Einsteigen - Verstehen - Beherrschen

DM 3,80 öS 30 sfr 3,80

computer kurs

Heft 31

Das Elternhaus von CP/M

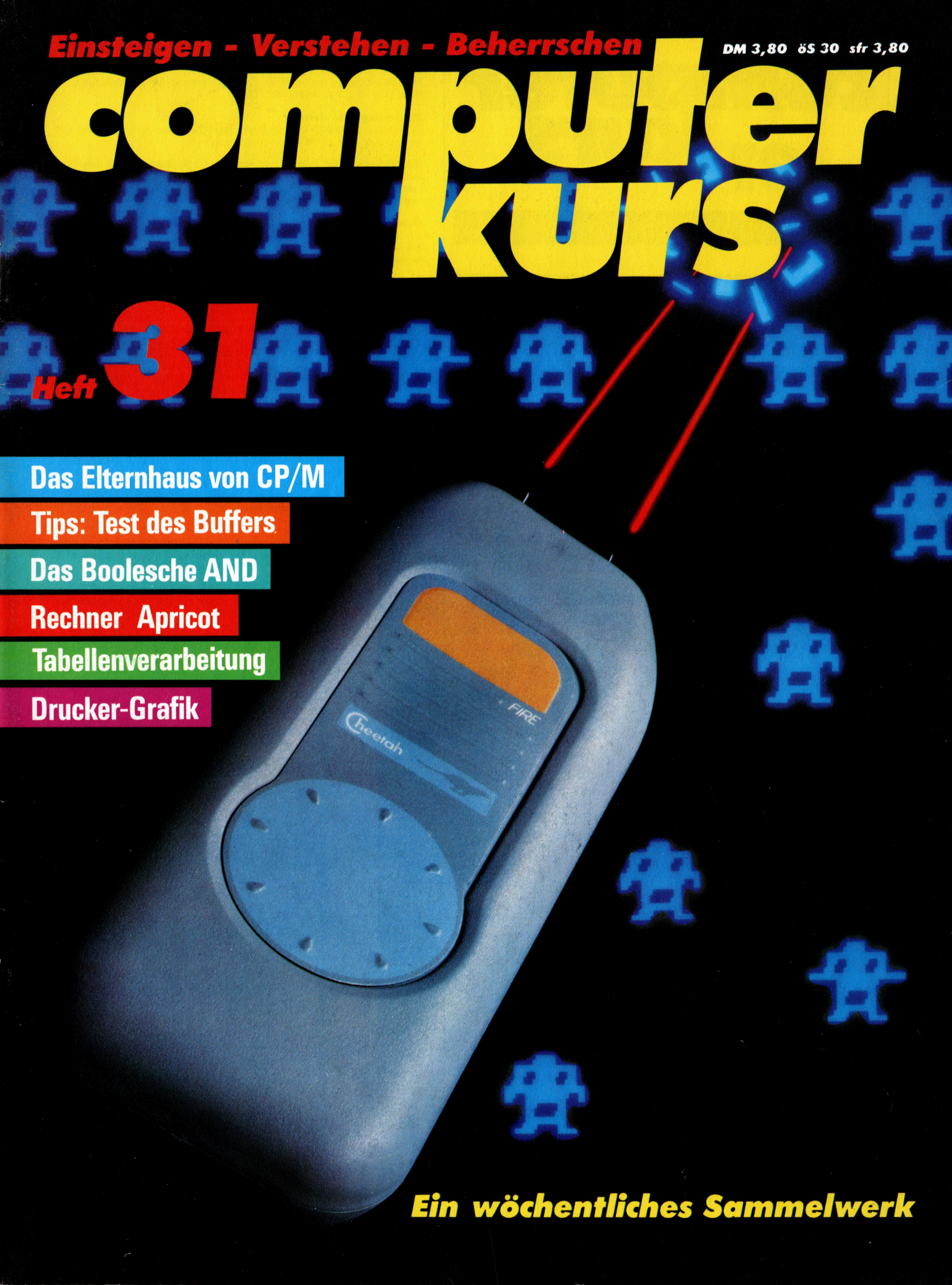
Tips: Test des Buffers

Das Boolesche AND

Rechner Apricot

Tabellenverarbeitung

Drucker-Grafik



Ein wöchentliches Sammelwerk

computer kurs

Heft 31

Inhalt

WIE SIE JEDE WOCHE IHR HEFT BEKOMMEN

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

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

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

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

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

Abonnement

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

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

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

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

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

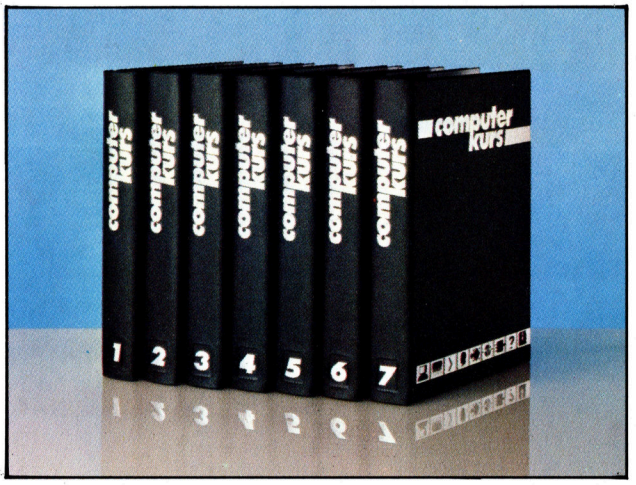
Schweiz: Der Sammelordner kostet sfr 15. Bitte wenden Sie sich an Ihren Kiosk; dort werden Sie jederzeit die gewünschten Exemplare erhalten.

INHALTSVERZEICHNIS

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

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

Vertrieb: Marshall Cavendish International Ltd., Heidenkampsweg 74, 2000 Hamburg 1, Tel.: 040/23 40 85



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

Große Roboterarme

Hier stehen zwei unterschiedliche Kategorien im Mittelpunkt: Roboter, die als Lernhilfen zur Demonstration der Technik Verwendung finden, und solche, die die derzeitigen Möglichkeiten der Robotik verwirklichen.

Die meisten der hier vorgestellten Roboter sind teuer. Einige davon kosten über viertausend Mark. Doch sie können nicht als Industrieroboter bezeichnet werden. Sie sind für die Verwendung daheim und in der Schule entwickelt worden.

Die erste zu betrachtende Gruppe sind Roboter, die einen hohen, anspruchsvollen Standard haben und teilweise durchaus den Roboterarmen vergleichbar sind, die in der Industrie eingesetzt werden. Der Hauptunterschied zwischen beiden Arten besteht darin, daß die hier vorgestellten für Ausbildungszwecke und für die Vermittlung von Grundlagen der Robotik entwickelt wurden. Sie sind kleiner und können nicht wie ihre in der Industrie verwendeten „Brüder" große Gegenstände bewegen.

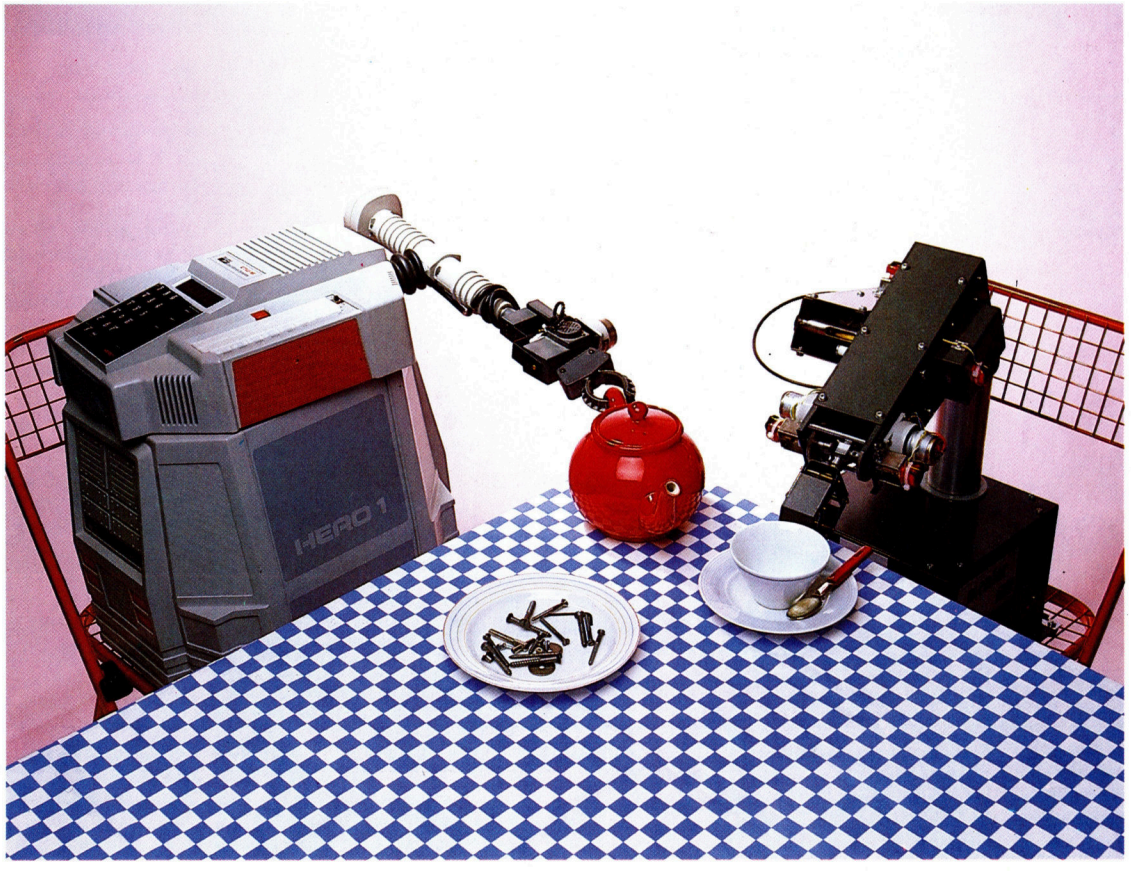
Natürlich überschneidet sich der Ausbildungsbereichsmarkt in einzelnen Bereichen mit dem Industriemarkt. – Dann etwa, wenn für industrielle Zwecke lediglich kleine, leichte Arme benötigt werden. So setzt man einerseits in der Industrie Arme ein, die mehrere hundert Kilo an Last (zum Beispiel Stahl) bewegen müssen, benötigt andererseits aber auch Arme, die lediglich kleine Teile auf elektronische Schaltungen zu setzen haben. Dafür sind natürlich keine großen Arme erforderlich.

Die zweite zu betrachtende Kategorie umfaßt Roboter, die den neuesten Stand der Entwicklung repräsentieren. Viele davon sind mit jenen Sensoren ausgestattet, die in anderen Teilen dieser Serie bereits behandelt wurden.

Arme zur Ausbildung

Ein vergleichsweise preiswerter Roboterarm ist der „Mentor" von „Cybernetic Applications". Er wird als Bausatz für rund 1400 Mark angeboten, verfügt über sechs Freiheitsgrade (Hüfte, Schulter, Ellenbogen und drei Drehwinkel am Handgelenk) und wird von Elektromotoren angetrieben.

Das Unternehmen bietet ebenfalls den „Neptune 1" und den „Neptune 2" an, die zwischen 5000 Mark und 8000 Mark (als Bausatz)



Da der Hero mobil ist und über einen Greifer verfügt, kann er seinem Besitzer den Tee zum Frühstück bringen (vorausgesetzt, daß das Schlafzimmer auf der selben Ebene wie die Küche liegt). Unser Foto zeigt Hero und Mentor beim gemeinsamen Frühstück.

Eigene Erfahrungen

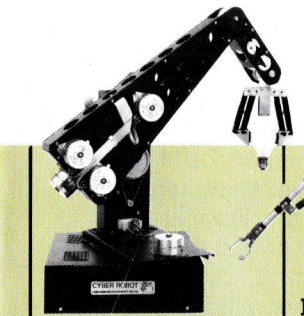
Die beiden wichtigsten Bereiche microelektronischer Foschung sind Chip-Konstruktion und Robotik. Nur wenige Hobbyisten können mit neuen Microchips daheim experimentieren. Roboter dagegen sind leicht zugänglich. Hauptzweck der hier vorgestellten Roboter ist, das Verständnis der Funktionsweise zu fördern. Am besten vertieft man sein Wissen um Roboter, indem man die Systeme studiert und sie zu verbessern versucht. Die hier gezeigten Arme, der „Micro Grasp" von Powertran Cybernetics und der „Armdroid" von Colne Robotics, sind über Computer programmierbar und haben fünf Freiheitsgrade. Sie können entweder fertig gebaut oder als Bausatz bezogen werden.

Name	Neptune 1 / Neptune 2	Mentor	Genesis P101	Hero
Typ	Arm	Arm	Arm	Bodenroboter
Hauptverwendung	Ausbildung	Ausbildung	Ausbildung	Ausbildung/Experiment
Sensoren	Potentiometer zeichnet Position auf, Greifer können Schließgrad feststellen	Potentiometer zeichnet Position auf, Greifer können Schließgrad feststellen	Rückmeldung der Position	Ultraschall zur Feststellung von Bewegung; Sensoren für 256 verschiedene Licht- und Schall-Größen; Tastsensoren am Greifer
Freiheitsgrade	6: Hüfte, Schulter, Ellenbogen und Handgelenkhebung, Handgelenkdrehung und Greifer	7: Hüfte, Schulter, Ellenbogen und Handgelenkhebung, Handgelenkdrehung, Handgelenkschwenken und Greifer	6: Hüfte, Schulter, Ellenbogen und Handgelenkhebung, Handgelenkdrehung und Greifer	6: Hüfte, Schulter, Ellenbogen, Handgelenkhebung, Handgelenkdrehung und Greifer	4: Schulter, Ellenbogen, Handgelenk und Greifer
Antrieb	Hydraulisch: Stromgetriebene Wasserpumpen	Hydraulisch: Stromgetriebene Wasserpumpen	Strom	Hydraulisch: Stromgetriebene Wasserpumpen	Wiederaufladbare Batterien
Zu verbinden mit:	Acorn B, Spectrum, VC 20	Acorn B, Spectrum, VC 20	Acorn B, Spectrum, VC 20	Programmierbar durch spezielle Box: RS232-Schnittstelle für Acorn B, Spectrum, VC 20	jedem Computer mit seriellem Port
Erhältlich bei:	Cybernetic Applications Ltd., West Portway Industrial Estate, Andover, Hampshire, SP10 3NN	Cybernetic Applications Ltd.	Cybernetic Applications Ltd.	Powertran Cybernetics, West Portway Industrial Estate, Andover, Hampshire, SP10 3NN	Zenith Data Systems, Bristol Road, Gloucester, GL2 6EE

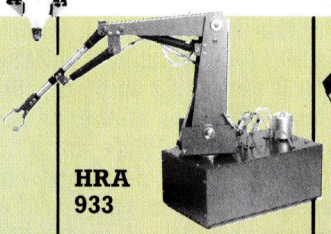
kosten. Diese Arme können Gewichte bis zu 2,5 kg heben und werden hydraulisch angetrieben. Dabei wird allerdings nicht die sonst übliche Hydraulikflüssigkeit verwendet, sondern Wasser. Auch diese Arme sind über den Acorn B, den VC 20 sowie den Sinclair Spectrum steuerbar. Neptune 2 verfügt über zwei unterschiedliche Arbeitsgeschwindigkeiten.

