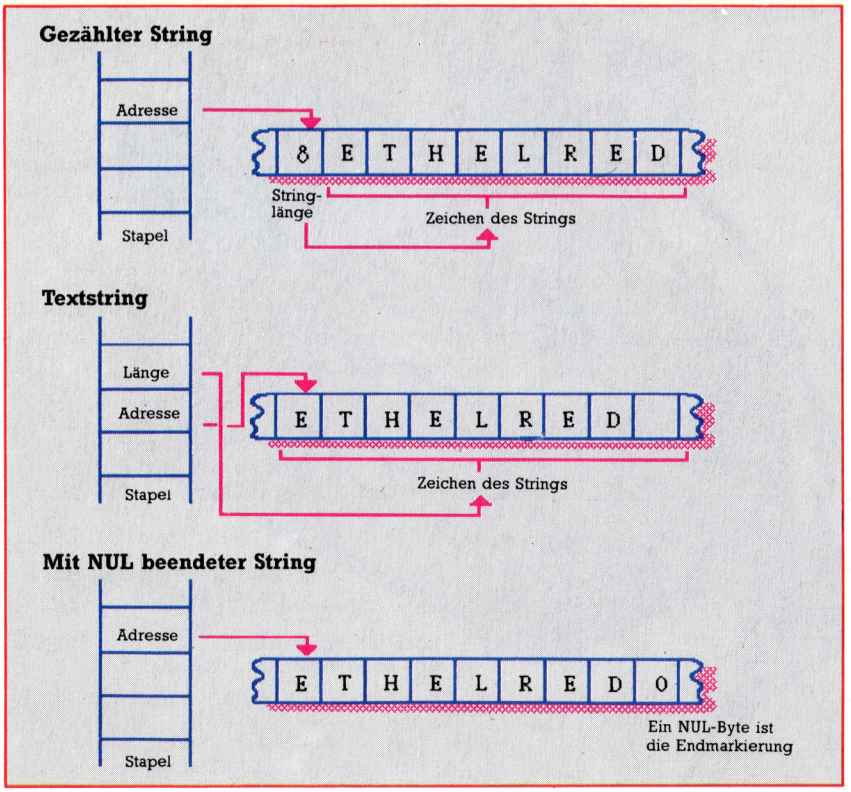
FORTH speichert Strings auf zwei Arten (als ASCII-Werte), die sich nur durch die Methode unterscheiden, mit der sie die Länge des Strings angeben.

Bei „gezählten Strings" wird die Länge (maximal 255 Bytes) in einem einzigen Byte unmittelbar vor den Zeichen gespeichert. Strings dieser Art lassen sich über die Adresse dieses Zählerbytes ansprechen. Zwar können Sie den String nicht auf den Stapel schieben, doch genügt es, seine Adresse dort abzulegen. Bei „Textstrings" wird die Länge nicht automatisch mit abgespeichert. Hier müssen Sie zwei Informationen angeben: die Adresse des ersten Zeichens und die Länge des Strings.

Der in Stringvariablen zur Verfügung stehende Platz ist begrenzt, da er im Parameterfeld liegt, dessen Länge mit ALLOT fest definiert wurde. Bei $VARIABLE müssen Sie daher nicht nur einen Anfangswert angeben, sondern auch die maximale Länge zukünftiger Zuordnungen. Dabei kann es vorkommen, daß die Länge der ersten Zuordnung für später eingegebene Strings nicht ausreicht.

Sehr praktisch ist das Wort ASCII. Es findet sich in den meisten FORTH-Versionen, gehört aber nicht zum Standard. ASCII schiebt den ASCII-Code des nachfolgenden Zeichens auf den Stapel. Bei einer Colon-Definition compiliert es für die angegebene Zahl ein numerisches Zeichen, das dann während der Laufzeit auf den Stapel geschoben wird.

```
32 CONSTANT BL ( ASCII-Code für ein
   Leerzeichen )
:ASCII ( —— ASCII-Code ) ( Zeichen im
   ASCII-Format )
   BL WORD ( nächstes Wort holen )
   1+ C@ ( und den ASCII-Code für sein
      erstes Zeichen holen )
   STATE @ IF ( IF gilt für die Colon-
      Definition )
      [ COMPILE ] LITERAL ( LITERAL ist
      unmittelbares Wort )
   THEN
;IMMEDIATE ( ASCII hat spezielle
   Compiliereigenschaften )
:$ VARIABLE ( Länge —— ) ( eingesetzt im
      Format: )
      ( Länge $VARIABLE Namen-
            initialisierer " )
   ( Im Parameterfeld wird 1 Byte für die )
   ( maximale Länge gefolgt von einem
   gezählten )
   ( String dieser Länge eingerichtet )
CREATE
255 MIN ( sicherstellen, daß der String
         nicht zu lang ist )
1 ALLOT ( für maximale Länge )
ASCII "WORD ( Länge, Initialisierer als
   gezählter String )
```



```
    DUP C @ ROT MAX ( Initialisierer, Max.
          Länge )
    DUP HERE 1— C! ( max. Länge
          einsetzen)
    HERE SWAP 1+ ALLOT ( Initialisierer,
          Addresse zur Speicherung )
    OVER C@ 1+ CMOVE ( Initialisierer ein-
        kopieren )
  DOES > (pfa —— pfa + 1 ) ( Wert als ge-
      zählten String auf Stapel schieben )
    1 +
```

Die folgenden Zeilen definieren $@ und $! für $VARIABLES. Der Stapel wird im Text-String-Format (Adresse und Länge) eingesetzt, um die Werte ansprechen zu können.

```
:$@ ( pfa + 1 für die Stringvariable ——
      Text-String )
    COUNT
;
:$! ( Text-String, pfa + 1 für die String
      Variable —— )
    ( kürzt den Wert, falls er für die Variable
      zu lang ist )
    DUP 1 — C@ ROT ( adr, pfa, max. Länge,
                          wirkliche Länge )
    MIN                ( adr, pfa+1, einge-
                          setzte Länge )
    OVER OVER SWAP ( addr, pfa+1, Länge,
                          Länge, pfa+1 )
    C!                 ( neue Länge spei-
                          chern )
    SWAP 1 + SWAP ( adr, pfa+2, Länge )
    CMOVE
    ;
;
```

Sie können nun Stringvariablen anlegen wie:

```
0 $VARIABLE ZIFFERN 0123456789"
```

(Die 0 wird automatisch auf 10 gesetzt, um die 10 vorhandenen Zeichen zu berücksichtigen. Danach ist die maximale Länge der Stringvariablen auf 10 festgelegt.)

```
ZIFFERN $ @ TYPE
```

(zeigt 0123456789 an)

```
255 $ VARIABLE XPAD XXX"
```

(ein stabilerer Pad für 255 Zeichen.)

Es wäre praktisch, wenn sich Stringwerte per Tastatur verändern ließen. Die Wörter " und $INPUT entsprechen den BASIC-Anweisungen LET A$ = „. . ." und INPUT A$.

```
:"      ( —— Adresse, Länge des Text-
String )
        ( außerhalb der Colon-Definition im
        Format — "text" eingesetzt )
    ASCII "WORD COUNT ( Text-String holen )
    XPAD $!           ( in XPAD speichern )
    XPAD $@           ( Adresse und Länge
                      bleiben unverändert)
    ;
```

Das Wort " erhält seinen String auf die gleiche Weise, wie bei Wörtern vom Typ VARIABLE ein neuer Name definiert wird. Diese Art Wörter können daher in Colon-Definitionen nicht funktionieren, lassen sich aber so einsetzen:

```
"Papagei" PET $! PET $@ TYPE
"Pinguin" PET $! PET $@ TYPE
```

Hier ein Wort, das Eingabebefehle imitiert:

```
:$INPUT ( pfa+1 der String Variable —— )
        ( wartet darauf, daß Sie einen
        String eingeben und kopiert ihn in
        die Variable)
    DUP DUP 1+   ( pfa+1, pfa+1, pfa+2 )
    SWAP 1—C@ 2—   ( pfa+1, pfa+2,
    max.Länge—2 )
    EXPECT
    SPAN @ SWAP C!
    ;
```

Das Programm informiert EXPECT, nicht mehr Zeichen zu holen, als Platz in der Variablen ist. Bei einer bestimmten FORTH-Version kann hinter dem String zusätzlich ein NUL-Zeichen eingesetzt werden. Zur Sicherheit wird von der maximalen Länge daher 2 abgezogen, da sonst im Vokabular der Kopfsatz des nächsten Wortes zerstört werden könnte.

In FORTH-79 und figFORTH schreibt EXPECT in jedem Fall ein oder zwei NUL. Diese Zeichen sind notwendig, da die älteren FORTH-Versionen kein SPAN enthalten. Dabei muß ein zusätzliches Wort die Zeichen bis NUL zählen und so die Länge des Strings feststellen.

Jetzt ist es möglich, das BASIC-Programm

```
10 PRINT "Wie heißt Du?"
20 INPUT A$
30 PRINT "Nett, Dich kennen-
   zulernen,";A$; "."
40 PRINT "Bis zum nächsten Mal! Tschüß!"
```

mit selbstdefinierten Befehlen auch in FORTH zu schreiben:

```
30 $VARIABLE NAME
:HALLO!
   ." Hallo! Ich bin der Computer. Wie heißt
     Du?" CR
   NAME $INPUT
   ." Das ist aber ein interessanter Name,"
     NAME
   $@ TYPE
   "." ( Punkt anzeigen ) CR
   ." Paß auf Dich auf, OK? Schönen Tag
       noch!"
   CR
   ;
```

Mit BASIC lassen sich schnell kurze Programme schreiben, denen alle Fähigkeiten der Sprache (Fließkomma, Arrays, Strings etc.) zur Verfügung stehen. Für längere Programme eignet sich BASIC jedoch nicht, da es immer an die niedrige Struktur der Sprache gebunden ist — z. B. die Zeilennummern. Sehen Sie sich im Vergleich einmal unser FORTH-Wort HALLO! an. Es ist einfach aufgebaut und gibt ein ungefähres Verständnis davon, wie $INPUT, $ @ und ähnliche Wörter arbeiten. Die genaue Funktion läßt sich immer über die darin eingesetzten Wörter feststellen. Für den Programmablauf brauchen Sie jedoch nur zu wissen, was ein Wort ausführt, d. h. welche Werte auf dem Stapel sein müssen und welche Ergebnisse dort abgelegt werden.