„Powertran Cybernetics" stellt den „Genesis P 101" her, der sechs Freiheitsgrade bietet und als Bausatz um 1400 Mark kostet. Das Modell wird hydraulisch betrieben, verfügt über eine Programmiereinheit und eine Standard RS232-Schnittstelle, die die Verbindung mit fast allen gängigen Computertypen erlaubt. Der Arm ist außerdem auch komplett – fertiggebaut und überprüft – erhältlich, kostet in dieser Form dann aber auch rund 7000 Mark.

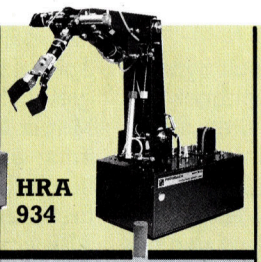
Ein sehr interessanter Arm in der mittleren Preisklasse ist der „Cyber 310" von „Cyber Robotics". Zusammengebaut ist er für etwa 2500 Mark zu haben. Versionen dieses Roboters stehen für Acorn B, Jupiter Ace, Apple II, Commodore PET 3000, 4000 und 8000 sowie den Hector HRX zur Verfügung. Er wird von Schrittmotoren angetrieben, hat aber nur eine Hebeleistung von 250 Gramm. Er ist somit ein Leichtgewicht, doch die Vielzahl der Optionen ist beeindruckend. Neben fünf Freiheitsgraden können Beschleunigung und Bremsung der Armgeschwindigkeit durch den Anwender gesteuert werden. Sämtliche Gelenke lassen sich gleichzeitig bewegen. Die Position des Arms ist entweder relativ (beispielsweise durch die Anweisung, daß der Arm sich um x Einheiten von seiner gegenwärtigen Position vorwärts zu bewegen habe) oder absolut (durch die Angabe einer Bewegung zu einem gewissen Punkt von einer zuvor bestimmten „Home"-Position aus) bestimmbar. Er kann sowohl in BASIC als auch in ROBO-FORTH programmiert werden. Diese Sprache ist eine Eigenentwicklung von Cyber Robotics.



Cyber 310



HRA 933



HRA 934

Arm	Arm	Arm
Ausbildung	Ausbildung	Ausbildung
Keine	Positionsfeststellung, Greifer können Schließgrad feststellen	Positions-Feststellung, Greifer können Schließgrad feststellen
6: Basis-Drehung, Schulter-Drehung (um 180 Grad), Ellenbogen, Handgelenkhebung, -drehung und Greifer	5: Basis-Drehung, Ellenbogen, Handgelenkdrehung, Handgelenkschwenken und Greifer	5: Basis-Drehung, Ellenbogen, Handgelenkdrehung, Handgelenkschwenken und Greifer
Strom	Strom	Strom
Acorn B, Jupiter Ace, Apple II, Commodore PET und Hector HRX	Apple II, Tandy TRS-80, Commodore PET, AIM 65, Acorn B, MAT 385	Apple II, Tandy TRS-80, Commodore PET, AIM 65, Acorn B, MAT 385
Cyber Robotics, 61 Ditton Walk, Cambridge, CB5 8QD	Feedback Systems Ltd., Park Road, Crowborough, East Sussex, TN6 2QR	Feedback Systems Ltd.

Für gehobene Ansprüche

Weitaus teurer sind die Modelle HRA 933 und HRA 934 von Feedback Instruments. Sie kosten zwischen 10 000 und 12 000 Mark und werden fertig gebaut geliefert. Bei beiden handelt es sich um hydraulisch betriebene Arme mit fünf Freiheitsgraden und einer Hebfähigkeit von 1,35 Kilo mit einer Positionierungsgenauigkeit von 3 mm. Neben Positionierungssensoren sind die Arme auch mit Tastsensoren an den „Händen" ausgestattet. Die Steuerung erfolgt über eine RS232-Schnittstelle.

Der „Hero-1" von „Zenith Data Systems" wird für etwa 10 000 Mark angeboten und hat ungewöhnliche Fähigkeiten. Der Bodenroboter ist mit einem Arm ausgestattet, der auf der Basis eines sphärischen Koordinatensystems arbeitet. Der Arm kann teleskopartig gestreckt und verkürzt werden. Ausgerüstet ist der „Hero-1" mit einer Vielzahl von Sensoren, die Bewegung, Licht und Geräusch wahrnehmen. Dazu gehört außerdem ein Ultraschall-Abstandssensor, der Zusammenstöße verhindert. Ein Speech-Synthesizer gibt dem Roboter einen umfangreichen Vokabelschatz. Sein Arm verfügt über fünf Freiheitsgrade.

Bei all diesen Robotern handelt es sich um vergleichsweise teure Unterhaltungsgeräte, betrachtet man einmal, was sie tatsächlich können. Bis heute besteht ihr Hauptzweck darin, für die Werbung eingesetzt zu werden, – etwa, um an Messeständen Prospekte zu verteilen oder Produkte vorzustellen. Dennoch repräsentieren sie den derzeitigen Stand der Roboter-Technologie. Sie alle verwenden Sensoren, können sprechen, akustische Signale wahrnehmen und darauf reagieren.

Natürlich wird der Preis viele Interessierte vom Kauf abhalten, doch begehrenswert sind sie allemal. Und vielleicht möchte der eine oder andere selbst einen Roboter dieser Art bauen. Zugleich aber verdeutlichen die vorgestellten Roboter, wie sich die Technik in den vergangenen Jahren verändert hat. Vor kurzem noch wären diese Roboterarme in dieser Form unvorstellbar gewesen.

Kalkulationen

Eines der ersten Programme, das die Möglichkeiten der Microcomputer voll zur Geltung brachte, war die Tabellenkalkulation. Da diese Systeme ursprünglich für kommerzielle Zwecke entwickelt wurden, sind sie im Bereich der Heimcomputer nie recht akzeptiert worden. Kalkulationsprogramme können aber nicht nur Informationen sammeln und darstellen, sondern lassen sich auch als „Ideengenerator" verwenden.

Wie Textsysteme und Datenbanken haben auch Kalkualtionssysteme Fähigkeiten, die viele Anwender nicht nutzen. Die meisten Besitzer von Heimcomputern finden, die Tabellenkalkulation sei langweilig und ohne praktischen Nutzen, da sie mit dem kommerziellen Einsatz nichts anfangen können. Abgeschreckt vom negativen Image einer Buchhaltungsmaschinerie unterschätzen sie dabei die Möglichkeiten finanzieller Planung. Sie übersehen, daß Kalkulationssysteme für Planungen die gleiche Bedeutung haben wie Textverarbeitungssysteme für den Entwurf von Schriftstücken.

Kalkulationssysteme sind eigentlich Synthesen aus Textsystem und Taschenrechner. Da ihre Datenfelder in Zeilen und Spalten angeordnet sind, werden sie auch als Tabellenkalkulationsprogramme bezeichnet. Ähnlich wie bei Tabellierpapier lassen sich die Felder auf mehrere Arten beschriften: Außer Text und numerischen Daten können sie mathematische Formeln enthalten, die die Inhalte verschiedener Zellen miteinander verknüpfen. Mit diesen Formeln verwandelt der Anwender das Kalkulationssystem in ein individuelles Programm, in das nur die entsprechenden Daten eingesetzt werden müssen. Bei der Eingabe von Daten berechnen automatisch alle angesprochenen Formeln ihren Zelleninhalt neu, so daß ständig ein aktueller Stand angezeigt wird.

Kalkulationssysteme können aber nicht nur Berechnungen anstellen, die sonst zu umständlich wären, sondern auch eine Reihe anderer Aufgaben erfüllen. Beispiele dafür sind der Druck von Inventarlisten, die Analyse von Sportergebnissen, Tabellenentwürfe, die Erstellung von Stichwortlisten für die Beleuchtungs- und Klangeffekte eines Theaters, die Berechnung des Lohnsteuerjahresausgleichs, die Untersuchung, ob die Miete oder der Kauf eines Fernsehers günstiger ist und vieles mehr. Sollten all diese Aufgaben in BASIC programmiert werden, dann würden Stunden vergehen, bis die vielen PRINT TABs, PRINT ATs und INPUTs fehlerfrei eingegeben sind. In einem Kalkulationssystem entsteht das Format jedoch fast automatisch, wenn die Beziehungen zwischen den einzelnen Variablen festgelegt werden, wobei die bildschirmgesteuerte Eingabe so unkompliziert ist wie der Eintrag von Daten auf Tabellierpapier.

Eine ganze Reihe von Spezialbefehlen vereinfacht den Aufbau der Bildschirmanzeige: Felder lassen sich einzeln oder blockweise kopieren, bewegen und löschen, ganze Zeilen oder Spalten können eingesetzt oder gelöscht werden, und das Format von Zellenblöcken (Zellengröße, Position des Dezimalpunktes) läßt sich einheitlich gestalten.

Auch die Kalkulationsfunktionen sind einfach zu bedienen. Ein einziger Befehl berechnet den Durchschnittswert einer Zeile oder Spalte, stellt fest, welche Tabelleneinträge ungleich Null sind, kalkuliert die Summe einer Matrix oder zeigt das Minimum und das Maximum einer Zahlenkolonne an. All diese Funktionen lassen sich mit den herkömmlichen mathematischen Zeichen wie "+", "/", SQR und ABS kombinieren. Die Zahl der Möglichkeiten hängt hauptsächlich von der Größe des Arbeitsspeichers und der Qualität des Programms ab, wobei einige Kalkulationssysteme nicht alle dieser Funktionen unterstützen.

Problemlose Handhabung

Einer der brauchbarsten Befehle einer Tabellenkalkulation ist REPLICATE. Er überträgt den Wert oder die Formel eines Feldes in andere Felder. Damit läßt sich eine Tabelle mit identischen Datenfeldern (zum Beispiel eine Liste der monatlichen Hypothekenzinsen) durch die Eingabe weniger Befehle zusammenstellen. Da sich komplizierte mathematische Ausdrücke mit einem Kalkulationssystem einfacher darstellen lassen als in BASIC, kann die Arbeit damit schon nach kurzer Zeit so vertraut werden wie die mit mathematischen BASIC-Befehlen.

Einmal angelegte Tabellen können auf Cassette oder Diskette gespeichert werden. Viele Programmversionen haben außerdem die Möglichkeit, Text und Daten in ein Dateiformat zu übertragen, das sich zur Weiterbearbeitung durch Textsysteme oder Datenbanken eignet. Die Ergebnisse von Berechnungen und Projektionen lassen sich so als Block in einen Text

In Reih und Glied

Formatieren
Mit dem Befehl FORMAT wurden Spaltenbreiten, Textstand und Zahlenanzeige festgelegt.

Kopieren
Jedes Datenfeld kann mit dem Befehl COPY in jeden anderen Bereich der Tabelle kopiert werden.

Multiplikationsfaktor
Um einen Vergleichswert zu erhalten, wird die Note mit diesem Faktor multipliziert.

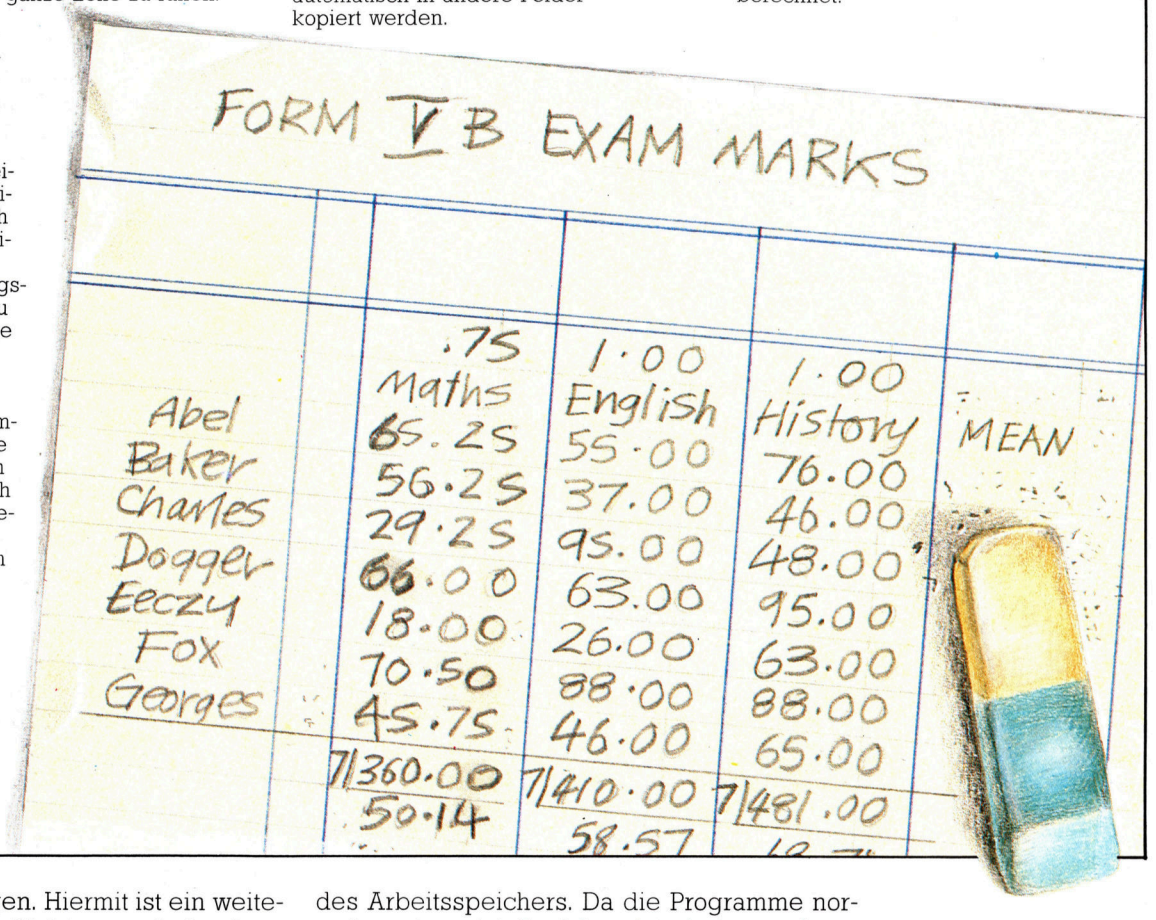
	C	D	E	F	G	H	I	J	K	L	M
1	MARKS ADJUSTMENT EXAMPLE										
2	**********	**********	**********	**********	**********	**********	**********	**********	**********	**********	**********
3			ACTUAL	MARKS		*		SCALED	MARKS		
4	**********	**********	**********	**********	**********	**********	**********	**********	**********	**********	**********
5						*	.75	.86	.73		
6		Maths	Eng	Hist	MEAN	*	Maths	Eng	Hist	MEAN	
7	Abel :	87.00	55.00	76.00	72.67	*	65.25	47.30	55.48	56.01	
8	Baker :	75.00	37.00	46.00	52.67	*	56.25	31.82	33.58	40.55	
9	Charles:	39.00	95.00	48.00	60.67	*	29.25	81.70	35.04	48.66	
10	Dogger :	88.00	63.00	95.00	82.00	*	66.00	54.18	69.35	63.18	
11	Eezy :	24.00	26.00	63.00	37.67	*	18.00	22.36	45.99	28.78	
12	Fox :	94.00	88.00	88.00	90.00	*	70.50	75.68	64.24	70.14	
13	George :	61.00	46.00	65.00	57.33	*	45.75	39.56	47.45	44.25	
14		==========	==========	==========	==========	*	==========	==========	==========	==========	
15	MEAN :	66.86	58.57	68.71	64.71	*	50.14	50.37	50.16	50.23	
16						*					
17	**********	**********	**********	**********	**********	**********	**********	**********	**********	**********	**********
18											

Textwiederholung
In Verbindung mit dem Befehl REPEAT TEXT genügt hier die Eingabe eines einzigen Sterns, um die ganze Zeile zu füllen.

Automatische Berechnung
Ist die Formel einmal in ein Feld eingegeben, kann sie mit dem Befehl REPLICATE automatisch in andere Felder kopiert werden.

Durchschnitt
Wird von dem Befehl AVERAGE (Feld# 1:Feld# 2) berechnet.

Um die Leistungen seiner Schüler in verschiedenen Fächern vergleichen zu können, suchte ein englischer Lehrer nach Maßstäben, die eine einheitliche Beurteilung zulassen. Er experimentierte für jedes Fach mit verschiedenen Multiplikationsfaktoren und mußte dabei die Leistungswerte immer wieder neu berechnen. Diese lästige und fehleranfällige Arbeit führt ein Kalkulationssystem in wenigen Minuten aus. In der Computertabelle werden alle Werte (außer den Noten der Schüler) automatisch berechnet. Bei der Änderung eines Multiplikationsfaktors entstehen in Sekunden die neuen Vergleichswerte dieses Faches.



oder eine Datei einfügen. Hiermit ist ein weiterer wichtiger Schritt in Richtung auf eine integrierte Datenverarbeitung getan. Diese Möglichkeit gibt es jedoch nur bei den teureren Programmpaketen.

Kalkulationssysteme finden ihre Grenzen in der Kreativität des Anwenders und der Größe des Arbeitsspeichers. Da die Programme normalerweise viel Speicherplatz belegen, kann der Einsatz großer Tabellen und komplexer Berechnungsvorgänge den vorhandenen Speicher schnell ausfüllen. Zu komplizierte mathematische Formeln verlangsamen die Arbeitsgeschwindigkeit der Programme.

Rasche Ratte

Die Firma Cheetah Marketing hat bereits einiges an Zubehör für den Sinclair Spectrum herausgebracht. Die neueste Errungenschaft für die Spiele-Fans ist die „Ratte", ein Steuerpult mit Infrarotübertragung, das wir Ihnen hier vorstellen.

Die professionellen Anhänger von Arcadespielen sind an allem interessiert, was zur Verbesserung der Spielqualität und Geschwindigkeit beitragen könnte – besonders, wenn es sich auf die Punktzahl auswirkt. Daher ist von großer Bedeutung, wie ein Spiel gesteuert wird. Bei Tastatursteuerung kommt es darauf an, daß die Kontrolltasten gut und schnell zu erreichen sind. Das ist ein wichtiger Aspekt bei der Softwareentwicklung. Bei anderweitiger Steuerung, etwa mittels Joystick, ist der kritische Faktor die Gestaltung der äußeren Form des Gerätes.

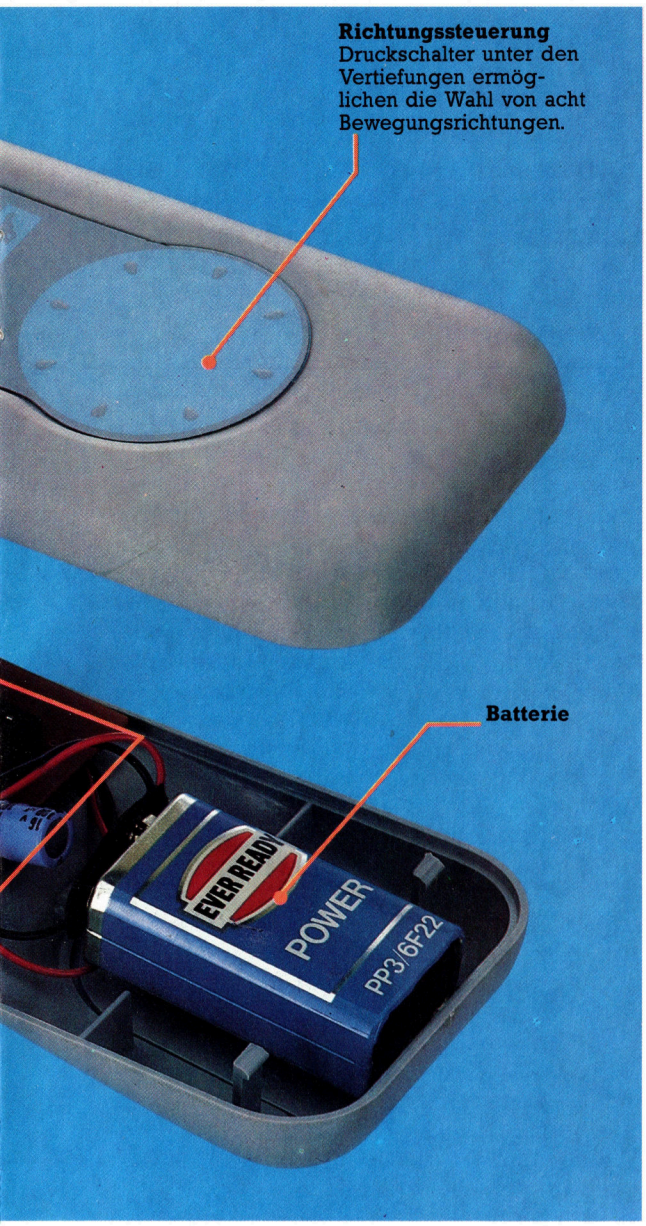
Bei einem Joystick ist Handlichkeit – das heißt Bewegungsfreiheit, schnelle Übertragung der Steuerungsdaten und prompte Spiel-Reaktion – oberstes Gebot für die Konstruktion. Der Designer muß bei der Entwicklung eines Joysticks einige Dinge beachten. Dazu gehören Kabellänge sowie Größe, Form und Stellung des Knüppels und die Anordnung der Feuerknöpfe. Viele Joystick-Hersteller haben versucht, diesen Unzuträglichkeiten zu Leibe zu rücken, aber keiner ist auf eine so konsequente Lösung gekommen wie die englische Firma Cheetah mit ihrem Infrarot-Joypad für den Sinclair Spectrum.

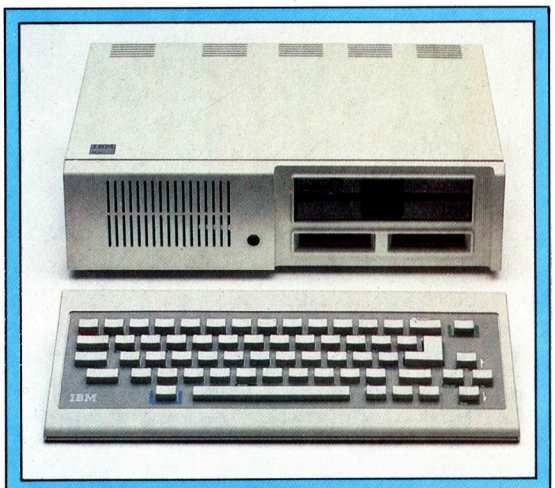
Konkurrenz zur Maus?

Cheetah hat diese Steuereinheit „RAT" (Ratte) genannt, angeblich als Abkürzung für „Remote Action Transmitter" (Fernsteuersender), aber dahinter steckt wohl eher eine Anspielung auf die „Maus", die beim Apple Macintosh und anderen Rechnern als handgeführtes Eingabegerät verwendet wird. Die „Ratte" ist flach und grau, mit einer runden blauen Steuerscheibe, dem Cheetah-Zeichen und einer orange leuchtenden Feuertaste. An der Vorderseite ragen zwei Infrarotdioden für die Datenübertragung aus dem Gehäuse hervor.

Zu dem Gerät gehört ein eigenes Interface,

Infrarotübertragung
Die Infrarotstrahlung (deren Wellenlänge größer als bei sichtbarem Licht, aber viel kleiner als bei Mikrowellen ist) wird im Sender durch Infrarot-Lumineszenzdioden erzeugt. Sie entsteht bei Stromdurchgang in einem winzigen Galliumarsenid-Kristall durch Rekombination von Ladungsträgern. Im Empfänger werden umgekehrt beim Auftreffen von Infrarot in einer Fotodiode Ladungsträger erzeugt, so daß Strom fließt. Bei Bedienung der „Ratte" emittieren die beiden Infrarotdioden einen Strahlenkegel von codierten Impulsen.





Zwiespältige Aufnahme
Die Original-Tastatur des IBM PC Junior arbeitete als erste im Microcomputerbereich mit Infrarotübertragung zum Rechner.

das in die Steckvorrichtung am Sinclair Spectrum eingesetzt wird. Dieser Kasten ist mit einem Erweiterungsstecker für weiteres Zubehör ausgestattet. An der Vorderseite befinden sich der Infrarotempfänger für den Kontakt mit der „Ratte".

Mitgeliefert wird ein Blatt mit einer Gebrauchsanweisung, worauf steht, für welche Spiele die Ratte verwendbar ist (zum Beispiel für Kempston-Joystick-kompatible Software). Zusätzlich werden Routinen in BASIC und Maschinencode aufgeführt, die es ermöglichen, die speziellen Steuercodes der „Ratte" in Ihre eigenen Spiele zu integrieren.

Laut Anleitungsblatt ist das Gerät bis zu etwa vier Meter Entfernung brauchbar, wenn man es in Richtung Rechner hält. Die Bewegungssteuerung erfolgt durch leichten Druck auf die blaue Steuerscheibe. Während Sie mit einer Hand die Ratte halten und die Bewegung auf dem Bildschirm steuern, können Sie mit der anderen die Feuertaste betätigen. Aufgrund der Form der Ratte spielt es keine Rolle, mit welcher Hand die Funktionen ausgeführt werden, so daß Rechts- und Linkshänder gleich gut bedient sind. Zur Stromversorgung dient eine kleine 9-V-Batterie, die in eine rückseitige Aussparung direkt unter der blauen Steuerscheibe eingesetzt wird.

Enorme Bewegungsfreiheit

Steckt das Interface im Rechner und ist die Batterie im Sender eingebaut, brauchen Sie nur noch ein Spiel für Joystick-Steuerung in Ihren Spectrum zu laden.

Im Test zeigte sich, daß das System auch bei mehr als vier Metern Senderentfernung noch zuverlässig arbeitet. Dabei brauchen Sie nicht den Rechner anzupeilen, sondern können auf Decke oder Fußboden zielen, auch nach hinten oder seitwärts. Zweifellos verschafft die Ratte jedem Spieler eine enorme Bewegungsfreiheit – der Nachteil besteht allerdings darin, daß Sie nur acht Richtungen ansteuern können, nämlich aufwärts, abwärts, links, rechts und die jeweils dazwischenliegenden Punkte. Eine differenziertere Steuerungsmöglichkeit wäre wünschenswert. Zudem erkennt man das Funktionieren des Systems leider erst dann, wenn sich auf dem Bildschirm etwas bewegt – eine Betriebsanzeige (etwa eine LED) wurde dem Gerät nicht eingebaut.

Da die „Ratte" keine beweglichen Teile enthält, ist das Gerät weniger verschleißanfällig als die üblichen Joysticks. Die Ratte dürfte also „langlebiger" sein, und der Hersteller räumt ein Jahr Garantie ein. Mit einem Preis von etwa hundert Mark kostet der Spaß etwas mehr als die meisten anderen Joysticks, die mit dem „Interface 2" arbeiten. Dafür bietet dieses System viel mehr Bewegungsfreiheit und eine weit bessere Spielkontrolle als die meisten herkömmlichen Joysticks.

„Remote Action Transmitter"
Spielsteuerpult von Cheetah
Erforderlicher Rechner: Sinclair Spectrum
Einsatzbereich: Alle Spiele, die Cheetah-kompatibel oder Kempston-Joystick-kompatibel sind
Vertrieb: Cheetah Marketing Ltd., 24 Roy St, London EC1

U-Boot-Jagd

Die folgende kleine Serie soll Ihnen helfen, die Grafikmöglichkeiten des Commodore 64 voll auszunutzen. Wir werden viele Aspekte betrachten, einschließlich Bildschirmdarstellung, Erstellung von Sprites sowie Tastatursteuerung.

In dem U-Boot-Jagdspiel, das wir im Verlauf der folgenden Artikel für den Commodore 64 erstellen wollen, werden vier der insgesamt acht verfügbaren Sprites verwendet. Diese Sprites repräsentieren ein Schiff, ein U-Boot, Wasserbomben und Explosionen.

Jeder Abschnitt des Programms wird als kurze Unterroutine geschrieben, die bei Bedarf von der Hauptprogramm-Schleife aufgerufen wird. Diese Art der Strukturierung erleichtert die Fehlersuche und eine eventuelle Programmerweiterung zu einem späteren Zeitpunkt erheblich.

Das Konzept dieses Spiels wird vielen bereits geläufig sein. Der Spieler steuert ein Schiff, das U-Boote jagt. Diese kreuzen in verschiedenen Tiefen und mit unterschiedlichen Geschwindigkeiten über den Bildschirm, während das Schiff Wasserbomben auf sie wirft. Die Punktzahl des Spielers wird immer dann erhöht, wenn ein U-Boot getroffen wurde. Dabei wird der Wert eines U-Bootes abhängig von seiner Tiefe und Geschwindigkeit berechnet. So erhält man für sehr tief und schnell fahrende U-Boote höhere Punktzahlen. Trifft man ein Unterseeboot nicht, so wird sein Wert von der Gesamtpunktzahl des Spielers abgezogen. Das Schiff wird über die Tastatur mit Hilfe der Tasten Z und X nach links und nach rechts gesteuert. Wasserbomben werden durch Drükken der Taste M geworfen. Auf dem Bildschirm wird ferner eine Stoppuhr dargestellt, die eine Spieldauer von drei Minuten vorgibt. Ist diese Zeit abgelaufen, wird der Spieler gefragt, ob er ein weiteres Spiel wagen will, und ein Verzeichnis der höchsten Punktzahlen wird auf dem Schirm dargestellt.

Die goldene Regel, die man bei der Entwicklung eines Action-Spieles beachten sollte, ist, die Hauptprogrammschleife so kurz wie irgend möglich zu halten. Das U-Boot-Programm verwendet für fast alle Funktionen innerhalb des Spieles Unterroutinen. Die einzigen Funktionen, die direkt in der Hauptschleife kontrolliert werden, sind folgende: Aktualisierung der Stoppuhr, Abfrage nach Eingaben über die Tastatur, Bewegung des Schiffes und Bewegung der U-Boote.

Es gibt zwei Wege, wie man beim Commodore 64 Zeichen auf den Bildschirm bringen kann. Der eine ist, die PRINT-Anweisung zu benutzen. Der andere Weg ist, Zahlen direkt in die Speicherstellen zu POKEn, die die Informationen über den Bildschirm enthalten. Wir werden bei der Erstellung des Hintergrundes beide Möglichkeiten verwenden.