Geflügelte Worte

STATE (-- Adresse) ist eine Variable, die unmittelbare Wörter verstehen können. Wenn ihr Wert auf wahr (nicht-null) steht, werden sie bei der Compilierung einer Colon-Definition aktiviert.

WORD (End-Zeichen -- gezählter String) liest das nächste Wort aus dem Eingabepuffer der Tastatur. Auf diese Weise erhält CREATE beispielsweise die Namen neuer Befehle. Das End-Zeichen ist normalerweise 32 (Leerzeichen). Es können aber auch andere End-Zeichen eingesetzt werden, wie * in unserer bekannten Definition von $VARIABLE.

EXPECT (Adresse, max. Zeichenzahl --) wird für neue Tastatureingaben eingesetzt. Die Zeichen werden bis zu dem angegebenen Maximum eingelesen und an der angegebenen Adresse abgelegt.

SPAN (-- Adresse) Diese Variable wird von EXPECT gesetzt und zeigt an, wieviele Zeichen eingelesen wurden. Nur FORTH-83 verfügt über SPAN. Ältere FORTH-Versionen markieren das Ende mit NUL-Zeichen.

PAD (-- Adresse des Pad) CMOVE (Ursprungsadresse, Zieladresse, Zeichenzahl --) kopiert die angegebene Anzahl Bytes von der Ursprungsadresse an auf die Zieladresse im Speicher und wird hauptsächlich für Stringkopien eingesetzt. Hier ist jedoch Vorsicht geboten, wenn der Zielbereich hinter dem Ursprungsbereich liegt und beide sich überlappen.

CMOVE> (Ursprungsadresse, Zieladresse, Zeichenzahl --) Dieses Wort gibt es nur in FORTH-83. Es löst das Problem von CMOVE, indem es die Bytes in einer anderen Reihenfolge kopiert.

COUNT (gezählter String -- Adresse, Länge des Text-Strings) Der String bleibt der gleiche, doch das Format wandelt sich vom gezählten String zum Text-String.

Baumschule

Mit dem „Entscheidungsbaum" haben wir eine einfache Methode zum Abfragen und Bearbeiten von unterschiedlichen Bedingungen in unser Adventure-Spiel eingeführt. Diese Technik wird jetzt durch zusätzliche Programmteile optimiert.

Im letzten Kursabschnitt haben wir eine Reihe von Bedingungen abgefragt und die entsprechende Antwort auf dem Bildschirm dargestellt. Was aber, wenn wir nicht auf alle, sondern nur auf einige Fragen antworten wollen? ON – GOTO-Befehle allein reichen dann nicht aus, wenn wir am Schluß zu eindeutigen Werten kommen wollen, die zur Erzeugung einer richtigen Antwortmeldung dienen. Auch hier kann der Entscheidungsbaum weiterhelfen.

Für unser Beispiel mit vier Bedingungen – Mensch, Tier, männlich oder weiblich – haben wir diesen Baum entwickelt. Alle Antworten, bei denen der Anwender nicht eindeutig definiert, ob es sich um ein männliches oder weibliches Wesen handelt, werden nun zurückgewiesen. Der neue Baum ist dem alten ähnlich: Es gibt immer noch vier Stufen, und weiterhin werden auf jeder Stufe die in Array c gespeicherten Elemente abgefragt. Die Nummern der Knotenpunkte sind allerdings nicht mehr auf die gleiche Weise verknüpft wie vorher. Das liegt daran, daß ein Endpunkt jetzt nicht mehr unbedingt an den „Wurzeln" des Baumes liegen muß, sondern auch weiter oben auftreten kann – wie etwa der Knotenpunkt 9 im neuen Baum.

Um den Baum ins Programm zu integrieren, müssen wir zuerst die Punkte numerieren, an denen eine Entscheidungsmöglichkeit besteht (Punkt 1–6). Wird beim Durchlaufen des Baumes eine Nummer über 6 erreicht, sind wir an einem Endpunkt angelangt.

Als weitere Vereinfachung bekommen die Punkte, an denen eine bestimmte Handlung erfolgen muß – in diesem Fall Punkt 7 und 8 – die nächsthöheren Nummern. Jeder dieser Punkte steht für eine der beiden Prüfbedingungen. Die folgenden Nummern werden an die restlichen Endpunkte, die für die Entscheidung unwichtig sind, vergeben.

Der Grund für diese Ordnung wird später noch deutlicher, wir wollen den Baum zuerst einmal ins Programm einbauen. Dazu brauchen wir ein neues Array t(6,1) – beim Spectrum t(6,2) – das Informationen über die Verzweigungen speichert und entsprechend der getroffenen Entscheidung zum nächstfolgenden Punkt hinführt. Die Werte des Array t sind in der Tabelle aufgelistet.

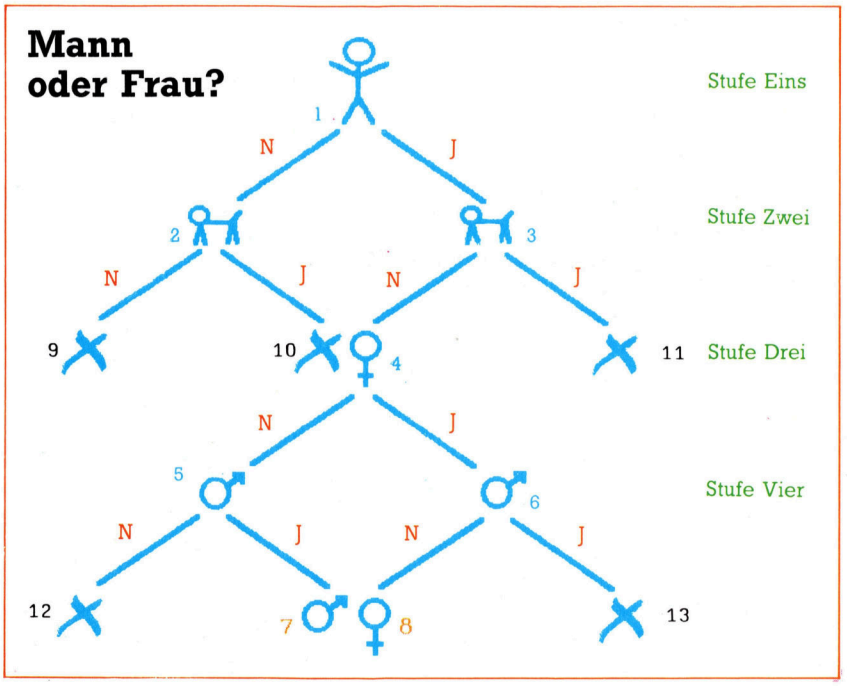
Geben Sie das folgende Programm ein. Es baut auf den Zeilen 10 bis 80 aus dem letzten Abschnitt und den Hauptroutinen des „Dog and Bucket"-Programms auf. BASIC-Dialekte der Hauptroutinen beim Spectrum, Commodore 64 und Acorn B wurden bereits beschrieben. Zwischen Zeile 20 und 30 gibt es einige zusätzliche Zeilen, neu ist auch die Programmzeile 500. Gemeinsam sorgen diese Erweiterungen für das Einlesen der Daten in das Array t, das durch den neuen Befehl DIM t(6,1) in Zeile 10 erzeugt wird.

Wenn beim Durchlaufen dieses Baumes nicht wahrheitsgemäß von einer Frau oder einem Mann geantwortet wird, führen die Verzweigungspunkte zu Endwerten, die das Programm als ungültige Aussagen erkennt. Der Baum ähnelt zwar dem Beispiel im letzten Abschnitt, er besitzt jedoch nicht nur auf Stufe 5 seine Endpunkte. Die Gabelungen mit alternativer Antwortmöglichkeit sind blau numeriert, Endpunkte in Rot und in Schwarz. Die roten Ziffern bezeichnen gültige Resultate (also „weiblicher bzw. männlicher Mensch").

Mensch Tier

Frau Mann

Ungültig



Baum-Tabelle

Das Array t(6,1) – beim Spectrum, der keine 0-Elemente in Arrays erlaubt, t(6,2) – enthält die Daten der Baumstruktur. Wenn etwa Stufe 4 erreicht ist und die Bedingung in c(4) als falsch erkannt wird, erfolgt eine Verzweigung zur Knotenpunkt-Nummer, die im Array t(4,0) steht. Ist die Bedingung wahr, wird zur Nummer in t(4,1) gesprungen. Beim Spectrum sind die entsprechenden Array-Indizes t(4,1) und t(4,2).

Knotenpunkt	Falsch	Wahr
1	2	3
2	9	10
3	4	11
4	5	6
5	12	7
6	8	13

```
10 h$="human ":a$="animal ":m$="male ":f
$="female ":q$="Are you ": DIM c(4),t(6,
1)
20 GOSUB 4050: REM clear the screen
22 REM set up tree array
24 c=6: REM this is number of choice nod
es
26 FOR x=1 TO c: READ t(x,1),t(x,0): NEX
T x
30 REM set up the four variables
40 PRINT q$;h$;: INPUT i$: c(1)=ABS(i$="
y" OR i$="Y")
50 PRINT q$;a$;: INPUT i$: c(2)=ABS(i$="
y" OR i$="Y")
60 PRINT q$;f$;: INPUT i$: c(3)=ABS(i$="
y" OR i$="Y")
70 PRINT q$;m$;: INPUT i$: c(4)=ABS(i$="
y" OR i$="Y")
80 PRINT
500 DATA 3,2,10,9,11,4,6,5,7,12,13,8
4000 REM
4010 REM low level system subroutines
4020 REM
4030 REM clear the screen
4040 REM
4050 CLS: RETURN
4060 REM
4070 REM beep
4080 REM
4090 PRINT CHR$(7);: RETURN
4100 REM
4110 REM get a character from the keyboa
rd
4120 REM
4130 i$=INKEY$: IF i$="" GOTO 4130
4140 RETURN
```

Zum Durchlaufen des Baumes und der Anzeige der Meldungen dienen diese Zeilen:

```
90 GOSUB 210: REM call tree sort routine
100 PRINT "Press a key to continue..."
110 GOSUB 4130: REM get a character
120 GOTO 40
200 REM sort tree
210 n=1: k=1: REM start at node 1, level
 1
220 n=t(n,c(k)):k=k+1: IF n<=c THEN 220
230 ON n-c GOTO 250,260
240 PRINT "Invalid input...Please try ag
ain": RETURN
250 PRINT "You're a man...": RETURN
260 PRINT "You're a woman": RETURN
```

Beim Starten des Programms merken Sie schnell, daß diesmal nur wahrheitsgemäße Aussagen männlicher bzw. weiblicher Personen akzeptiert werden. Im Unterschied zum vorherigen Programm wissen wir diesmal nicht, wieviele Stufen bis zum Erreichen eines Endpunktes durchlaufen werden müssen – die einfache Schleife „FOR n TO Zahl der Stufen" muß durch Einführung der Variablen k in Zeile 140 ersetzt werden, in der die aktuelle Stufe gespeichert wird.

Nachdem n für den Startpunkt (1) festgelegt ist, durchlaufen wir den Baum mit Zeile 150. Dabei wird die Formel n=t(n,c(k)) zum Heraussuchen des nächsten Verzweigungspunktes aus dem Array t verwendet. Solange noch kein Endpunkt erreicht ist, die Nummer des aktuellen Punktes also noch kleiner oder gleich c (Anzahl der Knotenpunkte mit Alternativen) ist, wird wieder zum Anfang der Zeile verzweigt und der Ablauf wiederholt.

Sie erkennen jetzt, warum die Knotenpunkte in der oben beschriebenen Weise numeriert sind. Teilt man die Endpunkte in zwei Gruppen – eine mit den zulässigen Werten männlicher bzw. weiblicher Personen, eine mit den unzulässigen Aussagen –, dann können wir einen ON...GOTO-Befehl zur Erzeugung der entsprechenden Meldungen verwenden.

Bisher haben wir uns auf Strukturen konzentriert, die etwas Gemeinsames haben – auf allen Stufen des Baumes wurden die gleichen Bedingungen geprüft. Das hat unser Problem vereinfacht – die abgefragte Bedingung mußte nicht bei jedem Entscheidungspunkt neu ermittelt werden. An einem Knotenpunkt auf Stufe 3 war klar, daß die zu prüfende Bedingung in c(3) gespeichert ist. Nicht immer ist es so einfach. Bei mehreren Prüfbedingungen wird die Baumstruktur komplexer. Die beste Methode ist, das Problem in verschiedene Teile zu zerlegen, wie wir es mit dem Hauptteil unseres Programms getan haben. Die Variation der Spielfiguren im „Dog and Bucket"-Programm steht bis jetzt auf einer niedrigen Stufe – Unterprogramme zum Holen, Fallenlassen, Essen, Trinken, Geben, Erhalten und Werfen von Gegenständen fehlen noch. Auch die Information des Spielers muß noch realisiert werden.

Die Unterprogramme „Fallenlassen/Werfen/Geben" können zu einer Gruppe zusammengefaßt werden – sie alle arbeiten mit einer Figur, die einen Gegenstand hat und ihn abgibt. Um das entsprechende Unterprogramm zu wählen, sind Prüfbedingungen erforderlich, etwa „Ist jemand im Zimmer?" und, falls ja, „Will derjenige den Gegenstand annehmen?". Auch die Stimmung der Figuren muß getestet werden, um beispielsweise zu wissen, ob die jeweilige Figur in absehbarer Zeit einen Gegenstand werfen wird. Ein gewisser Zufallsfaktor und weitere Hilfsroutinen sind ebenfalls nötig.

BASIC-Dialekte

Sinclair Spectrum

Die Abweichungen für Zeile 4000 bis 4140 sind bereits auf Seite 1648 abgedruckt. Im neuen Programmteil müssen Spectrum-Besitzer zusätzlich diese Änderungen vornehmen:

```
10 h$="human ":a$="animal ":m$=" male ":
   f$="female ":q$="Are you ": DIM c(4) ,t(6, 2)
40 PRINT q$;h$;: INPUT i$: c(1)=ABS(i$=" y" OR
   i$="Y")+1
50 PRINT q$;a$;: INPUT i$: c(2)=ABS(i$=" y" OR
   i$="Y")+1
60 PRINT q$;f$;: INPUT i$: c(3)=ABS(i$=" y" OR
   i$="Y")+1
70 PRINT q$;m$;: INPUT i$: c(4)=ABS(i$=" y" OR
   i$="Y")+1
```

Moderne Sprachen

? Warum wird noch immer über den Computerunterricht an Schulen diskutiert?

Traditionell wird von Schülern im Unterricht passives Verhalten erwartet. Übernimmt das Kind (wie beispielsweise beim Programmierunterricht) eine aktive Rolle, führt das zu Auseinandersetzungen.

Seit Ende der 70er Jahre ist Computerunterricht Bestandteil der Lehrpläne. Bei Eltern wie Schülern erfreut er sich ständig wachsender Beliebtheit. Nach Einschätzungen von Eltern (die Befragung wurde Anfang der 80er Jahre in England durchgeführt) steht die Wichtigkeit des Computerunterrichts als Fach an dritter Stelle nach Mathematik und Englisch. Diese Begeisterung wird allerdings nicht von allen Lehrern geteilt.

Manche Lehrer sind der Überzeugung, daß Computer in alle Unterrichtsfächer einbezogen werden sollten und nicht als getrenntes Fach zu betrachten seien. Aus diesem Grunde gibt es an manchen Schulen keinen richtigen Computer-Unterricht, nicht zuletzt auch deshalb, weil einige Schulen Probleme bei der Beschaffung der erforderlichen Lernmittel haben.

Oftmals fällt jedoch Kindern das Begreifen von Konzepten und Arbeitsabläufen leichter, wenn sie mit dem Computer arbeiten. Dieses Argument allein rechtfertigt den Programmierunterricht an Schulen.

Bis vor kurzem litt die Gestaltung dieses Unterrichtsfaches dadurch, daß die meisten Rechner mit BASIC arbeiteten. Aufgrund seines abstrakten Inhalts können Kinder unter zwölf BASIC kaum verstehen. Deshalb begann man an einigen Schulen mit dem Computerunterricht erst ab der 6. Klasse. Dazu kommt, daß die Computerindustrie dem Einsatz von BASIC im Unterricht skeptisch gegenübersteht. Grund: BASIC ist nicht strukturiert und kann den Schüler zu schlechten Programmiertechniken verleiten. Leider haben sich viele Leser mit eben dieser Materie (BASIC) vertraut gemacht und propagieren die Sprache, obwohl strukturierte Programmiersprachen wie LOGO, COMAL und PROLOG verfügbar sind.

? Welche Alternativen gibt es denn zu BASIC, das schlechte bzw. abstrakte Programmiertechniken vermittelt?

Die Einführung strukturierter BASIC-Versionen wie etwa das Acorn-BASIC hat einiges an Fehlern wieder wettgemacht. Der eigentliche Programmierunterricht ist auf die drei zuvor genannten Sprachen gerichtet. LOGO haben wir bereits in einer speziellen Serie behandelt. Die Sprache wird in einigen Ländern von Logo Users Groups und der Young Persons Logo Association gefördert. Alle wichtigen Microcomputerhersteller haben inzwischen Versionen der Sprache veröffentlicht.

Borge Christensen entwickelte am State Teachers College im dänischen Tondern die Programmiersprache COMAL (COMmon Algorithmic Language). Bei COMAL handelt es sich um eine strukturierte BASIC-Version. Christensen und seine Kollegen stellten fest, daß die von den Schülern geschriebenen BASIC-Programme unübersichtlich und fast unbrauchbar waren. „Nachdem ich mich durch eine Kilometer von BASIC-Programmen gearbeitet hatte", sagte er, „stellte ich fest, daß etwas faul war im Staate Dänemark." Christensens Argumente für COMAL haben die dänische Regierung dahingehend beeinflußt, daß COMAL an dänischen Schulen eingeführt wurde, und auch in Irland wird dieser Sprache der Vorzug vor BASIC gegeben. In England ist das Interesse an COMAL ebenfalls gewachsen. In einer Schule in Süd-London schreiben Schüler ein anspruchsvolles Satelliten-Kontrollprogramm in COMAL.

PROLOG (PROgrammieren in LOgic) wird von einer kleinen, in England arbeitenden Pädagogengruppe genutzt. Bei Verwendung von PROLOG kann ein in logischen Begriffen dargestelltes Problem leicht in ein Computerprogramm umgesetzt werden. In der Londoner Park House School arbeitete man an einem Projekt, mit dem Materialien für den Logikunterricht als Programmierdisziplin für 10- bis 13-jährige entwickelt wurde. Einige erlernten das Programmieren fließend, andere konnten sich mit dem Klassenstandard durchaus messen, schrieben aber kaum eigene Programme. In höheren Klassen wurden bessere Ergebnisse erreicht, nachdem ein allgemeiner Computerunterricht an der Schule eingeführt worden war.

? Welche Vorteile bietet der LOGO-Unterricht für Kinder?

Bei LOGO wurden intensive Forschungen betrieben, um Kritiker zu widerlegen, die behaupteten, daß Kinder durch den LOGO-Unterricht kaum Vorteile hätten. Die amerikanische National Science Foundation initiierte das Brookline-LOGO-Projekt, um die Anwendung von LOGO durch Kinder zu studieren. In der ersten Phase arbeitete man mit elfjährigen Schülern in einem Computer-Laboratorium und beobachtete ihre Arbeitsweisen und Aktivitäten. In der zweiten

Phase installierte man Computer an Schulen und entwickelte auch Lehrmaterial. Interessantes Ergebnis der ersten Phase war, daß alle Kinder erfolgreich mit LOGO arbeiteten, egal, wie ihre Leistungen in anderen Fächern waren. Dieses Ergebnis schlug sich auch in anderen LOGO-Projekten nieder.

? Wurden weitere Untersuchungen durchgeführt, um Erfolg oder Mißerfolg des Programmierunterrichts aufzuzeigen?