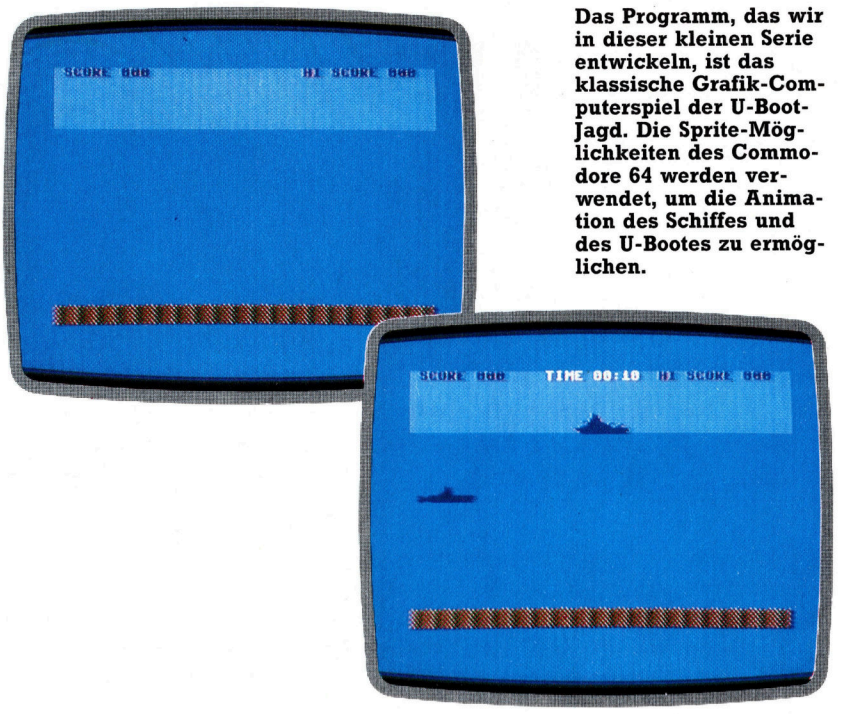
Der Bildschirm des Commodore ist aus 25 Reihen mit je 40 Zeichen aufgebaut. Mit anderen Worten gibt es 1000 Positionen auf dem Bildschirm, an denen ein Zeichen dargestellt werden kann. Zu jeder Position auf dem Bildschirm gehören zwei Zahlen. Die erste Zahl ist eine Bildschirm-Code-Nummer, die dem Computer angibt, welches Zeichen an dieser Position dargestellt werden soll. Die zweite Zahl ist eine Farbkennziffer, die bestimmt, welche Farbe das dargestellte Zeichen haben soll. Es existieren also zwei Speicherblöcke mit jeweils 1000 Adressen: Der eine Block enthält die Informationen über den Bildschirm-Code und der andere die über die Farbe jeder einzelnen Position auf dem Bildschirm. Der Block, der den Bildschirm-Code enthält, wird Bildschirmspeicher genannt und liegt im Bereich von Adresse 1024 bis 2023. Der Farbspeicher liegt zwischen Adresse 55296 und 56295.

Jedes Zeichen hat seinen eigenen Bildschirm-Code, welche auf Seite 132 des Anwenderhandbuches aufgelistet sind. Beim Commodore 64 gibt es 16 Farben. Die Farbcodes sind:

0	Schwarz	8	Orange
1	Weiß	9	Braun
2	Rot	10	Hellrot
3	Blau 1	11	Grau 1
4	Violett	12	Grau 2
5	Grün	13	Hellgrün
6	Blau 2	14	Hellblau
7	Gelb	15	Grau 3

Zusätzlich zu diesen beiden Speicherbereichen sind zwei weitere Adressen besonders interessant. Speicherstelle 53280 kontrolliert die Randfarbe, und Adresse 53281 setzt die Bildschirmfarbe. Um die Meereslandschaft für unser Spiel zu entwerfen färben wir die oberen sechs Reihen des Bildschirms für den Himmel hellblau, wogegen der Rest des Bildschirms und der Rand eine dunkelblaue Farbe als Darstellung der See erhalten soll (mit Ausnahme der unteren zwei Reihen, die den Meeresgrund reprasentieren).

Um die Bildschirmfarbe hellblau und die



Das Programm, das wir in dieser kleinen Serie entwickeln, ist das klassische Grafik-Computerspiel der U-Boot-Jagd. Die Sprite-Möglichkeiten des Commodore 64 werden verwendet, um die Animation des Schiffes und des U-Bootes zu ermöglichen.

Randfarbe dunkelblau zu färben, werden die folgenden beiden POKE-Befehle benötigt:

```
POKE 53281,14:POKE 53280,6
```

Die siebte Reihe des Bildschirmes beginnt bei Speicherstelle 1264. Die zweite Reihe vom unteren Rand gesehen beginnt bei Adresse 1944. Das Meer wird gefärbt, indem wir den Bildschirm-Code für eine invertierte Leerstelle in die Adressen 1264 bis 1943 POKEn. Außerdem muß der Wert 6 in die zugehörigen Farbadressen gePOKEt werden.

Der Bildschirm-Code für eine Leerstelle ist 32. Der Code eines inversen Zeichens kann einfach errechnet werden, indem man den Wert 128 zum normalen Code addiert. Der Code für ein invertiertes Leerzeichen errechnet sich somit aus 32 + 128 = 160. Die folgenden Programmzeilen verwenden eine einfache FOR...NEXT-Schleife, um das Meer wunschgemäß einzufärben:

```
FOR I=1264 TO 1943
POKE I,160:POKE I+54272,6
NEXT I
```

Der Meeresgrund besteht aus zwei Reihen karierter Zeichen mit dem Bildschirm-Code 102 und der Farbe Braun.

```
FOR I=1944 TO 2023
POKE I,102:POKE I+54272,9
NEXT I
```

Die PRINT-Anweisung ist eine weitere Möglichkeit, Bildschirmdarstellungen zu erzeugen. Farb- und Cursor-Positionierung können mit einer PRINT-Anweisung gesteuert werden, entweder durch Verwendung der speziellen Commodore-Kontroll-Zeichen oder durch Verwendung von CHR$-Codes. Wir werden die letztere Methode verwenden, da sie später im Listing des Programms leichter verständlich ist. Nachstehend sind die Codes ausgeführt, die die Farben und Cursor-Positionen beeinflussen:

CHR$(5)	Farbe Weiß
CHR$(144)	Farbe Schwarz
CHR$(147)	Löscht Bildschirm und positioniert Cursor in der oberen linken Ecke
CHR$(19)	Positioniert Cursor in der oberen linken Ecke
CHR$(17)	Verschiebt Cursor eine Stelle nach UNTEN
CHR$(145)	Verschiebt Cursor eine Stelle nach OBEN
CHR$(157)	Verschiebt Cursor eine Stelle nach LINKS
CHR$(29)	Verschiebt Cursor eine Stelle nach RECHTS

Als Teil unserer Bildschirmaufbau-Routine sollen die Worte PUNKTE und BESTWERT in der obersten Zeile des Bildschirmes dargestellt werden. CHR$(19) gewährleistet, daß sich der Cursor am Anfang der ersten Zeile befindet. Der folgende Befehl PRINTed den Anfangspunktstand in Schwarz.

```
PRINT CHR$(19);CHR$(144),"PUNKTE 000"
```

Der BESTWERT muß ebenfalls in der ersten Zeile positioniert werden, jedoch auf der rechten Seite. Die SPC-Funktion ermöglicht es, die entsprechende Anzahl an Leerzeichen einzufügen. Der PRINT-Befehl kann nun erweitert werden, um auch das Wort BESTWERT in der ersten Zeile auszugeben:

```
PRINT CHR$(19);CHR$(144);"PUNKTE
000";SPC(16);"BESTWERT 000"
```

Die Bildschirmaufbau-Routine ist ein Unterprogramm mit Beginn der Zeilennummer 1000. Die eben besprochene Unterroutine kann mit den folgenden Programmzeilen getestet werden:

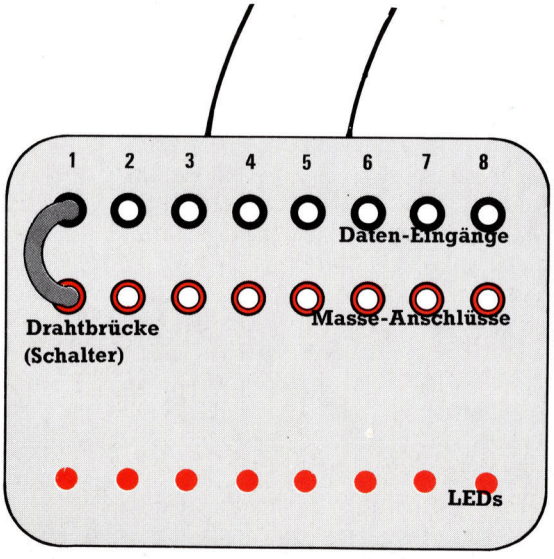
```
  10 GOSUB 1000:REM BILDSCHIRM-
     AUFBAU
  20 END
1000 REM *** BILDSCHIRMAUFBAU ***
1010 PRINT CHR$(147) REM LOESCHT DEN
     BILDSCHIRM
1020 :
1030 REM ** FAERBEN DES MEERES **
1040 POKE 53281,14:POKE 53280,6
1050 FOR I=1264 TO 1943
1060 POKE I,160 POKE I+54272,6
1070 NEXT
1080 :
1090 REM ** MEERESGRUND **
1100 FOR I=1944 TO 2023
1110 POKE I,102:POKE I+54272,9
1120 NEXT
1130 POKE 650,128:REM WIEDERHOLT
     TASTEN
1140
1150 REM ** PUNKTE **
1160 PRINT CHR$(19),CHR$(144);"PUNKTE
     000";SPC(16);"BESTWERT 000"
1170 RETURN
1180 :
1190 :
```

Wenn Sie die Routine eingegeben haben, ist es ratsam, diese auf Cassette oder Diskette abzuspeichern, bevor Sie sie starten.

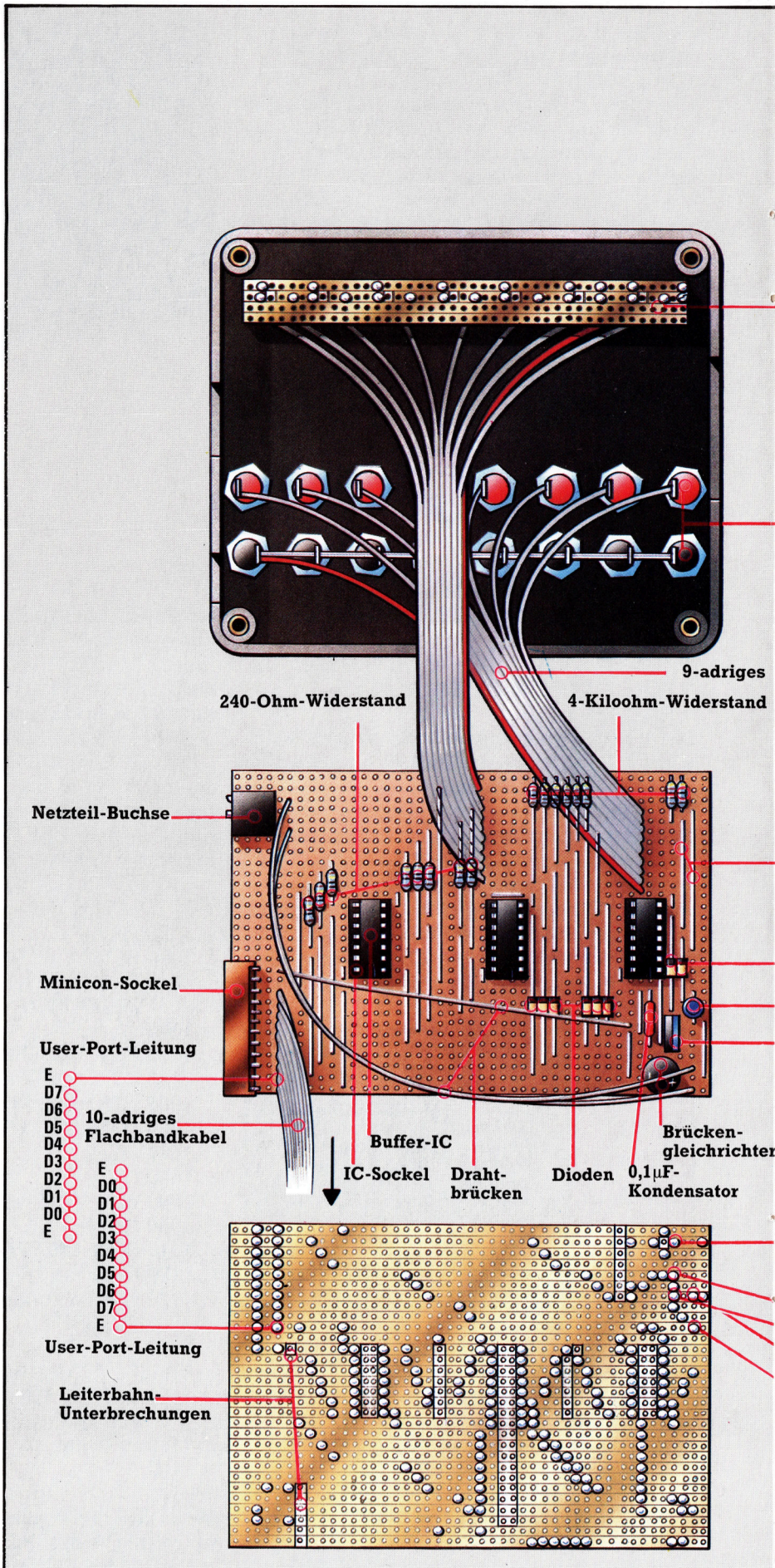
Test des Buffers

Den User Port des C 64 und des Acorn B kennen Sie nun schon recht genau. Mit unserem selbstgebauten Buffer läßt sich die Datenein- und -ausgabe steuern. Jetzt ist es an der Zeit, diesen Buffer zu testen.

Nach der Fertigstellung des Eingangs-Buffers können Sie das Gerät durch ein unkompliziertes Programm als Reaktionszeit-Messer testen. Dabei werden Bit 7 des Datenregisters für den Eingangsschalter, Bit 0 bis 6 für die Ansteuerung der LEDs verwendet. Ziel ist es, den Zeitabstand vom Aufleuchten der LEDs bis zum Schließen des Kontakts zwischen Anschluß 7 des Buffers und der Masse zu bestimmen. Sie benötigen nicht einmal einen richtigen Taster – die Verbindung läßt sich auch durch ein Stück Draht herstellen. Bei geschlossenem „Schalter" (Draht oder Kabel zwischen Bit 7 und Masse) sollte der Wert von Bit 7 „Eins" sein. Die Abbildung zeigt die einzelnen Verbindungen:

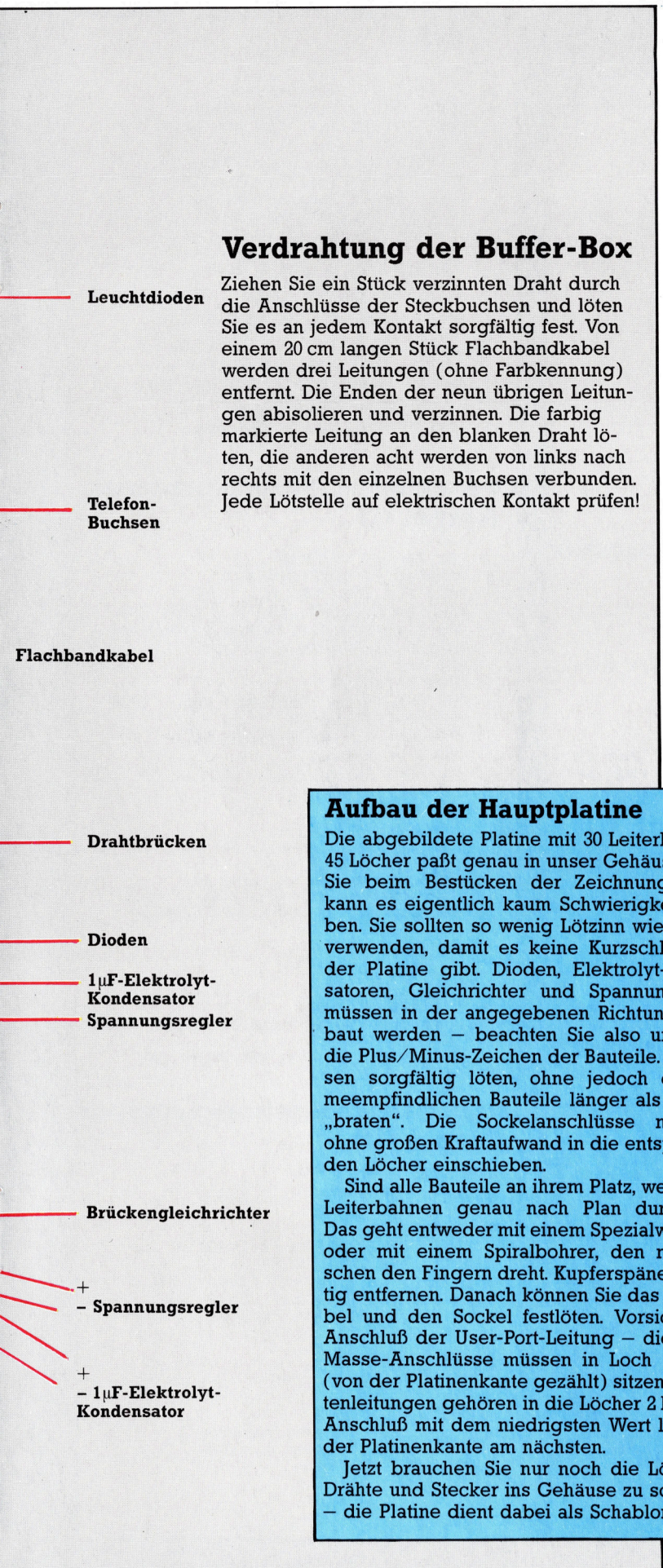


Das Testprogramm stellt das Daten-Richtungs-Register für Bit 7 auf Eingabe, alle anderen Bits auf Ausgabe und setzt den Inhalt des Datenregisters auf Null. Nach einer kurzen Wartezeit werden die LEDs eingeschaltet und die interne „Uhr" des Rechners gestartet. Die Zeitabnahme läuft, bis Bit 7 durch Schließen des Kontaktes „LOW" wird. Nicht vergessen: Die LEDs sind so angeschlossen, daß sie bei einer Null im Datenregister leuchten!