Die New York Academy of Sciences und drei Schulen in New York City initiierten ein Projekt, das den LOGO-Unterricht an Schulen unterstützen sollte. Computer wurden in Klassenzimmern aufgestellt. Die Projektmitarbeiter besuchten wöchentlich die Räumlichkeiten, und Lehramtsanwärter nahmen an Weiterbildungsseminaren teil. Michael Tempel, der Projektleiter, stellte eine Reihe von Übereinstimmungen mit dem Brookline-Projekt fest. Eine der Grunderkenntnisse war, daß der LOGO-Unterricht zu einer positiven Interaktion zwischen Schülern führte. Kinder, die in anderen Fächern weniger erfolgreich waren, arbeiteten ausgezeichnet mit LOGO. „Der pädagogische Vorteil war offenkundig", so Tempel. „Kinder engagierten sich und machten intellektuelle wie soziale Fortschritte. Wir haben bedeutsame Veränderungen im Verhältnis der Kinder zur Schule und zum Lernen an sich feststellen können."

Am Center for Children and Technology in New York wurde ein 30 Monate währendes Projekt durchgeführt, um die Übertragbarkeit von Problemlösungsstrategien auf andere Lehrstoffe zu untersuchen und so die Auswirkungen von LOGO-orientierter Arbeit auf soziale Interaktion festzustellen. Planen wird als wichtige kognitive Fähigkeit betrachtet. Untersuchungen haben ergeben, daß viele Kinder kaum Voraussetzungen dafür mitbringen. Mit Hilfe der LOGO-Prozeduren können Programmierprobleme während des Planungsstadiums in kleinere Einheiten aufgelöst werden. Die Untersuchung beschäftigt sich mit den Auswirkungen, die das Programmieren auf das Lernvermögen hat.

Interessant war die Beobachtung, auf welche Weise Computer zur Kooperation und Interaktion der Kinder beitragen. „Es gab mehr aufgabenbezogene Interaktion während der Arbeit mit dem Computer als bei anderen Aktivitäten, in die kein Lehrer einbezogen war", merkte der Projektleiter an. „Am interessantesten aber war, daß es auch mehr aufgabenbezogene Gespräche während des Umgangs mit dem Computer gab." Hier erfolgte also ein wesentliches Stimulans durch den Computer als Kommunikationspartner. Ähnliche Erkenntnisse wurden an der Lamplighter School in Texas gesammelt, der Texas Instruments 50 Microcomputer mit TI-LOGO zur Verfügung stellte.

Die Implementierung verschiedener Sprachen auf Microcomputern trägt nur langsam zur Verbreitung des Programmierunterrichts an Schulen bei. Einige Sprachen wie PROLOG können allein aufgrund ihres Aufbaus sinnvoll angewandt werden. Andere, etwa COMAL und LOGO, sind empfehlenswert, weil sie eine Einführung in diszipliniertes und leicht überschaubares Programmieren geben.

? Was sind Hinderungsgründe für die praktische Anwendung dieser umfangreichen Erkenntnisse?

Die Schulbehörden widersetzen sich, Unterrichtsstunden zum Programmieren einzuplanen, wenn die Vorteile für den Lehrplan nicht deutlich sind. Ebenso stehen nur wenig finanzielle Mittel und folglich nur wenig Hardware zur Verfügung. Überdies ist es schwer, eine kontrollierte Studie anzufertigen, da die Einrichtung einer Kontrollgruppe in einer Schule bedeutet, daß nur eine gewisse Gruppe von Kindern an den Computer kommt. Der Vergleich mit einer Schülergruppe an anderen Schulen ist aufgrund unterschiedlicher Lehrmethoden einzelner Lehrer schwer.

? Welche allgemeinen Vorteile lassen sich aus dem Programmierunterricht ableiten und erkennen?

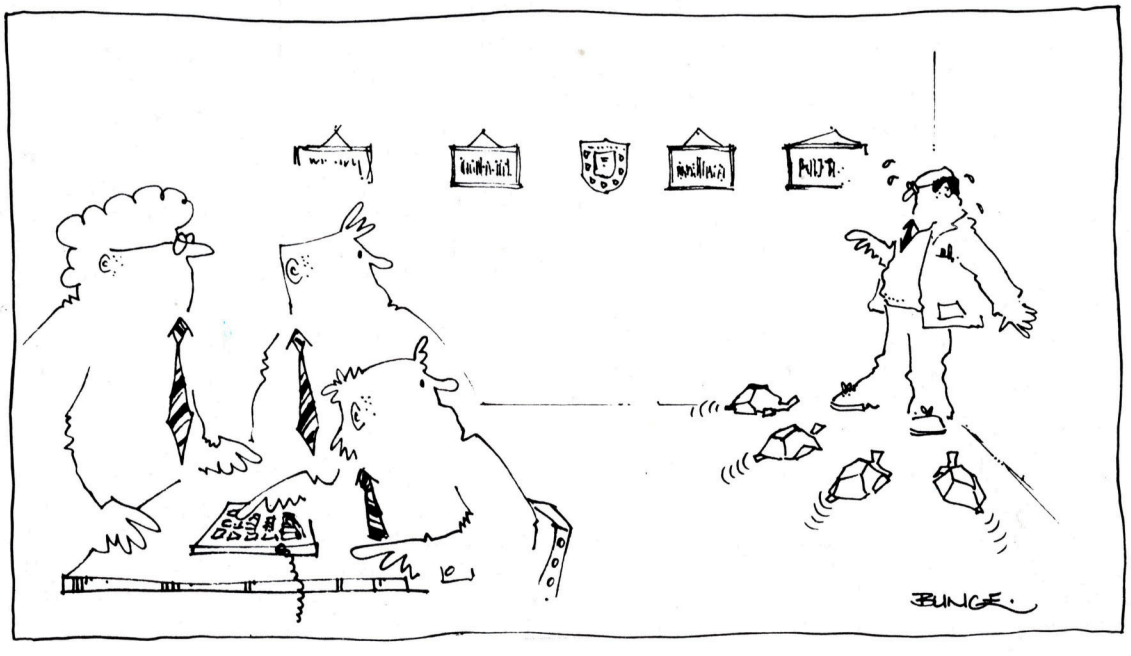
LOGO bietet als Programmiersprache für kleinere Kinder den Zugang zu aufregender, neuer Aktivität. Bisher dominiert noch immer BASIC in den oberen Klassen. Deshalb ist es wichtig, LOGO im gesamten Schulsystem anzuwenden. Andernfalls wird die Verwirrung im Computerunterricht noch größer. Denn sobald gute LOGO-Programmierer in höhere Klassen aufrücken, sehen sie sich plötzlich mit BASIC konfrontiert. Zwingend ist deshalb die Standardisierung der Sprachen an Schulen, ebenso wie die Standardisierung der Computer an sich.

Der Programmierunterricht bietet dem Schüler ungeheure Vorteile. Kinder haben die Möglichkeit, sich mit einem anderen Medium auseinanderzusetzen und werden dazu ermutigt, präzise und organisiert zu denken. Schließlich ergänzt dieses Fach viele andere Lernbereiche und trägt zu einem weiteren Verständnis des Lehrstoffes bei. Wenngleich es noch viel in diesem Bereich zu erforschen gibt, sind die bisherigen Ergebnisse positiv und sehr ermutigend.

? Neben der Entwicklung logischer Problemlösungstechniken – was kann ein Kind denn mit dem in der Schule erworbenen Programmierwissen anschließend praktisch anfangen?

Ist ein Kind erst einmal mit dem Programmieren vertraut, gibt es eine Fülle von Anwendungen. Eine Kindergruppe hat Turtles zum Choreographieren benutzt. Die Kinder haben die Turtles so programmiert, daß sie sich in Formation bewegen, und diese Bewegungen dann in Tanz umgesetzt. LOGOs Wort- und List-Programmiermöglichkeiten wurden dazu genutzt, Texte zu generieren. Schüler haben die Möglichkeit, Verzweigungen einzubauen und so den Verlauf der Geschichten zu modifizieren. Dazu kommen die beachtlichen Soundmöglichkeiten von LOGO. So, wie SQUARE zur Definition von PATTERN benutzt wird, kann NOTE CHORD definieren, CHORD dann TUNE usw. Kinder können also den Computer für musikalische Zwecke einsetzen.

Interessant ist der Einsatz von LOGO bei der Erstellung von Grafiken. Hier bieten sich tausendfache Möglichkeiten an. Gerade bei der Einführung in die Geometrie ist die Turtle ein faszinierender Lehrer. Schnell erkennen die Kinder mathematische Zusammenhänge und können diese mit Bleistift und Lineal im Heft nachvollziehen, nachdem bereits alles auf dem Bildschirm zu sehen war.



Ein Paukenschlag

Seit Beginn des „Computerkurs" hat die Microcomputer-Industrie eine dramatische Entwicklung erlebt. Viele Hard- und Software-Hersteller haben den Druck stagnierender Absatzzahlen nach einer explosiven Wachstumsphase zu spüren bekommen.

Schneider CPC 6128

Schneider CPC 464

Schneider CPC 664

Selbst große Namen wie Acorn und besonders Sinclair sahen sich vor einiger Zeit wirtschaftlich am Rande des Abgrundes. Nur ein Unternehmen, nämlich Amstrad, entwickelte sich anders, als der Trend vermuten ließ. Der Grund, warum Amstrad, erfolgreicher internationaler Unterhaltungselektronik-Hersteller, sich in den Microcomputer-Markt wagte, ist noch immer reine Spekulation. Man erzählt, daß das Unternehmen Ende 1983 einen Posten Fernsehmonitore hatte. Was sollte man damit tun? Die Lösung war einfach: Man legte einen der Marketingerfolge neu auf. Amstrad hatte sich mit der Entwicklung von kompletten HiFi-Systemen einen Namen geschaffen – sogenannte „Turmsysteme", die 1978 auf den Markt kamen. Warum sollte man die Monitore nicht als Komponenten eines Microcomputer-Komplettsystems verwenden? So brachte das Unternehmen im April 1984 den CPC 464 (in Deutschland als Schneider CPC 464 angeboten) auf den Markt, der bis dahin von Giganten wie Sinclair und Commodore Business Machines beherrscht wurde.

Der CPC 464 war ein Marketing-Geniestreich. In den Läden stellte er seine Konkurrenten förmlich in den Schatten. Neben ihm wirkte der Spectrum wie ein Spielzeug und, verglichen mit dem C 64, bot er ein eingebautes Cassettenlaufwerk und einen monochromen Monitor für nur wenig mehr Geld.