Verdrahtung der Buffer-Box

Ziehen Sie ein Stück verzinnten Draht durch die Anschlüsse der Steckbuchsen und löten Sie es an jedem Kontakt sorgfältig fest. Von einem 20 cm langen Stück Flachbandkabel werden drei Leitungen (ohne Farbkennung) entfernt. Die Enden der neun übrigen Leitungen abisolieren und verzinnen. Die farbig markierte Leitung an den blanken Draht löten, die anderen acht werden von links nach rechts mit den einzelnen Buchsen verbunden. Jede Lötstelle auf elektrischen Kontakt prüfen!



Aufbau der Hauptplatine

Die abgebildete Platine mit 30 Leiterbahnen à 45 Löcher paßt genau in unser Gehäuse. Wenn Sie beim Bestücken der Zeichnung folgen, kann es eigentlich kaum Schwierigkeiten geben. Sie sollten so wenig Lötzinn wie möglich verwenden, damit es keine Kurzschlüsse auf der Platine gibt. Dioden, Elektrolyt-Kondensatoren, Gleichrichter und Spannungsregler müssen in der angegebenen Richtung eingebaut werden – beachten Sie also unbedingt die Plus/Minus-Zeichen der Bauteile. Sie müssen sorgfältig löten, ohne jedoch die wärmeempfindlichen Bauteile länger als nötig zu „braten". Die Sockelanschlüsse möglichst ohne großen Kraftaufwand in die entsprechenden Löcher einschieben.

Sind alle Bauteile an ihrem Platz, werden die Leiterbahnen genau nach Plan durchtrennt. Das geht entweder mit einem Spezialwerkzeug oder mit einem Spiralbohrer, den man zwischen den Fingern dreht. Kupferspäne sorgfältig entfernen. Danach können Sie das Flachkabel und den Sockel festlöten. Vorsicht beim Anschluß der User-Port-Leitung – die beiden Masse-Anschlüsse müssen in Loch 1 und 10 (von der Platinenkante gezählt) sitzen. Die Datenleitungen gehören in die Löcher 2 bis 9, der Anschluß mit dem niedrigsten Wert liegt also der Platinenkante am nächsten.

Jetzt brauchen Sie nur noch die Löcher für Drähte und Stecker ins Gehäuse zu schneiden – die Platine dient dabei als Schablone.

```
10 REM * ACORN REAKTIONSZEIT **
15 :
20 DDR=&FE62: DATREG=&FE60
30 ?DDR=127: REM ZEILEN 0—6 OUTPUT
40 ?DATREG=127: REM LEDS AUS
50 :
60 CLS: PRINT "FERTIG"
70 DELAY=3000+RND(9000)
80 FOR I=1 TO DELAY:NEXT:REM DELAY SCHLEIFE
85 FOR D=1 TO 200:NEXT D
90 :
97 REPEAT UNTIL ?DATREG AND 128=1
100 ?FE60=0: REM LEDS AN
110 TIME =0: REM UHR INIT
120 REPEAT
130 UNTIL ?DATREG AND 128=0: REM S/W ON
140 :
150 PRINT "ZEIT BETRUG= "TIME/100"SEK."
160 END
```

Das Programm verwendet den Befehl TIME, der zur Ausgabe der verstrichenen Hunderstelsekunden seit dem letzten TIME = 0 führt. Mit dem logischen AND in Zeile 130 kann Bit 7 (Wert 128) unabhängig von den anderen Bits im Datenregister geprüft werden. Beim Schließen des Kontakts fällt sein Wert von Eins auf Null ab.

Das entsprechende Programm für den Commodore 64 unterscheidet sich durch den Befehl TI. Anders als TIME gibt TI die Länge der seit dem Einschalten des Rechners verstrichenen Zeit in Sechzigstelsekunden aus. Also muß von der Zeit TI am Ende des Intervalls die Zeit TI von dessen Anfang abgezogen werden, um die tatsächliche Zeitspanne zwischen dem Aufleuchten der LEDs und dem Schließen des Kontakts zu bestimmen.

```
10 REM ** C 64 REAKTIONSZEIT **
20 :
30 DDR=56579: DATREG=56577
40 POKE DDR,127: REM ZEILEN 0—6 OUTPUT
50 POKE DATREG,127: REM LEDS AUS
60 :
70 PRINT CHR$(147): REM SCHIRM LOESCHEN
80 PRINT "FERTIG"
90 DE=3000+INT(9000*RND(1))
100 FOR N=1 TO DE:NEXT: REM DELAY SCHLEIFE
110 :
120 POKE DATREG,0: REM LEDS AN
130 T=TI: REM START ZEIT
140 IF PEEK(DATREG) AND 128<>0 THEN 140
150 :
160 TM=(TI—T)/60: REM RECHENINTERVALL
170 PRINT "ZEIT BETRUG= ";TM;"SEK."
180 END
```

Im nächsten Kursabschnitt wollen wir höhere Ausgangsspannungen erzeugen, mit denen Motoren angetrieben werden können. Außerdem sollen Programme für die Steuerung von Geschwindigkeit und Laufrichtung entstehen.

Kleine Experimente

1) Schreiben Sie ein Programm, das die LEDs nacheinander aufleuchten läßt.

2) Ändern Sie das Programm, so daß LED 0 bis 6 nacheinander aufleuchten, die Laufrichtung sich aber mit einem Schalter in Leitung 7 umkehren läßt. Können Sie auch ein Lauflicht aus je drei LEDs programmieren?

3) Simulieren Sie mit sechs LEDs und einem Schalter einen Würfel.

4) Imitieren Sie mit drei LEDs die Funktion einer Verkehrsampel.

5) Schreiben Sie ein Programm, das die während einer Rotphase ankommenden „Autos" (Schaltimpulse) zählt. Bei mehr als 10 Autos oder einer Wartezeit von mehr als 1 Minute soll die Ampel umschalten.

Lösungsvorschläge

für die Experimente der letzten Seite

LED-Lauflicht (I)

```
 10 REM CBM 64 VERSION 2.1
 20 DDR=56579: DATREG=56577
 30 POKE DDR,255: REM ALL OUTPUT
 40 POKE DATREG,255: REM ALL LEDS OFF
 50 GET A$
 60 FOR N=1 TO 7
 70 POKE DATREG,255-(2^N)
 80 NEXT N
 90 IF A$="" THEN 50
100 END
```

```
 10 REM BBC VERSION 2.1
 20 DDR=&FE62:DATREG=&FE60
 30 ?DDR=255: REM ALL OUTPUT
 40 ?DATREG=255: REM ALL LEDS OFF
 50 REPEAT
 60 A$=INKEY$(1)
 70 FOR N=0 TO 7
 80 ?DATREG=255-(2^N)
 85 FOR D=1 TO 200: NEXT D
 90 NEXT N
100 UNTIL A$<>""
110 END
```

LED-Lauflicht (II)

```
 10 REM CBM 64 VERSION 2.2
 20 DDR=56579: DATREG=56577
 30 POKE DDR,127: REM L7 INPUT
 40 POKE DATREG,255: REM ALL LEDS OFF
 50 GET A$
 60 FOR N=0 TO 7 STEP S
 70 POKE DATREG,255-(2^N)
 80 NEXT N
 90 IF PEEK (DATREG) AND 128=0 THEN S=-1
100 IF PEEK(DATREG) AND 128=1 THEN S=1
110 IF A$="" THEN 50
120 END
```

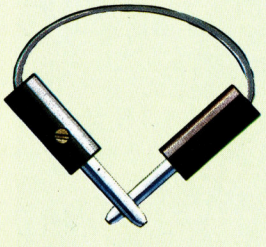
```
 10 REM BBC VERSION 2.2
 20 DDR=&FE62:DATREG=&FE60:S=1
 30 ?DDR=127: REM L7 INPUT
 40 ?DATREG=255: REM ALL LEDS OFF
 50 REPEAT
 60 A$=INKEY$(1)
 70 FOR N=0 TO 7 STEP S
 80 ?DATREG=255-(2^N)
 85 FOR D=1 TO 200:NEXT D
 90 IF DATREG AND 128=0 THEN S=-1 ELSE S=1
100 NEXT N
110 UNTIL A$<>""
120 END
```

LED-Lauflicht (III)

Damit drei LEDs zusammen aufleuchten, müssen Sie nur zwei Zeilen von Programm II ändern:

```
CBM VERSION
60 FOR N=2 TO 7 STEP S
70 POKE DATREG,255-((2^N)+2^(N-1)+2^(N-2))

BBC VERSION
70 FOR N=2 TO 7 STEP S
80 ?DATREG=255-((2^N)+2^(N-1)+2^(N-2))
```

Verbindungsstecker
Mit einem etwa 8 cm langen isolierten Stück Litze (etwa eine einzelne Ader vom Flachband-Kabel) werden ein roter und ein schwarzer Bananenstecker verbunden. Mit so einer „Brücke" können Sie zwei Buchsen verbinden. Sie brauchen acht Stück.



Würfel

```
 10 REM CBM 64 VERSION 2.3
 20 DDR=56579: DATREG=56577
 30 POKE DDR,127: REM L7 INPUT
 40 POKE DATREG,255: REM ALL LEDS OFF
 50 GET A$
 60 IF PEEK(DATREG) AND 128<>0 THEN 60
 70 NR=INT(RND(1)*6)+1: REM SELECT NO.
 80 POKE DATREG,255-((2^NR)-1)
 90 IF PEEK(DATREG) AND 128<>1 THEN 90
100 IF A$="" THEN 50
110 END
```

```
 10 REM BBC VERSION 2.3
 20 DDR=&FE62:DATREG=&FE60:S=1
 30 ?DDR=127: REM L7 INPUT
 40 ?DATREG=255: REM ALL LEDS OFF
 50 REPEAT
 60 A$=INKEY$(1)
 70 REPEAT UNTIL (?DATREG AND 128)=0
 80 NUMBER=RND(6): REM SELECT NUMBER
 85 FOR D=1 TO 200:NEXT D
 90 ?DATREG=255-((2^NUMBER)-1)
100 REPEAT UNTIL ?DATREG AND 128=1
110 UNTIL A$<>""
120 END
```

Ampel (I)

```
 10 REM CBM 64 VERSION 2.4
 20 DDR=56579: DATREG=56577
 30 POKE DDR,255: REM ALL OUTPUT
 40 POKE DATREG,255: REM ALL LEDS OFF
 45 REM BIT2=RED,BIT1=AMBER,BIT0=GREEN
 50 RD=255-4: AM=255-2:GN=255-1
 60 GET A$
 70 POKE DATREG,RD
 80 FOR N=1 TO 200:NEXT:REM DELAY LOOP
 90 POKE DATREG,RD+AM
100 FOR N=1 TO 40:NEXT:REM DELAY LOOP
110 POKE DATREG,GN
120 FOR N=1 TO 200:NEXT: REM DELAY LOOP
130 IF A$="" THEN 60
140 END
```

```
 10 REM BBC VERSION 2.4
 20 DDR=&FE62:DATREG=&FE60:S=1
 30 ?DDR=255:REM ALL OUTPUT
 40 ?DATREG=255: REM ALL LEDS OFF
 50 REM BIT2=RED, BIT1=AMBER,BIT0=GREEN
 60 RED=255-4:AMBER=255-2:GREEN=255-1
 70 REPEAT
 80 A$=INKEY$(100)
 85 FOR D=1 TO 200:NEXT D
 90 ?DATREG=RED
100 FOR N=1 TO 200:NEXT:REM DELAY LOOP
110 ?DATREG=RED+AMBER
120 FOR N=1 TO 40:NEXT:REM DELAY LOOP
130 ?DATREG=GREEN
140 FOR N=1 TO 200:NEXT:REM DELAY LOOP
150 UNTIL A$<>""
160 END
```

Ampel (II)

Ändern Sie die Lösung von Experiment 4 etwas ab:

```
CBM VERSION
30 POKE DDR,127: REM L7 INPUT
65 T=TI: REM INIT TIMER
75 IF PEEK(DATREG) AND 128=0 THEN C=C+1
76 IF PEEK(DATREG) AND 128<>1 THEN 76
77 IF C<=10 AND T-TI<3600 THEN 75


BBC VERSION
30 ?DDR=127: REM L7 INPUT
75 T=0:C=0
95 REPEAT
96 IF ?DATREG AND 128=0 THEN C=C+1
97 REPEAT UNTIL ?DATREG AND 128=1
98 UNTIL T>6000 OR C>10
```

Der ACT Apricot

Der Apricot wird von der britischen Firma ACT hergestellt, die auch den erfolgreichen Sirius-Computer baute. Das Gerät besteht aus drei Teilen – dem Prozessorgehäuse, einem Bildschirm und einer separaten Tastatur.

Der Apricot ist in zwei Versionen erhältlich: entweder mit zwei eingebauten Diskettenlaufwerken im 3 1/2-Zoll-Format oder mit einem 3 1/2-Zoll-Laufwerk und einer 10-MByte-Festplatte. Da der Apricot für den kommerziellen Einsatz konzipiert wurde, besitzt er weder Farbgrafik noch Cassettenrecorderanschluß, keine Joysticks und auch keine Fernseherbuchse. Seine Standardausrüstung umfaßt einen Monochrom-Monitor, einen parallelen Druckeranschluß, eine RS232-Schnittstelle, eine Buchse für eine Maus, Software und eine ausgezeichnete Tastatur.

Das interessanteste Merkmal des Apricot ist seine flexible und ungewöhnliche Tastatur. Neu ist dabei der Microscreen – eine LCD-Anzeige mit zwei Zeilen zu je 40 Zeichen, die sich rechts oberhalb des Tastenfeldes befindet. Beim Einschalten des Gerätes zeigt der Microscreen Wochentag, Monat, Jahr und Zeit an. Über ein Hilfsprogramm lassen sich Datum und Zeit ändern. Die batteriegetriebene Uhr des Computers läuft auch nach dem Abschalten weiter.

Einführung mit The Manager

Beim Einschalten des Gerätes wird automatisch ein Testprogramm aufgerufen, das die Größe des verfügbaren Speichers anzeigt (die standardmäßigen 265 KByte lassen sich auf 768 KByte ausbauen) und den Einsatz der MS-DOS-Systemdiskette verlangt. Anwender, die mit den Betriebssystemen CP/M und MS-DOS nicht vertraut sind, können über ein benutzerfreundliches Menü namens „The Manager" die Anwendungssoftware (zum Beispiel SuperCalc, Multiplan, Microsoft-BASIC) auswählen oder Hilfsprogramme (beispielsweise die Tastaturanpassung oder den Zeicheneditor) aufrufen.

Der Microscreen ist programmgesteuert und kann nicht nur Datum und Zeit anzeigen. Die Funktionen von sechs programmierbaren Tasten lassen sich jederzeit auf dem LCD-Schirm darstellen. Auch können Menüoptionen, die auf dem Monitor angezeigt werden, auch auf dem Microscreen erscheinen. Über die Funktionstasten lassen sich ebenso Abläufe auslösen wie über die Anwahl der entsprechenden Bildschirmoption mit den Cursortasten. Hierbei stört nur, daß die Membrantasten schwergängiger sind als die gewohnten Schreibmaschinentasten des übrigen Feldes.

Der ACT-Apricot sieht ungewöhnlich und interessant aus. Die Maschine zeichnet sich durch viele beachtenswerte Merkmale aus. Darin eingebaut ist ein 16-Bit-Microprozessor mit einem Standard von 265 K RAM. Mitgeliefert wird ein Monitor von hoher Qualität.

Es gibt weiterhin acht normale Funktionstasten, die mit Standardfunktionen wie HELP, PRINT, MENU, FINISH beschriftet sind. Über das mitgelieferte Hilfsprogramm Keyedit lassen sich alle Tasten neu definieren. Obwohl die Plazierung der Control- und Escape-Tasten etwas ungewöhnlich ist, entspricht die Tastatur dem, was man von einem kommerziellen Gerät erwartet. Für den Transport läßt sich die Tastatur in die Unterseite des Gerätes einrasten.

Im Lieferumfang des Apricot enthalten ist außer einer Reihe von Systemprogrammen auch SuperCalc, das mit dem Modul SuperPlanner für schnelle Berechnungen zur Verfügung steht. Mitgeliefert werden auch MS-DOS von Microsoft, CP/M-86 sowie Concurrent CP/M von Digital Research, ferner ein Kommunikationsprogramm, das den Datenaustausch mit anderen Apricots, dem Sirius und dem IBM PC ermöglicht. Von SuperCalc wird nur die Version Eins mitgeliefert. Version Zwei und Drei werden jedoch preiswert angeboten.

Kurz nach der Vorstellung des Apricot wurde kritisiert, daß das MS-DOS schlecht angepaßt und zu langsam sei. Das Problem scheint inzwischen jedoch gelöst zu sein. Doch obwohl die mitgelieferte Anwendungssoftware ausreichend schnell arbeitete und die Benchmark-Programme zum Testen des MSBASIC von Mi-

Kommerzielle Microdisketten
Apricot erhebt den Anspruch, der erste populäre Microcomputer zu sein, der im kommerziellen Bereich das kleine Diskettenformat einsetzte. ACT entschied sich für die Disketten von Sony im 3 1/2-Zoll-Format, die von einem harten Gehäuse umgeben sind. Sie sind robuster als die herkömmlichen.

Der Apricot XI
Da die Kapazität der Microdisketten begrenzt ist, bietet ACT den schwarzen Apricot XI an. Er besitzt eine 10-MB-Festplatte.

Bildschirm
Ein hochauflösender Bildschirm mit grün Phosphoranzeige bie eine gut lesbare An

crosoft ebenfalls mit annehmbarer Geschwindigkeit liefen, vermittelt der Apricot den Eindruck, daß er nicht so schnell arbeitet, wie man es von einer Maschine mit einem 8086-Prozessor erhofft.

Die Dokumentation von Apricot enthält eine Einführung für Anfänger, einen umfassenden Führer durch das MS-DOS-Betriebssystem, Leitfäden für SuperCalc und SuperPlanner und ausführliche Handbücher für WordStar und Multiplan. ACT bietet nur wenige Hardwareinformationen, die Systemprogramme für die Einrichtung des Computers lassen jedoch kaum einen Aspekt unberücksichtigt. Informationen über die Speicheraufteilung oder Systemaufrufe können angefordert werden.

Da der Apricot sehr auf den kommerziellen Einsatz ausgerichtet wurde, eignet er sich nicht für den Hobbyisten. Sollte das Gerät den gleichen Erfolg haben wie der Sirius von ACT, werden unabhängige Hersteller schon bald Zusatzgeräte anbieten, während ACT selbst weitere Speicherplatinen und ein Modem auf den Markt bringt. Selbst wenn der für einen kommerziellen Computer vertretbare Preis außer acht gelassen wird, ist der Apricot schon durch die verfügbare MS-DOS-Software attraktiv.

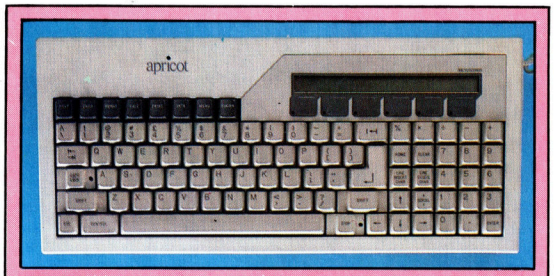
LCD-Anzeige
Die zweizeilige Flüssigkristallanzeige kann als Kalender/Uhr oder als Bildschirmersatz genutzt werden.

Sony-Microlaufwerke
Auf den 3½-Zoll-Microdisketten lassen sich je 315K speichern.

Funktionstasten
Diese Tasten enthalten Standardfunktionen wie HELP und REPEAT, die in verschiedenen Programmen zur Anwendung kommen.

Folientasten für Funktionsaufrufe
Die Funktionen dieser sechs programmierbaren Tasten können auf dem LCD-Schirm angezeigt werden.

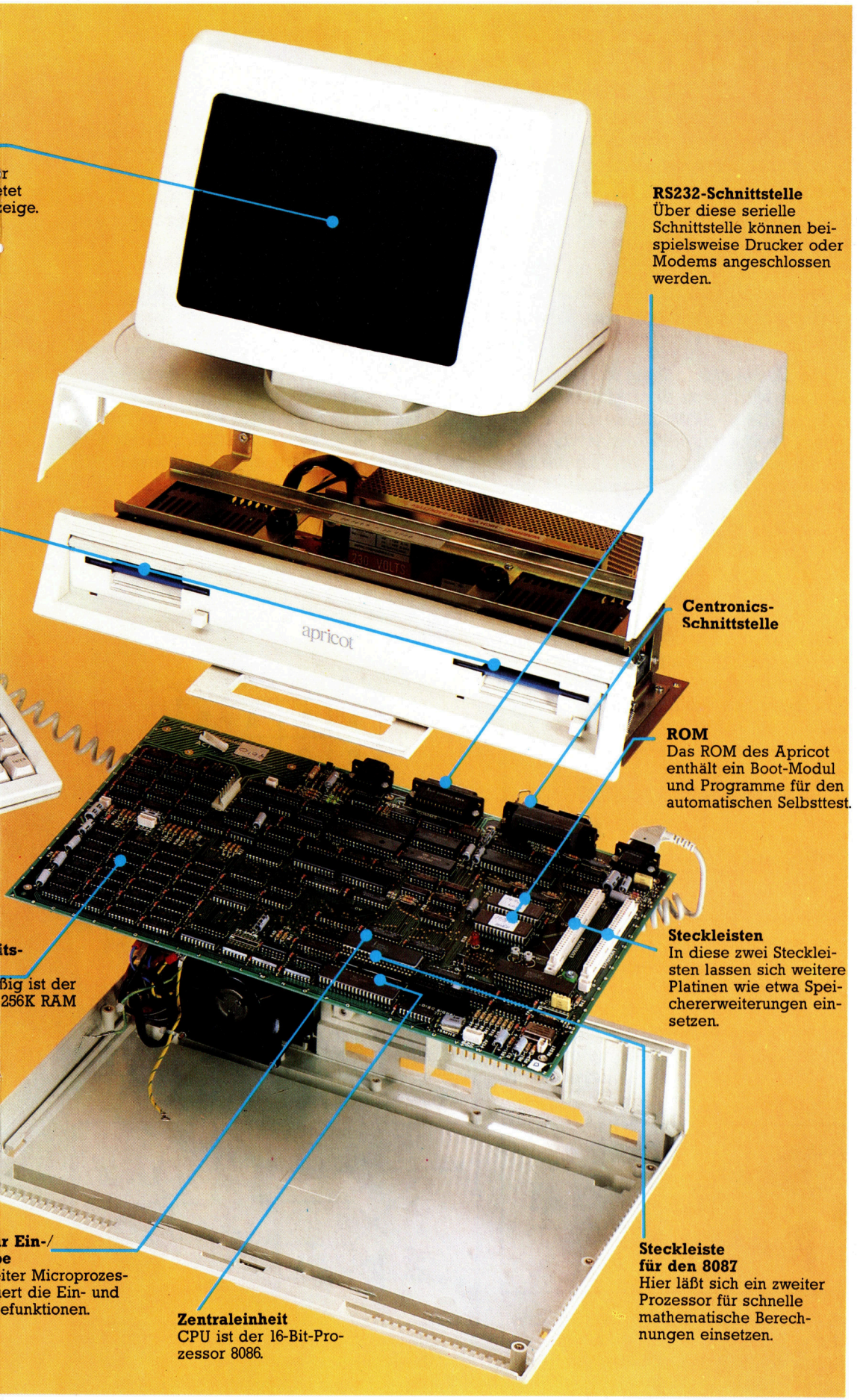
256K-Arbe speicher
Standardmä Apricot mit ausgerüstet



Die Tastatur des Apricot
Außer den hochwertigen Tasten besitzt der Apricot sechs Folientasten, die in verschiedenen Programmen Spezialfunktionen ausführen. Da diese Funktionen sich von Programm zu Programm verändern, zeigt der Apricot die augenblicklichen Funktionen auf dem LCD-Schirm an.

Der Monitor
Obwohl der Bildschirm eine Diagonalweite von nur 23 cm besitzt, kann er im Textmodus 50 Zeilen mit je 132 Zeichen darstellen. Standard sind jedoch 25 Zeilen zu je 80 Zeichen. Die Schrift ist gut lesbar, und der Bildschirm ist mit einer nicht reflektierenden Schicht ausgerüstet. Aufgrund der Größe des Monitors kann der Apricot nicht den Anspruch erheben, „tragbar" zu sein. Haupteinheit plus Monitor wiegen zwölf Kilo.

Chip fü Ausgab
Ein zwe sor steu Ausgab



ACT Apricot

ABMESSUNGEN

488 x 413 x 313 mm

ZENTRALEINHEIT

8086 mit der Möglichkeit, einen 8087-Mathematikprozessor zusätzlich anzuschließen.

SPEICHERKAPAZITÄT

256 K RAM, auf 768 K zu erweitern.

BILDSCHIRMDARSTELLUNG

Text kann mit 50 Zeilen zu je 132 Zeichen oder mit 25 Zeilen zu je 80 Zeichen dargestellt werden. Die grafische Auflösung beträgt 800 x 400 Punkte – einfarbig.

SCHNITTSTELLEN

Centronics, RS232, Buchse für eine „Maus", interne Steckleisten für Erweiterungen.

DISKETTENSTATIONEN

Ein oder zwei Laufwerke für Microdisketten von Sony. 315K oder 720K. Der Apricot XI hat eine 10-MB-Festplatte und ein Microlaufwerk.

BETRIEBSSYSTEME

MS-DOS, CP/M-86 und Concurrent CP/M-86.

TASTATUR

90 Schreibmaschinentasten und sechs programmierbare Folientasten, deren Funktionen auf der LCD-Anzeige dargestellt werden können.

HANDBÜCHER

Umfassend und gut aufgebaut.

STÄRKEN

Eine angenehme Maschine, die bessere Eigenschaften hat als viele Bestseller der kommerziellen Computer. Das Gerät sieht außerdem gut aus.

SCHWÄCHEN

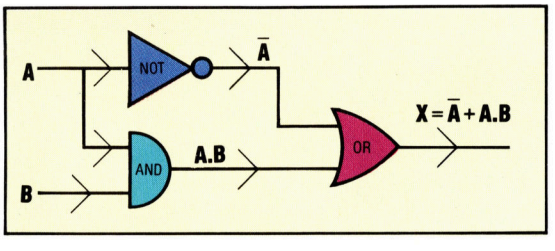
Obwohl der Apricot für seinen Preis einiges bietet, ist er für den Hobbyprogrammierer zu teuer.

Rechner-Addition

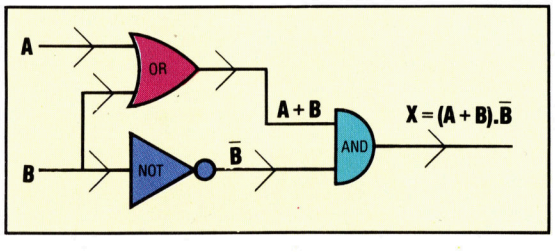
Im ersten Abschnitt des Kurses haben wir die drei grundlegenden logischen Operationen (AND, OR und NOT) sowie das Prinzip einfacher Logikschaltungen kennengelernt. Danach ist es nun leicht zu verstehen, wie der Computer eine Addition ausführt.

Die algebraische Behandlung logischer Beziehungen wird nach dem Mathematiker George Boole (1815–1864) „Boolesche Algebra" genannt. Im Computerbereich ist die Boolesche Algebra unverzichtbar, weil sie die mathematische Vereinfachung logischer Schaltungen ermöglicht. In der Praxis bedeutet das: weniger Gatterbausteine für eine gegebene Funktion – und damit eine höhere Arbeitsgeschwindigkeit.

Die Boolesche Symbolik für die Ausgabe der drei wichtigsten logischen Gatter kennen Sie bereits: AND (A.B), OR (A+B) und NOT ($\overline{A}$). Kompliziertere Schaltungen entstehen durch Zusammensetzung dieser Ausdrücke. $X=\overline{A}+A.B$ entspricht zum Beispiel dieser Schaltung:



Wichtig ist die Reihenfolge, in der die Operationen AND und OR ausgeführt werden. Dabei gilt die Regel AND vor OR (und vor NOT). Diese Reihenfolge läßt sich mit Hilfe von Klammern ändern. Beispiel: $X=(A+B).\overline{B}$. In diesem Ausdruck werden A OR B mit dem entgegengesetzten Wert von B durch AND verknüpft. Hier die Schaltung:



Meist ist es einfacher, Schaltungen, die den Booleschen Ausdrücken entsprechen, vom Ausgang her zu zeichnen – es wird auf diese Art übersichtlicher.

OR kann zwei unterschiedliche Bedeutungen annehmen. Als Booleschen Ausdruck kennen Sie es bisher in dieser Bedeutung:

Das eine OR das andere OR beide.