Innerhalb von 18 Monaten hatte Amstrad in England einen Marktanteil von 25 Prozent. Hinzukam, daß dieser Marktanteil zu einer Zeit vergrößert wurde, als andere Microcomputerhersteller Absatzschwierigkeiten hatten.

Amstrads klare und konsequente Marketing-Strategie ist allein Alan Sugar zu verdanken, der das Unternehmen 1968 als „Amstrad Consumer Electronics" gründete. Schwerpunkt damals: Großhandel mit Autozubehör und anderen elektronischen Komponenten. Das Unternehmen entwickelte sich erst richtig 1980, kurz nach Eroberung des HiFi-Marktes. In diesem Jahr war Amstrad zu einem Multi mit einem Jahresumsatz von über 36 Millionen Mark geworden. Der Umsatz verdoppelte sich anschließend jährlich. Ergebnis im Juni 1984: Runde 35 Millionen Umsatz.

Bevor Amstrad auf den Markt kam, resultierte der gute Ruf der englischen Microcomputerindustrie aus der Leistung, die beispielsweise Clive Sinclair und Acorns Curry und Hauser bei der Entwicklung der Rechner vorzuweisen hatten. Alan Sugar repräsentierte jedoch eine neue Art von Persönlichkeit der Computer-Industrie, da er aus der Marketingabteilung kommt.

Marktlücke gefüllt

In der kurzen Zeit, die sich Amstrad im Computermarkt befindet, hat das Haus einige kühne Schritte gewagt. Der bisher bemerkenswerteste wurde im Frühling 1985 getan. Im April kündigte das Unternehmen eine verbesserte 464-Version, den CPC 664 an, der mit einem 3-Zoll-Laufwerk statt Cassettenlaufwerk ausgestattet ist und über eine gekürzte CP/M-Version verfügt. Kaum war dieses Gerät im Angebot, kündigte Amstrad den CPC 6128 an, der die doppelte Speicherkapazität des 664 hat (und mit einem echten CP/M ausgestattet ist) – zu einem noch niedrigeren Preis. Wenngleich Amstrad mit diesem CP/M-Rechner ein Zeichen gesetzt hat, dürften sich doch viele Konsumenten, die den CPC 664 kauften, betrogen fühlen. Es schien fast so, als ob das Unternehmen mit sich selbst konkurrierte.

Amstrads jüngstes Kind ist der PCW 8256, der im August 1985 auf den Markt kam. Er bietet eine Textverarbeitung (256 KByte, Monitor, Diskettenstation und Matrix-Drucker) für einen sehr niedrigen Preis.

Platz für Manöver

Einfache Utility-Programme, wie das Variablennamen-Suchprogramm, können vollständig in BASIC geschrieben werden. Für kompliziertere und umfangreichere Hilfsprogramme jedoch müssen wir auf die Maschinensprache zurückgreifen.

Um das Variablen-Suchprogramm zu benutzen, haben wir es mit dem zu bearbeitenden Programm verbunden. Bei dieser Methode war die einzige vom Betriebssystem benötigte Information die Adresse, an der das BASIC-Programm startet. Das Ende des zu durchsuchenden Programms wurde gefunden, indem die Zeilennummern mit der niedrigsten Nummer des Hilfsprogramms verglichen wurden.

Das Hilfsprogramm dient zum Austauschen von Variablen. Stellen Sie sich vor, wieviel Zeit ein solches Programm spart, wenn Sie in einem Programm einen Variablennamen verwendet haben und dann feststellen, daß er illegal ist. Ähnlich verhält es sich, wenn Sie jemandem ein Programm geben wollen, er aber die Variablennamen nicht problemlos interpretieren kann.

Die drei Computer, mit denen wir uns hier befassen – Acorn B, Commodore 64 und Spectrum – verwenden verschiedene Zeiger, um dem Betriebssystem und dem BASIC-Interpreter anzugeben, wo ein BASIC-Programm und seine Variablen abgelegt werden sollen.

Beim Acorn B gibt es vier wichtige Zeiger: PAGE und TOP, die die Start- und Endadresse des BASIC-Programms angeben; LOMEM, welches die Startadresse der BASIC-Variablen angibt; und schließlich HIMEM, das die Endadresse des BASIC-Speicherbereichs angibt. Diese vier Zeiger werden als „eingebaute" BASIC-Variablen gespeichert, so daß wir ihre Werte mit einfachen BASIC-Anweisungen lesen und ändern können. Haben wir ein BASIC-Programm im Speicher und wollen ein anderes hinzufügen, ändern wir PAGE auf einen Wert höher als TOP – unter Verwendung des Befehls OLD zum Zurücksetzen von TOP und LOMEM – und können dann das neue Programm hinzufügen, ohne das vorhandene zu beeinflussen. Zwischen den Programmen kann dann umgeschaltet werden, indem PAGE und HIMEM neue Werte zugewiesen werden.

Originalwerte kopieren

Ist das Hilfsprogramm einmal gestartet, beziehen sich die Werte der Zeiger auf dieses Hilfsprogramm. Damit die Utility den Anfang und das Ende des zu bearbeitenden Programms finden kann, müsssen wir die Originalwerte in einen Speicherbereich kopieren, der beim Wechsel zwischen den beiden Programmen nicht geändert wird. Eine andere Methode, das Programmende zu finden, ist die Benutzung der Endmarkierung, die der BASIC-Interpreter einfügt. Ist lediglich ein Byte mit dem Wert 128 (oder höher), das direkt dem RETURN-Zeichen am Ende der letzten Programmzeile folgt. Dieses und das darauffolgende Byte werden als HI- und LO-Bytes der nächsten Zeilennummer interpretiert. Da das HI-Byte dieser Nummer 128 oder größer ist, ergibt das eine Zeilennummer 32768 (256 × 128) oder mehr. Da die höchste gültige Zeilennummer 32767 ist, können wir sicher sein, das Ende des Programms gefunden zu haben.

Sieben Zeiger

Der Commodore 64 verwendet sieben in der Zero Page gespeicherte Zeiger, die sich auf verschiedene Teile des BASIC-Programmbereiches beziehen. TXTTAB bei den Adressen 43 und 44 zeigen auf den Start des BASIC-Programms. VARTAB, ARYTAB, STREND, FRETOP und FRESPC bei den Adressen 45 bis 54 zeigen auf die verschiedenen Abschnitte der Variablentabelle. MEMSIZ bei den Adressen 55 und 56 zeigt auf das Ende des BASIC-Speicherbereichs. Um einen separaten Bereich für ein BASIC-Programm zu schaffen, können diese Zeiger dann mittels des POKE-Befehls geändert werden.

Beim Commodore 64 wird das Ende eines BASIC-Programms durch zwei Bytes mit dem Inhalt 0 gekennzeichnet, die direkt dem Null-Byte folgen, das das Ende der letzten Zeile im Programm markiert.

Das Schreiben des Hilfsprogramms für den Spectrum ist erheblich komplizierter. Anstelle eines separaten Bereichs für das BASIC-Programm gibt es nur einen zusammenhängenden Speicherblock, der nicht nur das BASIC-Programm nebst Variablen, sondern auch den gesamten Arbeitsbereich des Betriebssystems und des BASIC-Interpreters beinhaltet. Durch diese Speicherorganisation ist es schwierig, zwei BASIC-Programme im Speicher aufzubewahren. Daher kopieren wir unser Programm über RAMTOP und bearbeiten es dort. Dadurch löst sich jedoch nicht das Problem, das Programm nach den Änderungen wieder in den Hauptspeicher zu bekommen. Hierfür benötigen wir eine Maschinenroutine.

Wenn Sie Utilities für den Spectrum programmieren wollen, ist das Buch „The complete Spectrum ROM Disassemly" von Dr. Ian Logan und Dr. Frank O'Hara zu empfehlen. Hier wird die Funktionsweise sämtlicher ROM-Routinen ausführlich erklärt.

Zwei der wichtigsten Unterroutinen im ROM für die Verwendung in Hilfsprogrammen sind die Routinen, die Platz im Arbeitsspeicher bereitstellen oder reservieren. Im nächsten Artikel werden wir uns mit diesen Routinen beschäftigen.

Im Handbuch des Spectrum stehen sehr gute Informationen, wie ein BASIC-Programm gespeichert wird und wofür verschiedene Speicherbereiche verwendet werden. Trotzdem ist es schwierig, Hilfsprogramme ohne Maschinenspracheroutinen aus dem ROM zu schreiben.

Experimentieren mit BASIC

Ein Programm läßt sich während der Ausführung ändern, doch sollte es zuvor gespeichert werden, da Änderungen oft einen Systemabsturz bewirken. Verwenden Sie ein MONITOR-Programm, das eine Änderung der Speicherinhalte ermöglicht. Fügen Sie am Programmanfang einige zusätzliche REM-Anweisungen ein und probieren Sie zuerst unsere Vorschläge:

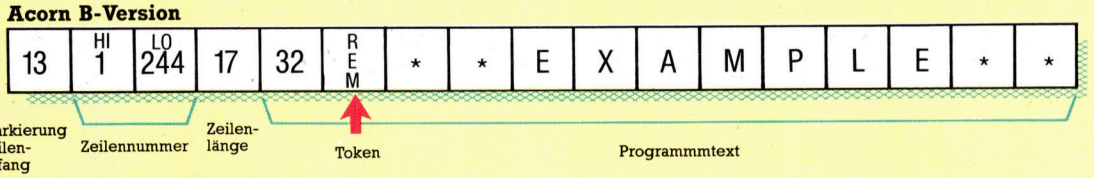
- Finden Sie den Beginn des BASIC-Speicherbereichs und untersuchen Sie das MONITOR-Programm im Speicher, bis Sie die Programmzeilen identifizieren können.
- Ändern Sie die Werte der Bytes nach einer REM-Anweisung, beenden Sie das Programm und listen Sie die geänderte Zeile.
- Versuchen Sie, einen Wert größer als 127 in eine REM-Zeile zu legen – listen Sie die geänderte Zeile.
- Ändern Sie die Bytes der Zeilennummern einer Zeile – das hat unvorhersehbare Ergebnisse zur Folge, besonders wenn die neue Nummer außerhalb des ursprünglichen Bereichs liegt.
- Sie können das Zeilenlängen-Byte ändern, doch sollten Sie an der angegebenen Position eine neue Endmarkierung setzen. Versuchen Sie zuerst einmal, den Wert einfach nur zu verkleinern.
- Beim Commodore 64 können Sie das Link-Adreß-Byte ändern: Versuchen Sie die Link-Adresse einer Zeile durch die der folgenden Zeile zu ersetzen, und listen Sie das Programm auf.
- Sie können auch mit Hilfe Ihres Handbuches den Variablen-Speicherbereich untersuchen. Er beginnt normalerweise dort, wo der BASIC-Programmtextbereich endet. Es gibt insgesamt bis zu sechs verschiedene Variablentypen, jeder mit seinem eigenen Speicherformat: numerische Variablen, numerische Arrays, Integer-Variablen, Integer-Arrays, String-Variablen und String-Arrays.
- Sie können auch die Werte der internen Darstellung (Token) einer Zeile ändern: auf diese Art werden die Befehlswörter geändert.

Wie BASIC-Programme gespeichert werden

Die meisten Microcomputer speichern den BASIC-Programmbereich im gleichen Format ab. Der Programmtext wird in mehr oder weniger geänderter Form gespeichert, wobei BASIC-Schlüsselwörter durch 1-Byte-Codezahlen, genannt „Tokens", ersetzt werden.

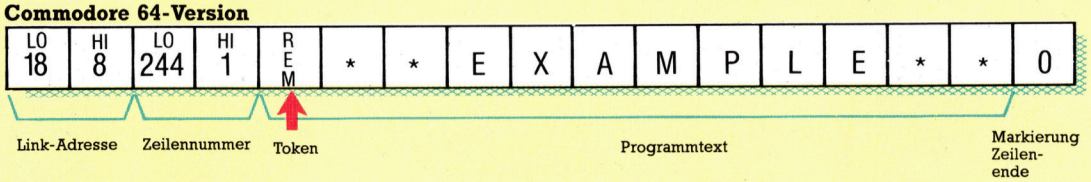
500 REM ** EXAMPLE **

Acorn B-Version



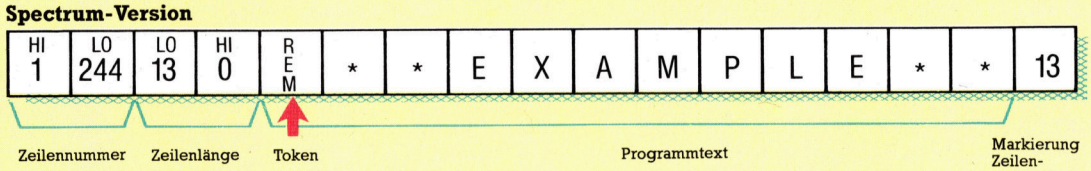
In diesem Format ist 13 (ASCII für RETURN) die Markierung für einen Zeilenanfang. Normalerweise wird dieser Code am Ende einer Zeile plaziert. Die Zeilenlänge wird durch ein Byte dargestellt.

Commodore 64-Version



Die Link-Adreß-Bytes enthalten die Zwei-Byte-Adresse des ersten Bytes der nächsten Programmzeile. Der BASIC-Programmtextbereich beginnt bei Adresse 2049. Diese Zeile ist 17 Bytes lang.

Spectrum-Version



Die Besonderheiten des Spectrum sind immer wieder überraschend: Die Zeilenlänge belegt zwei Bytes, so daß eine einzige Programmzeile 65535 Zeichen lang sein könnte!

Der erste Block

In unserer Untersuchung des Acorn-Betriebssystems haben wir uns bisher mit den breitgefächerten Funktionen von OSBYTE beschäftigt. Wir untersuchen eine weitere Gruppe dieser Art – OSWORD.

Die bisher vorgestellten OSBYTE-Routinen haben einen wesentlichen Nachteil: Sie können nur zwei Parameter übergeben – einen im X-Register und einen in Y. Müssen nur wenige Parameter an das Betriebssystem übergeben werden, so reicht dies völlig aus, doch können die meisten Routinen nicht mit zwei Parametern gesteuert werden. Mit OSWORD lassen sich mehr als zwei Parameter einsetzen. Diese Modulgruppe ist eine praktische Alternative zu OSBYTE.

OSWORD-Routinen setzen einen Speicherbereich, den „Parameterblock" ein, um Parameter an das OS zu übergeben. Die Größe dieses Bereichs variiert je nach OSWORD-Routine und kann sich an jeder beliebigen Position des RAM befinden. Beim Aufruf von OSWORD teilt der Wert von Register A dem System mit, welche der Routinen angesprochen werden soll; die Register X und Y geben die Adresse des ersten Bytes des Parameterblocks an. X enthält dabei das niederwertige Adreßbyte und Y das höherwertige. Für einen Parameterblock bei der Adresse &0400 ist der Wert des X-Registers 00 und der von Y &0A:

Assembler	BASIC
LDX #0 LDY #&0A	X%=0 Y%=&0A

OSWORD wird über die Adresse &FFF1 aufgerufen und arbeitet wie OSBYTE mit Vektoren. Die Vektoren von OSWORD befinden sich bei Adresse &20C. Für den Aufruf der OSWORD-Routine mit Register A=1 müssen wir einen Parameterblock mit den entsprechenden Werten anlegen und die Routine dann mit folgendem Format aufrufen („parameter block" ist die Adresse des ersten Parameterblockbytes):

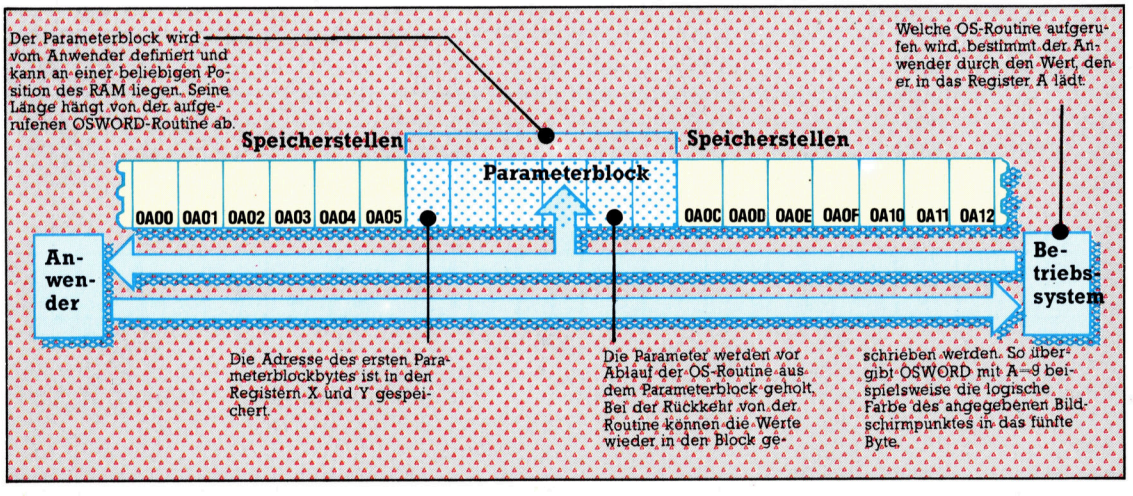
```
LDX  #parameter_block MOD 256
LDY  #parameter_block DIV 256
LDA  #1
JSR  &FFF1
```

Welche Werte in den Parameterblock geladen werden müssen, hängt natürlich von der angesprochenen OSWORD-Routine ab.

Die OSWORD-Routinen

Mit OSWORD-Routinen lassen sich Daten aus dem aktuellen Eingabekanal in den Speicher laden, die Befehle SOUND und ENVELOPE ausführen, das Diskettensystem ansprechen und viele weitere Funktionen steuern. Wir werden einige OSWORD-Calls genauer untersuchen. Die Routinen werden durch den Inhalt des A-Registers bezeichnet, das beim Aufruf die Routine eindeutig identifiziert (beispielsweise OSWORD mit A=0).

● **OSWORD mit A=0:** Diese Routine liest eine Zeile aus dem aktuellen Eingabekanal. Dabei kann ohne weiteres ganz einfach festgelegt werden, wieviele Zeichen das System maximal annehmen soll und welche niedrigsten und höchsten ASCII-Werte akzeptiert werden. Die ersten zwei Bytes des Parameterblocks geben die Speicheradresse an, bei der die Daten abgelegt werden. Der Parameterblock sieht dabei so aus:



OSWORD ist eine flexible Methode, OS-Routinen zu steuern, die mit mehr als zwei Parametern arbeiten.

Parameterblock von OSWORD mit A=0	
Byte	**Funktion**
0	Niederwertiges Byte der Bufferadresse
1	Höchstwertiges Byte der Bufferadresse
2	Maximale Anzahl der Zeichen
3	kleinstmöglicher ASCII-Code
4	größmöglicher ASCII-Code

Die Register X und Y zeigen auf Byte 0 des Blocks. Das folgende Programm gibt ein Beispiel des Aufrufs. Der Parameterblock legt dabei fest, daß immerhin bis zu sieben Zeichen mit ASCII-Codes zwischen 32 und 90 (inklusive) akzeptiert werden.

```
10 DIM C 20
20 A%=0
30 X%=C MOD 256
40 Y%=C DIV 256
50 C?0=&00
60 C?1=&0A: REM buffer at &0A00
70 C?2=7
80 C?3=32
90 C?4=90
100 CALL &FFF1
110 PRINT $(&0A00)
```

Starten Sie das Programm und geben Sie einige Zeichen ein. Delete funktioniert wie gewohnt, und Return oder Escape beenden die OSWORD-Routine. Zeichen mit ASCII-Codes außerhalb des angegebenen Bereiches werden zwar auf dem Bildschirm angezeigt, nicht aber in den Speicherbuffer übertragen. Werden zuviel Zeichen eingegeben, ertönt ein akustisches Signal. Bei Beendigung der Routine mit Escape setzt der Maschinencode das C-Flag, andernfalls steht C auf 0. Im Y-Register steht die Anzahl der eingegebenen Zeichen, wobei das Return mitgezählt wird.

● **OSWORDs mit A=1 und A=2:** Mit diesen Routinen kann der Maschinencodeprogrammierer auf die BASIC-Variable TIME zugreifen. OSWORD mit A=1 liest den aktuellen Wert von TIME. Die fünf Bytes dieses Wertes werden im Parameterblock gespeichert. Das Byte mit dem niedrigsten Wert ist in Byte 0 des Parameterblocks untergebracht und das höchstwertige Byte in Byte 5.