Es gibt aber eine zweite, für Logikschaltungen sehr wichtige andere Bedeutung dieses Ausdrucks:

Das eine OR das andere, aber nicht beide.

Ein Beispiel für das einschließende OR wäre, daß Sie zur Teilnahme an einem Zweirad-Rennen ein Fahrrad OR ein Motorrad haben müßten – Sie dürfen aber auch beides besitzen. Anders bei Gegenüberstellungen: Sie können nur entweder groß OR klein sein – da beides nicht gleichzeitig wahr sein kann, schließt dieses OR die Richtigkeit einer der beiden Behauptungen aus.

Das Exklusiv-OR

In Logikschaltungen wird das ausschließende OR (XOR) durch Kombination aus AND, OR und NOT-Gattern erzeugt.

Wahrheitstabelle:

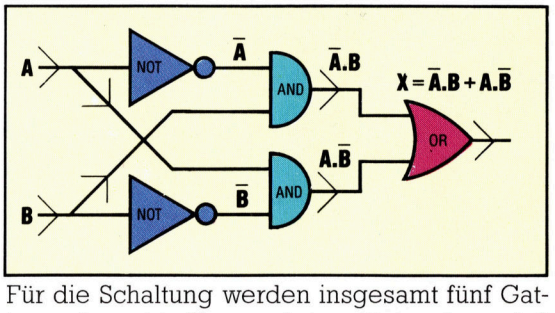
EINGÄNGE		AUSGANG
A	B	$A \veebar B$
0	0	0
0	1	1
1	0	1
1	1	0

Wie aus der zweiten und dritten Spalte sichtbar, erscheint am Ausgang die Eins, wenn

NOT(A) und B durch AND

OR

A und NOT(B) durch AND verknüpft werden. Der Boolesche Ausdruck dafür heißt $X=\overline{A}.B+A.\overline{B}$. Eine der möglichen Logikschaltungen für die ausschließende OR-Funktion (Exklusiv-OR) sieht so aus:



Für die Schaltung werden insgesamt fünf Gatter gebraucht. Sie werden später sehen, daß sich diese Anzahl durch Anwendung der Boo-

leschen Algebra auf vier vermindern läßt.

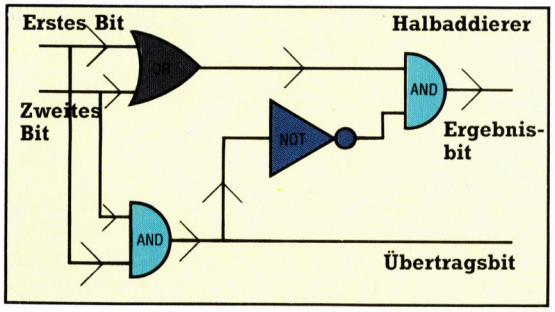
Obwohl die meisten Heimcomputer alle Rechenfunktionen beherrschen, wird nur die Addition von den üblichen Logik-Bausteinen direkt ausgeführt. Die anderen Rechenarten basieren auf der Verbindung integrierter Additions-Schaltungen mit spezieller Steuerungs-Software. Bevor wir näher auf die technische Realisierung der binären Addition eingehen, müssen wir einen Blick auf die Theorie des Additionsprozesses selbst werfen. Beispiel:

	8er	4er	2er	1er
	0	**1**	**0**	**1**
	0	**1**	**1**	**1**
	1	**1**	**0**	**0**
Übertrag	1	1	1	

Betrachten Sie die Zweier-Spalte allein, und untersuchen Sie die verschiedenen Ein- und Ausgaben von und zu dieser Spalte. Eingaben: Zwei Bits sollen addiert werden, außerdem kommt der Übertrag von der vorhergehenden Spalte hinzu. Ausgaben: ein Bit in die Zweier-Zeile des Ergebnisses, ein Bit Übertrag zur nächsten Spalte. Ein Addierer ist jedes Gerät, das diese Ein- und Ausgabeprozeduren fehlerlos vollführen kann. Volladdierer sind etwas komplizierter. Wir beginnen deshalb mit einem Halbaddierer, der den Übertrag von der vorhergehenden Spalte ignoriert. – Dadurch wird das Problem zweier Eingänge und zweier Ausgänge vorläufig umgangen. Die Wahrheitstabelle eines Halbaddierers sieht so aus:

EINGÄNGE		AUSGÄNGE	
Erstes Bit	**Zweites Bit**	**Übertragsbit**	**Ergebnisbit**
0	**0**	**0**	**0**
0	**1**	**0**	**1**
1	**0**	**0**	**1**
1	**1**	**1**	**0**

Es ist leicht ersichtlich, daß der Übertrag genau dann Eins wird, wenn das erste Bit AND das zweite Bit Eins sind. Das Ergebnisbit erhält man durch Verknüpfung beider Eingänge mit OR. Das Ergebnisbit ist also Eins, wenn das erste OR das zweite Bit Eins ist, aber NOT Eins, wenn das Übertragsbit Eins ist. Diese Schaltung erzeugt das Ergebnis:



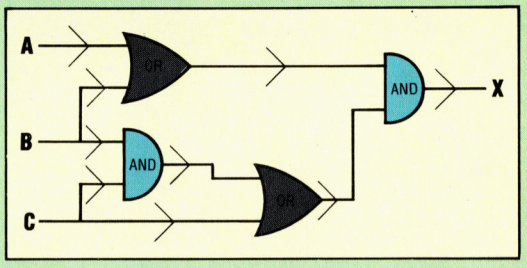
Übung 2

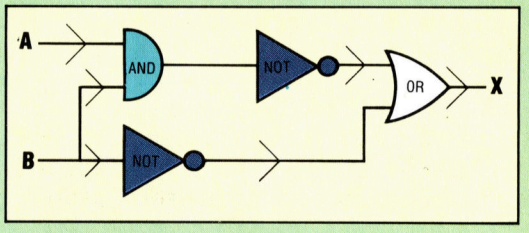
1) Zeichnen Sie zu den folgenden Booleschen Ausdrücken die entsprechende Logikschaltung:

a) $X=(A+B).C$
b) $X=A.B+(A+C)$
c) $X=\overline{A}.B+(A+B)$
d) $X=\overline{A.B}.(A+B)$

2) Schreiben Sie den Booleschen Ausdruck für einen Halbaddierer. Verwenden Sie A und B als Eingänge.

3) Schreiben Sie den Booleschen Ausdruck für die beiden abgebildeten Schaltungen auf.





Lösungen zu Übung 1

1)

Hat Führerschein	Fahrlehrer dabei	Darf fahren
Falsch	Falsch	Falsch
Falsch	Wahr	Wahr
Wahr	Falsch	Wahr
Wahr	Wahr	Wahr

2)

Cassettenrecorder vorhanden	Diskettenlaufwerk vorhanden	Programm für anderen Computer	Programm läuft
Falsch	Falsch	Falsch	Falsch
Falsch	Falsch	Wahr	Falsch
Falsch	Wahr	Falsch	Wahr
Falsch	Wahr	Wahr	Falsch
Wahr	Falsch	Falsch	Wahr
Wahr	Falsch	Wahr	Falsch
Wahr	Wahr	Falsch	Wahr
Wahr	Wahr	Wahr	Falsch

3)

A	B	P	Q	C
0	**0**	**1**	**0**	**0**
0	**1**	**1**	**1**	**1**
1	**0**	**0**	**1**	**0**
1	**1**	**0**	**1**	**0**

Sternensuche

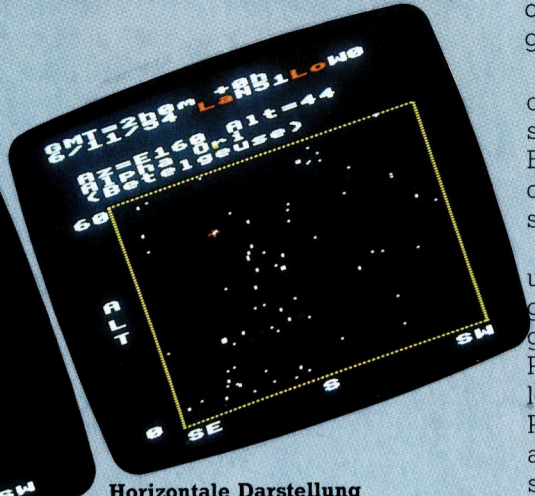
Für den Acorn B ist eine Fülle sogenannter „educational" Software entwickelt worden. Eines dieser Programme, „Starfinder", wurde speziell für Amateur-Astronomen gestaltet.

Astronomen haben stets mit einem speziellen Problem zu kämpfen – dem Wetter. Es ist keineswegs ungewöhnlich, daß professionelle Sternkundler Wochen damit verbringen, sich auf ein besonderes Himmelsereignis vorzubereiten, um am Tage des Geschehens festzustellen, daß eine dichte Wolkendecke die Sicht nimmt. Das ist der Grund dafür, warum die meisten Observatorien an sehr hohen Standorten gebaut werden, wo die Wolkenbildung gering ist – oder sogar im All.

Mit „Starfinder" hat „Century Software" das Universum auf einen kleinen Bildschirm gebracht. Die Packung enthält eine Cassette und ein Anleitungsbuch. Starfinder erlaubt dem Anwender die Betrachtung jedes Teils des Himmels von jedem beliebigen Teil der Erde zu jedem Zeitpunkt des 20. Jahrhunderts.

Nachdem das Programm geladen wurde, hat der Anwender die Auswahl zwischen verschiedenen Optionen. Eine davon ist der Blick auf den Himmel, wie er sich derzeit darstellt. Bei Beginn wird der Himmel so gezeigt, wie man ihn von London aus um Mitternacht (Greenwich Mean Time) am 21. November 1984 sah.

Vertikale Darstellung



Horizontale Darstellung

Im oberen Bildschirmteil werden Zeit und Datum angezeigt sowie die Beobachtungsposition in Längen- und Breitengrad. Unter dieser Information sieht man die Sternenkarte, die den Nachthimmel von Südosten nach Südwesten in einem Blickwinkel von 60 Grad zeigt.

Die Sterne werden als weiße Quadrate dargestellt, hellere Sterne sind größer als weniger helle. Bedauerlich ist, daß die Acorn-Rechner keinen „Helligkeitsbefehl" kennen, der eine realistischere Darstellung ermöglichen würde. Die Bildschirmausgabe der Planeten ist ebenfalls rechteckig, erfolgt aber in Rot, wogegen die Sonne als großes gelbes Rechteck gezeigt wird und der Mond als kleiner gelber Punkt. Durch Einsatz der sogenannten „Raumsonde", die durch die Cursor-Steuerungstasten bewegt wird, kann man die gezeigten Sterne identifizieren. Sobald die Raumsonde (ein rotes Kreuz) über einem Stern positioniert ist, werden über der Karte die astronomischen Angaben sowie der Name des betreffenden Himmelskörpers gleichzeitig mit den Raumkoordinaten gezeigt. Letztere werden in numerischer Form nach Höhe und Scheitelkreis ausgewiesen. (Die Höhenangabe erfolgt als positive oder negative Zahl, wobei der Horizont den Wert Null hat. Der Azimut wird in östlichem oder westlichem Grad in bezug auf Nord angegeben.)

Nach Rückkehr in das Hauptmenü ist es dem Benutzer möglich, den Beobachtungsstandort beliebig zu verändern, ebenfalls die Beobachtungszeit. Das Programm bietet außerdem die Option, beliebige Himmelskörper zu suchen – ob Stern, Planet oder Komet.

Das Programm ist zwar sehr umfangreich, unterliegt aber doch gewissen Beschränkungen. Eine der offensichtlichsten ist die begrenzte Anzahl der dargestellten Sterne. Der Programmierer Alpiar hat sich entschlossen, lediglich Sterne der 4. Größe und darunter ins Programm zu integrieren („Größe" bezieht sich auf die Helligkeit eines Sterns, wobei gilt: Je schwächer ein Stern leuchtet, desto höher ist der Größenwert). Das bloße Auge aber kann schon Sterne der 6. Größe wahrnehmen. Folglich sind im Programm nicht alle tatsächlich sichtbaren Sterne enthalten.

Das Anleitungsbuch erläutert den Gebrauch des „Starfinder", enthält aber außerdem Tips für Amateur-Astronomen.

Starfinder: Für den Acorn B und den Electron

Hersteller: Century Software, Portland House, 12–13 Greek Street, London, W1V 5LE

Autoren: Buch von Heather Couper, Programm von Alpiar

Joystick: Nicht erforderlich

Format: Cassette

Digital Research

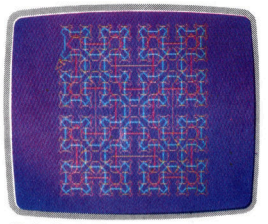
Seit seiner Einführung ist das Control Program for Microprozessors (CP/M) zum Industriestandard unter den Betriebssystemen geworden. Der phänomenale Erfolg von CP/M hat das Leben seines Entwicklers Gary Kildall völlig verändert. Er gab seinen Beruf als Dozent auf und gründete ein eigenes Unternehmen, Digital Research.

Gary Kildall schrieb 1972 seine erste CP/M-Version, um einen Compiler (ein Übersetzungsprogramm) für PL/M zu schaffen. 1975 schrieb er einen Editor (ED), einen Assembler (ASM) und einen Debugger (Korrektur- bzw. Fehlersuchprogramm) (DDT). Er bot das neue Betriebssystem Intel zur Nutzung an, doch das Unternehmen lehnte ab. Das war das Beste, was Kildall passieren konnte. Gemeinsam mit Dorothy McEwan begann er, eine Reihe von Computermagazinen für Hobbyisten zu publizieren und verkaufte CP/M privat. Kildalls CP/M erzielte bald bessere Erfolge als die Magazine.

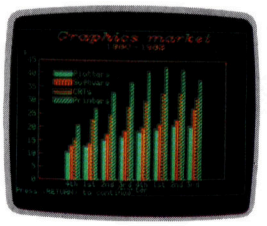
Ob es nun an der Programmart lag oder reine Glücksache war: Kildall hatte mit diesem Betriebssystem genau das geschaffen, was eine der „Kinderkrankheiten" der Microcomputer „heilte", nämlich die mangelnde Kompatibilität. Die Ende der siebziger Jahre wichtigsten Computer für den privaten Bereich (PET, Apple und Tandy) waren mit inkompatiblen Diskettenbetriebssystemen ausgestattet. Software-Anbieter mußten sich für eines der Systeme entscheiden. Der Code mußte jeweils völlig neu erstellt werden, um Software auf einer anderen Maschine lauffähig zu machen. CP/M änderte all das: Seine Popularität war Grund für die meisten Hersteller, es zu adaptieren. So wurde ein echter Standard geschaffen. Viele Computerhersteller, die für den Bau ihrer Rechner den Microprozessor 8080 von Intel oder den Zilog Z80 verwendeten, entschieden sich wegen der einfachen Handhabung und des unkomplizierten Zugriffs zu Bildschirm, Drucker, Diskettenstation und Keyboard für CP/M. Mit wachsender Popularität war immer mehr CP/M-Software verfügbar, womit gleichzeitig immer mehr Firmen dieses Betriebssystem in ihre Rechner integrierten.

Schon 1976 war Kildall vom Erfolg seines Produktes und der Nachfrage so überwältigt, daß er seine Professur als Computerwissenschaftler an einer Marinehochschule in Monterey aufgab und in Pacific Grove, Kalifornien, die Firma Digital Research gründete.