OSWORDs mit A=2 setzt TIME auf einen bestimmten Wert. Auch hier wird vor dem Aufruf das Byte mit dem niedrigsten Wert in Byte 0 des Parameterblocks geladen und das höchstwertige in Byte fünf. Das folgende Programm setzt den Wert von TIME auf 100:

```
10 DIM C 20
20 X%=C MOD 256
30 Y%=C DIV 256
40 A%=2
50 C?0=100
60 C?1=0
70 C?2=0
80 C?3=0
90 C?4=0
100 CALL &FFF1
```

● **OSWORDs mit A=3 und A=4:** Der Acorn B besitzt eine interne Uhr – den „Interval-Timer" –, die nicht von TIME eingesetzt wird, sondern von dem „Ereignissystem" (events system) des Computers (wir gehen später genauer darauf ein). Der Timer zählt von einem gesetzten Wert an in Hundertstelsekunden rückwärts und löst bei Null ein „Ereignis" aus. Das Ereignis läßt sich programmieren und kann nach der angegebenen Zeit im Computer Abläufe auslösen.

OSWORD mit A=3 liest den aktuellen Wert des Interval-Timers auf die gleiche Weise in den Parameterblock, in der OSWORD mit A=1 die Variable TIME speichert. Durch OSWORD mit A=4 kann der Timer auf einen bestimmten Wert gesetzt werden; hier gibt es Parallelen zu OSWORD mit A=2. Wenn ein Parameterblock den Wert &FFFFFFFC enthält, wird das Ereignis nach 0,004 Sekunden ausgelöst.

● **OSWORDs mit A=5 und A=6:** Diese beiden Spezialroutinen sprechen den zweiten Prozessor des Acorn B an. Der Zusatzprozessor kommuniziert über die Tube-Schnittstelle mit dem Acorn B und behandelt den Computer als E/A-Prozessor. Der Acorn B funktioniert nun als sogenannter „Sklave", der für den Zusatzprozessor alle zeitraubenden Arbeiten ausführt – beispielsweise die Abfrage von Tastatureingaben, die Steuerung des Bildschirmaufbaus und die allgemeinen Ein- und Ausgabefunktionen. Über die beiden OSWORD-Routinen kann der zweite Prozessor Speicherstellen im E/A-Prozessor ansprechen.

Parameterblock von OSWORD mit A=5 und A=6		
Byte	**OSWORD mit A=5**	**OSWORD mit A=6**
0	LSB der E/A-Prozessoradresse	LSB der E/A-Prozessoradresse
1	Adresse des zu lesenden Bytes	Adresse des zu schreibenden Bytes
2	Adresse des zu lesenden Bytes	Adresse des zu schreibenden Bytes
3	MSB des E/A-Prozessors	MSB des E/A-Prozessors
4	Nach dem Aufruf das Byte der oben angegebenen Adresse	Das Byte, das an die oben angegebene Adresse geschrieben wird

Die Tabelle zeigt den Aufbau des Parameterblocks für A=5 (ein Byte aus dem E/A-Speicher des Prozessors lesen) und für A=6 (in den E/A-Speicher des Prozessors schreiben). Wenn die Adresse das einfache 16-Bit-Format hat, wird ihr niederwertiges Byte bei (XY+0) abgelegt und das höherwertige bei (XY+1).

● **OSWORDs mit A=7 und A=8:** Über diese beiden Routinen läßt sich der Klang verändern (von Maschinencodeprogrammen und von BASIC aus). OSWORD mit A=7 simuliert den BASIC-Befehl SOUND, während OSWORD mit A=8 der ENVELOPE-Anweisung entspricht.

Hier die Aufrufe im einzelnen:

● **OSWORD mit A=7:** Die Routine benötigt einen Parameterblock von acht Bytes (die Tabelle zeigt die Belegung) und die Information, welcher Eintrag des Parameterblocks den Parametern des Befehls SOUND entspricht.

Parameterblock von OSWORD mit A=7	
Byte	**Funktion**
0	LSB der Kanalnummer
1	MSB der Kanalnummer
2	LSB der Amplitude
3	MSB der Amplitude
4	LSB der Tonhöhe
5	MSB der Tonhöhe
6	LSB der Dauer
7	MSB der Dauer

Das folgende Programm zeigt den Ablauf der Routine. Es simuliert SOUND 1,−15,100,30:

```
10  DIM C 20
20  X%=C MOD 256
30  Y%=C DIV 256
40  A%=7
50  C?0=1
60  C?1=0:REM Channel 1
70  C?2=-15 MOD 256
80  C?3=-15 DIV 256
90  C?4=100
100 C?5=0
110 C?6=30
115 C?7=0
120 CALL &FFF1
130 END
```

Da sich die gleiche Wirkung mit SOUND viel einfacher erzeugen läßt, hat diese Routine allerdings nur wenig Wert und wird daher auch wohl kaum Anwendung finden.

● **OSWORD mit A=8:** Der Parameterblock dieser Routine umfaßt vierzehn Bytes. Wie gewohnt zeigen dabei die Register X und Y auf den Block.

Der erste Blockeintrag gibt die Nummer von ENVELOPE an, die restlichen dreizehn Bytes enthalten die anderen ENVELOPE-Parameter. Auch hier läßt sich ENVELOPE vom BASIC aus leichter einsetzen.

Byte	Wert
0	2
1	4
2	0
3	0
4	0

Maschinencodeversion

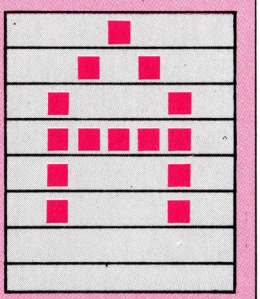
● **OSWORD mit A=9:** Die Routine ist die Maschinencodeversion des BASIC-Befehls POINT. Sie meldet die logische Farbe des angegebenen Pixels (oder den Wert 255, wenn der spezifizierte Punkt nicht aktiviert ist). Der Wert wird als fünftes Byte des Parameterblocks übergeben und hat folgenden Aufbau:

Parameterblock von OSWORD mit A=9	
Byte	**Funktion**
0	LSB der X-Koordinate
1	MSB der X-Koordinate
2	LSB der Y-Koordinate
3	MSB der Y-Koordinate
4	Logische Farbe

OSWORD mit A=10

Mit dieser praktischen Routine lassen sich vergrößerte Zeichen auf den Schirm bringen. Sie können damit die Zeichendefinitionen der Modi 0 bis 6 abrufen. Es wird ein Parameterblock von neun Bytes gebraucht, auf den wie gewöhnlich die X- und Y-Register zeigen. Das erste Byte des Blocks sollte den ASCII-Code des Zeichens enthalten.

Parameterblock von OSWORD mit A=10	
Byte	**Funktion**
0	gewünschtes Zeichen
1	erste Definitionszeile
2	zweite Zeile
3	dritte Zeile
4	vierte Zeile
5	fünfte Zeile
6	sechste Zeile
7	siebte Zeile
8	letzte Definitionszeile



Die in den Parameterblock zurückgegebenen Werte sind eine binäre Darstellung der Zeichendefinition.

● **OSWORD mit A=11:** Mit diesem Aufruf lassen sich die Farben untersuchen, die von VDU 19-Befehlen bestimmt wurden. Damit kann festgestellt werden, welche logische Farbe der dargestellten Farbe entspricht. Die Routine arbeitet mit einem Parameterblock von fünf Bytes, wobei das erste Byte den Wert der gewünschten logischen Farbe enthält. Hier ein Beispiel:

Nehmen wir an, daß der Befehl VDU 19,2,4,0,0,0 schon ohne weitere Vorbehalte ausgeführt wurde. OSWORD mit A=11 und dem Wert 2 im ersten Byte des Parameterblocks liefert dann entsprechend die Werte der nebenstehenden Tabelle.

● **OSWORD mit A=12:** Eine schnelle Version von VDU 19

● **OSWORD mit A=13:** Diese Routine liefert die X- und Y-Koordinaten der letzten beiden Punkte, die der Grafikcursor bei unserer Arbeit am Bildschirm angesprochen hat. Die Routine kann bei der Grafikprogrammierung sehr hilfreich sein.

Fachwörter von A bis Z

Light Pen = Lichtgriffel
Der Lichtgriffel erlaubt das Arbeiten am Bildschirm ohne Tastaturbenutzung. Sie setzen einfach die Griffelspitze auf den Schirm und können damit Rechnerfunktionen auslösen, Menüvarianten auswählen oder Linien ziehen.

Das praktische Gerät ist über ein Kabel mit dem Rechner verbunden. In die Spitze sind eine Linse und ein Fotodetektor eingebaut, der bei Lichteinfall ein elektrisches Signal abgibt. Es wird verstärkt, zu einem definierten Impuls geformt und an den Rechner weitergeleitet.

Seine Funktion beruht darauf, daß der Zeitpunkt registriert wird, zu dem das Lichtsignal beim zeilenweisen Aufbau des Bildes die Stiftspitze passiert. Der Video-Chip des Rechners „weiß", welche Zeile gerade geschrieben wird und wann sie begonnen worden ist. Aus der Zeit, die bis zum Eintreffen des Griffelsignals vergeht, kann der Computer anhand der bekannten Geschwindigkeit des Bildaufbaus ausrechnen, wo die Spitze auf den Schirm gesetzt wurde.

Der Lichtgriffel enthält selbst keine Lichtquelle, sondern er registriert den Vorbeigang des Kathodenstrahls auf dem Bildschirm; daraus erkennt der Rechner die Stelle, auf die mit der Stiftspitze gezeigt wurde. Der Lichtgriffel eignet sich vor allem für die interaktive Grafik und für die Menüsteuerung.

Line Printer = Zeilendrucker
Ein Zeilendrucker ist ein extrem schneller, mechanischer Drucker. Zeilendrucker wurden früher hauptsächlich in Verbindung mit Großrechnern eingesetzt; sie sind heute in der Druckgeschwindigkeit (etwa 30 Zeilen/s) von den elektrostatischen Laserdruckern überholt worden. Die englische Bezeichnung „Line Printer" wird auch für Typenrad- und Matrixdrucker verwendet.

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.

Unabhängig von der technischen Ausführung haben alle Drucker gemein, daß die Daten vom Rechner zunächst in einem RAM-Buffer abgelegt werden. Die Druckgeschwindigkeit ist in jedem Fall kleiner als die Übertragungsrate, so daß die CPU ohne Buffer für andere Funktionen blockiert wäre. So muß sie nur ab und zu einen Bufferinhalt abschicken und kann zwischendurch anderweitig tätig sein – der Drucker meldet sich, wenn er den Buffer abgearbeitet hat und fordert neue Daten an.

LISP = LISP
LISP ist eine Programmiersprache, die speziell für die Verarbeitung von Informationslisten (LISt Processing) ausgelegt ist. Die Listen werden mit Unterstützung des Systems definiert und mit Namen versehen. Mit den Listenelementen kann beliebig hantiert werden, etwa um festzustellen, ob bestimmte Bedingungen erfüllt sind oder nicht.

Die Programmierung in LISP besteht in der Festlegung der Verarbeitungsfunktionen; LISP ist eine „funktionale" Sprache, wobei ähnlich wie in LOGO komplexe Anweisungen aufgebaut werden können.

Local Area Network = Lokalnetz
Ein Lokalnetz oder (engl. abgek.) „LAN" besteht aus einer Anzahl von Rechnern, die untereinander und mit einer gemeinsamen Peripherie kommunizieren. Dabei ist weder ein einheitlicher Rechnertyp noch die enge räumliche Nachbarschaft der Computer zwingend.

Ein LAN kann auf dreierlei Weise realisiert werden: Beim Stern-System ist jeder beteiligte Computer mit einem Zentralrechner verbunden, der das ganze Netz und die Peripherie steuert. Der Datenverkehr zwischen zwei Satellit-Rechnern erfolgt dabei grundsätzlich über die Zentrale.

Beim Ringnetz sind die Rechner dagegen alle hintereinander geschaltet. Ringnetze sind für größere Teilnehmerzahlen ungeeignet, weil die Information dann auf dem Weg ans Ziel über zu viele Rechner laufen muß.

Beim Bus-System als drittem Typ, zum Beispiel beim Acorn-B-Econet, wird jeder Rechner über ein eigenes Anschlußkabel mit einer gemeinsamen Bus-Leitung verbunden, so daß unmittelbare Kontaktaufnahme zwischen Teilnehmern möglich ist, ohne daß eine Zwischenstation in Anspruch genommen werden muß.

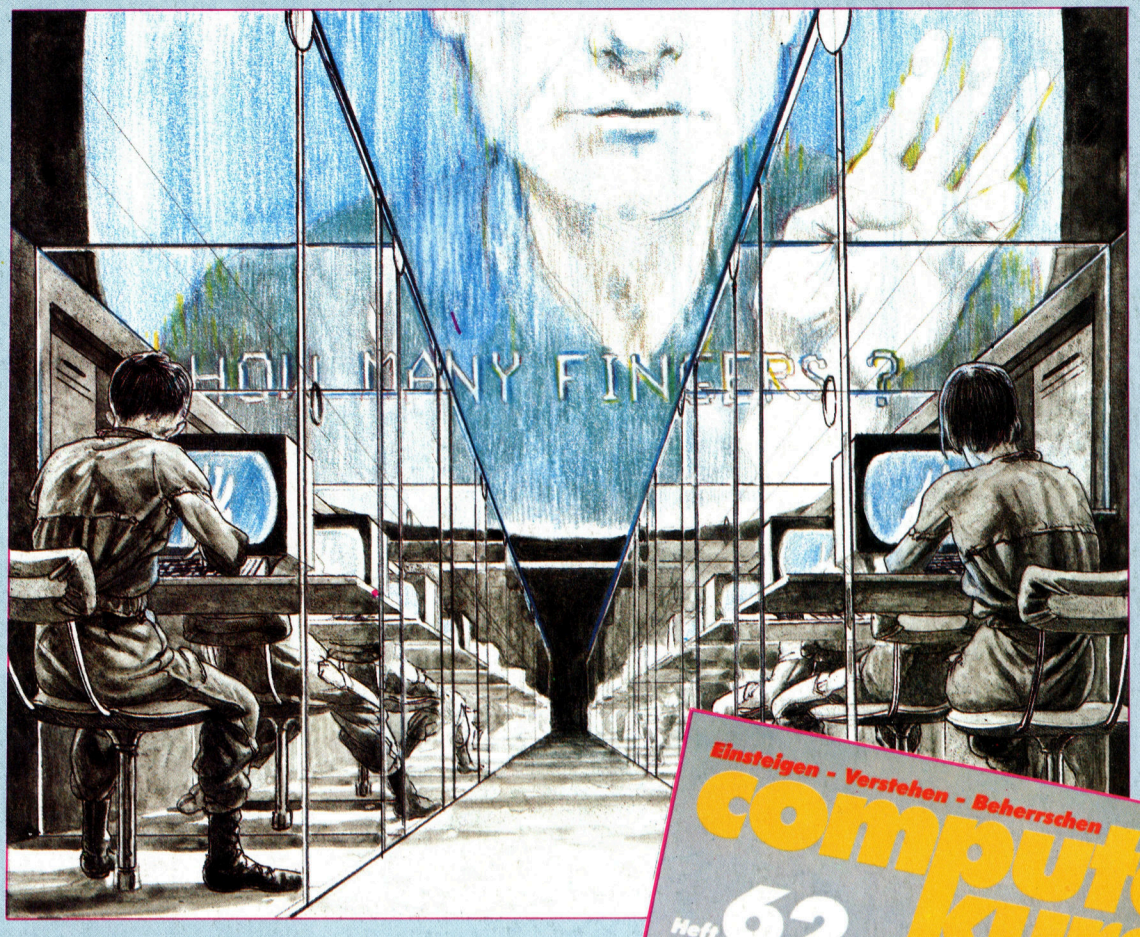
Logic = Logik
Die Logik ist das Fundament der gesamten Datenverarbeitung. Die Logik als Wissenschaft befaßt sich mit den Formalismen, die durch Verknüpfung einer Anzahl von Parametern – etwa mit Hilfe der Booleschen Algebra – zu bestimmten Schlußfolgerungen führen. Logische Voraussetzungen und Aussagen sind entweder wahr oder falsch – Zwischenwerte gibt es nicht. Deshalb eignet sich auch das Zweiersystem, mit dem jeder Digitalrechner arbeitet, ganz besonders für die Darstellung logischer Zusammenhänge.

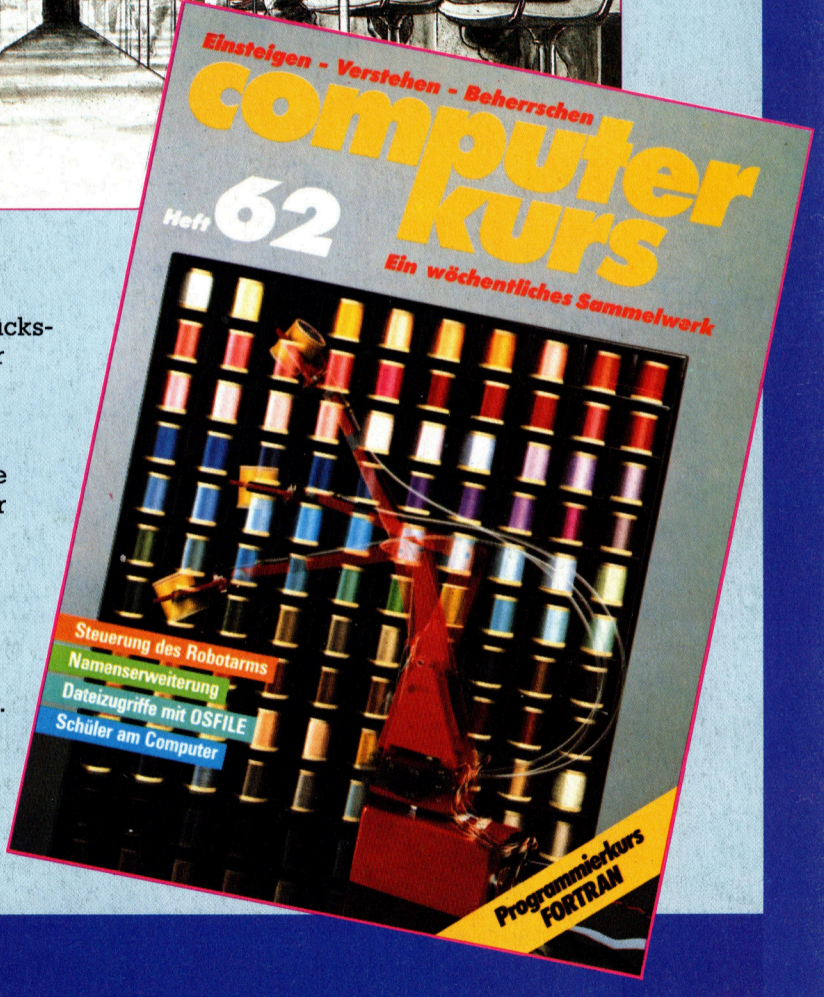
Als Logik wird oft auch die Elektronik für die Ausführung der zugehörigen Operationen bezeichnet. Im Rechner finden Sie zur Nachbildung der logischen Verknüpfungen AND-, NAND-, NOT-, OR-, NOR- und EXOR-Schaltkreise, die die Basis des ganzen digitalen Systems bilden.

Bildnachweise
1681, 1686, 1687, 1692: Ian McKinnell
1689, 1705, 1706: Liz Dixon
1691: Tony Sleep
1694, 1695: Kevin Jones
1696: Caroline Clayton
1701: Alan Adler

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

computer kurs Heft 62





Auf der Rennbahn
Der Microcomputer im Einsatz bei Glücksspielen. Zu Beginn vier Programme für Anhänger des Pferderennens.

Schatzsuche
Aussicht auf wertvolle Gewinne für die Lösung kniffliger Probleme. Ein Zepter aus Gold und Silber.

Dateizugriff
Wir untersuchen den Aufruf von OSFILE im Dateisystem des Acorn B. Daten werden an den Speicher geschickt.

Auf Armeslänge
Steuersoftware für unseren Robot-Arm. Für alle Micros ein Programm, so steuert jeder seinen Roboter.