CP/M fand weiteren Zuspruch. Indessen widmete sich Digital Research den Mehrplatz-Systemen und entwickelte MP/M. Es war in jeder Hinsicht mit CP/M kompatibel, hatte aber nicht denselben Erfolg. Die Aufteilung der Anwenderbereiche und andere Konfigurationen, die für einen Systemprogrammierer sinnvoll sein mögen, waren keineswegs einfach in der Handhabung. In manchen Bereichen wich die Arbeitsmethode von der bei CP/M auch ab. Die rein technischen Kosten für die Herstellung von Microprozessoren sind bei gleichzei-



Digital Research entwickelt Sprachen und ist mit DR LOGO Marktführer. Wie bei allen guten LOGO-Programmen ist auch hier Grafik der wichtigste Faktor.



GSX ist ein bahnbrechendes Software-Paket, das Grafikprogramme zwischen unterschiedlichen Rechnern übertragen kann, so die hier gezeigten Geschäftsgrafiken.

John Rowley, Präsident von Digital Research Incorporated, rechts Gary Kildall

Der Firmensitz von Digital Research Incorporated ist in Massachusetts, USA.

tig steigender Produktion erheblich gesunken. Daher sank auch der Bedarf für die gleichzeitige Nutzung eines Prozessors durch mehrere Anwender merklich. Somit wurde das Ganze unwirtschaftlich.

1981 erhielt Digital Research eine Finanzspritze durch Risiko-Kapitalgesellschaften und weitete das Unternehmen zu einem echten Multi aus. Das Unternehmen ist in Europa besonders weit verbreitet. Hier gibt es Niederlassungen in England, Deutschland und Frankreich. Ungefähr zur selben Zeit war Digital Research als eines der ersten Häuser darum bemüht, einen Vertrag für die Entwicklung eines Betriebssystems für den neu entwickelten Personal Computer von IBM zu bekommen. Zwar erhielt Microsoft den IBM-Auftrag, doch Digital Research war keineswegs geschlagen. Das CP/M für die Intel-8088/8086-Prozessoren wurde überarbeitet und dem MS-DOS angepaßt.

Entwicklungen nur mit C

Eine der strategischen Entscheidungen bei Digital Research, aber auch vieler anderer System- und Sprachentwicklungshäuser, war, die gesamte Entwicklungsarbeit in der C-Sprache durchzuführen. Das ist für die Übersetzung besonders wichtig. In C geschriebene Codierungen müssen, um auf anderen Microprozessoren zu laufen, lediglich neu kompiliert werden. Allerdings brachte dies den Vorwurf ein, die Codierung sei sehr schwerfällig. Die Methode hat mittlerweile ungeheuer an Popularität gewonnen, und da das weit verbreitete UNIX-Betriebs-System auch in C geschrieben wurde, scheint der Trend zu dieser Sprache unaufhaltsam zu sein.

Heute bietet Digital Research eine Vielzahl von Sprachen für eine Reihe von Microcomputern an. Der untere Marktbereich bedient die Consumer Products Division von Digital Research mit Personal-BASIC, Personal-CP/M und einer eigenen LOGO-Version. Wie CP/M-86 ist auch Personal-CP/M auf ROM gespeichert und soll laut Vertrag mit Zilog bald auch auf einem Z80-Chip erhältlich sein. Digital Research nennt das „Microware" und ist davon überzeugt, die Vielzahl alternder Standard-CP/M-Programme dadurch zu aktivieren bzw. ihre Lebensdauer zu verlängern, daß man sie preiswert genug macht, um den Heimcomputer-Anwender zum Kauf zu motivieren.

Weitere vielversprechende Entwicklungen sind VIP und GSX. VIP ist ein preiswertes „Gerüst", das es Programmautoren möglich macht, unabhängig von den verwendeten Anwendungsprogrammen eine einheitliche Schnittstelle anzusteuern. Mehrere Applikationen nutzen dieselben Datas, und diese können vom einen zum anderen übertragen werden. Unter diesem Gesichtspunkt ist VIP Apples Lisa- und Macintosh-Technik sehr ähnlich, benötigt aber viel weniger Speicherplatz.

Was CP/M für den Diskettenbetrieb ist, ist GSX im Grafikbereich. Das Programm nutzt eine standardisierte Reihe von Grafikfunktionen, die auf verschiedenen Systemen läuft. Ein GSX-Programm kann auf einem Farbbildschirm oder einem Schwarzweißschirm laufen, auf einem Matrix-Drucker oder einem Plotter, ohne daß es einer Veränderung bedarf.

Digital Research hat sich als eines der wichtigsten Software-Häuser im Microcomputergeschäft etabliert. Doch man ruht sich nicht auf seinen Lorbeeren aus. Nach Entwicklung der Programme GSX, VIP und DR LOGO ist die Firma nun daran interessiert, wie Microsoft im Bereich der Anwendungsprogramme tätig zu werden. In Anbetracht des CP/M-Erfolges hat die Gesellschaft beste Zukunftschancen.

Last In First Out

Der Stack (Stapel) ist ein fest definierter Speicherbereich des Computers. Er steht der CPU für „Notizen" zur Verfügung und spielt beim Ablauf von Unterroutinen eine wichtige Rolle. In dieser Folge erklären wir Aufgaben und Funktionsweise des Stacks.

Die Speicherverwaltung ist die Grundlage der Assemblerprogrammierung. Fast alle bisher untersuchten Befehle übertragen Daten in bestimmte Speicherstellen oder laden sie von dort. Speicherstellen können über verschiedene Adressiermethoden angesprochen werden, wobei der Operand fast immer eine Speicheradresse enthält. Es gibt jedoch eine Befehlsklasse, die ohne Adreßoperanden auf den Speicher zugreift. Diese Befehle arbeiten mit einem Speicherbereich, der als Stack (Stapel) bezeichnet wird.

Der Stack steht der CPU und dem Programmierer als Kurzzeitspeicher zur Verfügung. Er ist eine Art „Notizblock", auf dem Daten notiert werden, die schnell gelesen und auch schnell wieder gelöscht werden können. Die Stackbefehle kopieren Daten der CPU-Register in freie Stapelbereiche oder laden sie wieder in die CPU-Register zurück. Da der Stack Pointer (Stapelzeiger) – selbst ein CPU-Register – immer die Adresse des nächsten freien Stackbytes enthält, benötigen diese Befehle keinen Adreßoperanden. Alles, was auf dem Stack abgelegt werden soll, wird automatisch auf das Byte geschrieben, dessen Adresse der Stack Pointer enthält, während alle Daten, die vom Stack heruntergezogen werden, immer von dem Byte des Stacks kommen, das zuletzt beschrieben wurde.

Der Stack des 6502 besteht aus den 256 RAM-Bytes von $0100 bis $01FF, wogegen im Z80 Systembereich und Größe des Stacks variabel sind und vom Betriebssystem bestimmt werden. Der Stack Pointer des 6502 besteht aus einem Byte, der des Z80 aus zwei Bytes.

Der Stack Pointer

Der 6502 sieht den Inhalt des Stack Pointers als niederwertiges Byte der Stackadresse an, auf das automatisch $01 – das „neunte Bit" des Stack Pointers – als höherwertiges Byte addiert wird. Da das zusätzliche Bit immer gesetzt (bzw. „Eins") ist, liegen alle Stackadressen des 6502 auf der Seite eins des Speichers.

Mit dem Stack Pointer des Z80 lassen sich Adressen zwischen $0000 und $FFFF (der gesamte Adreßraum des Z80) ansprechen. Der Stack kann daher an jeder beliebigen Stelle des RAMs liegen. Eine Verlegung des Stapelbereichs ist jedoch nicht zu empfehlen, da das Betriebssystem diesen Bereich unmittelbar nach dem Laden festlegt und dort alle Daten speichert.

Die folgende Routine tauscht die Inhalte der Speicherstellen LOC1 und LOC2 gegeneinander aus:

6502		Z80	
LDA	LOC1	LD	A,(LOC1)
PHA		PUSH	AF
LDA	LOC2	LD	A,(LOC2)
STA	LOC1	LD	(LOC1),A
PLA		POP	AF
STA	LOC2	LD	(LOC2), A

Der Inhalt von LOC1 wird in den Akkumulator geladen und von dort auf den Stack gelegt (PUSH). Danach wird der Inhalt von LOC2 in den Akkumulator geladen und auf LOC1 kopiert. Durch Laden des obersten Stackbytes in den Akkumulator (POP) befindet sich nun der ursprüngliche Inhalt von LOC1 im Akkumulator. Das Kopieren des Akkumulatorinhalts auf LOC2 beendet das Programm. Bemerkenswert ist dabei, daß der Stackbefehl den Inhalt von LOC1 ohne Angabe einer Speicheradresse (nur durch die implizite Angabe des nächsten freien Stackbytes) speichert.

Dieses Programm gibt Einblick in die Methodik des Stacks. Außer der Umkehrung des Vorgangs beim Laden und Lesen fällt auf, daß alle Daten sequentiell bearbeitet werden: Zuletzt auf den Stack gelegte Daten werden als erste wieder heruntergezogen. Mehrere Informationen, die nacheinander – ohne dazwischenliegende Leseoperation – auf den Stack geschrieben werden (PUSH), sind in aufeinanderfolgenden Adressen des Stacks abgelegt.

Stellen Sie sich den Stack als eine Anzahl Karten vor, die Sie mit Notizen beschreiben und auf Ihrem Schreibtisch stapeln. Im weiteren Verlauf nehmen Sie die jeweils oberste Karte herunter, lesen diese und werfen sie weg. Die Karte, die Sie als letztes beschrieben haben, liegt immer oben auf dem Stapel. Die Datenstruktur des Stapels wird daher auch als LIFO (Last In – First Out: Das zuletzt Eingegebene wird zuerst wieder ausgegeben) bezeichnet. Die Umkehrung dieser Datenstruktur nennt man FIFO (First In – First Out, die zuerst eingegebenen Daten werden auch zuerst wieder ausgegeben).

In unserem Programmbeispiel wurden ebensoviele Daten auf den Stack geschoben wie heruntergezogen werden. Auf dieses Gleichgewicht braucht zwar nicht unbedingt Rücksicht genommen zu werden, doch können Unterroutinen bei Nichtbeachtung dessen leicht falsche Rücksprungadressen erhalten, die zum Programmabbruch führen. Dieser häufig auftretende Fehler läßt sich jedoch vermeiden, wenn man die Zahl der PUSH- und POP-Befehle miteinander vergleicht.

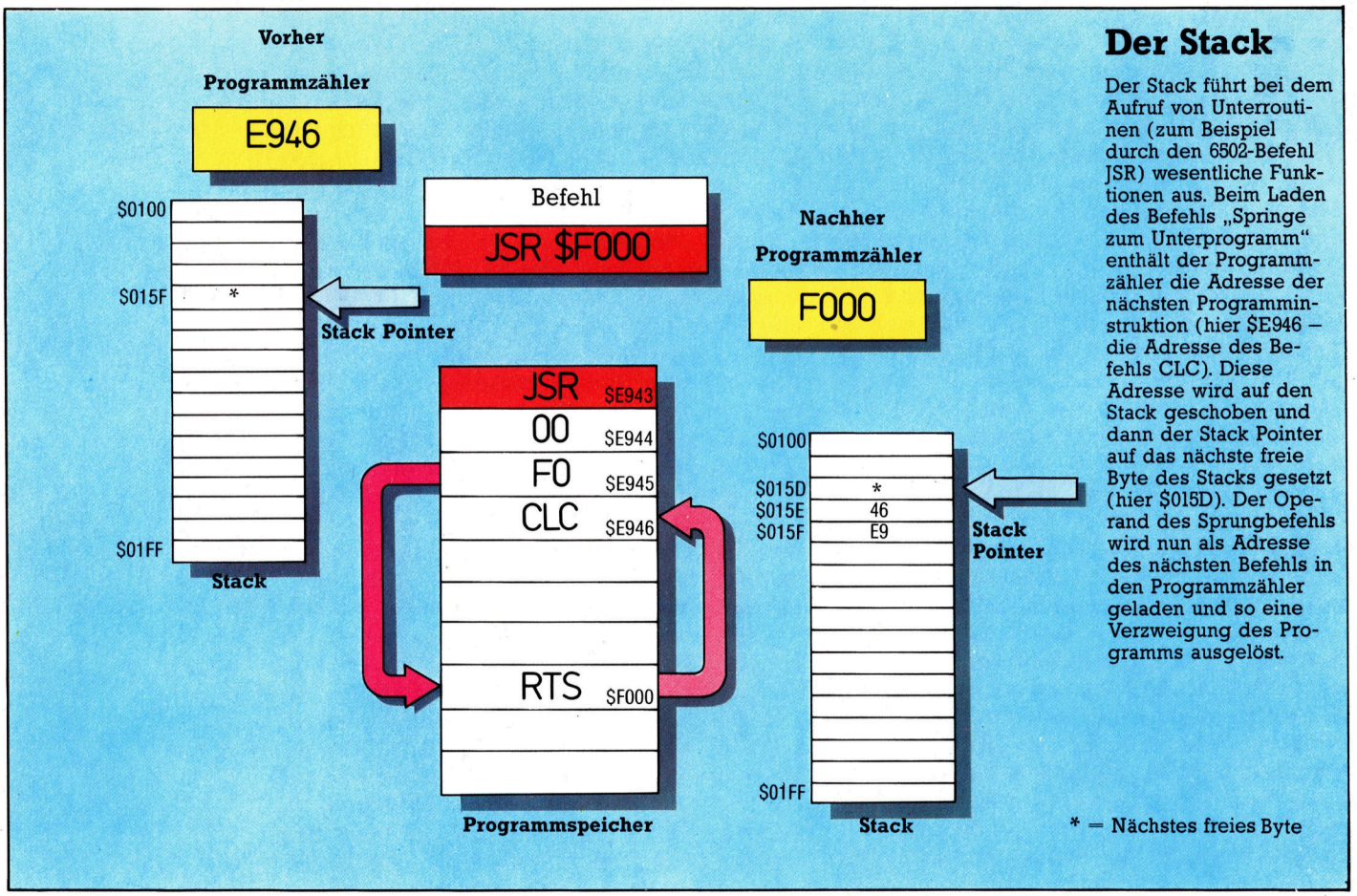
Zwei-Byte-Register des Z80

Die Version des Z80 unterscheidet sich in einem wesentlichen Aspekt von der des 6502: Während der 6502 nur Ein-Byte-Register auf den Stack schieben kann, speichert der Z80 dort ausschließlich Zwei-Byte-Register. Wenn Sie mit PUSH oder POP den Akkumulator des Z80 ansprechen, greifen Sie auch auf das Prozessor-Status-Register zu, da die CPU diese beiden (Ein-Byte-) Register als ein Zwei-Byte-Register mit dem Namen AF (Akkumulator-Flag) behandelt. Die universelle Einsetzbarkeit des Z80 beruht zum großen Teil auf seiner Fähigkeit, Zwei-Byte-Register verarbeiten zu können.

Es ist guter Programmierstil, den Inhalt aller CPU-Register am Anfang einer Unterroutine auf den Stack zu schieben und sie vor dem Rücksprung wieder zurückzuladen. Auf diese Weise stellen Sie sicher, daß die CPU nach dem Rücksprung den gleichen Zustand hat wie vorher, und brauchen außerdem nicht zu befürchten, daß wichtige Daten des Hauptprogramms verändert wurden. Sehen Sie sich folgendes Unterprogramm an:

	6502		Z80	
	LDA	LOC1	LD	A,LOC1
SUM	ADC	#$6C	ADC	A,$6C
GSUB	JSR	SUBR0	CALL	SUBR0
TEST	BNE	SUM	JR	NZ,SUM
EXIT	RTS		RET	
SUBR0	PHP		PUSH	AF
	PHA		PUSH	HL
	TXA		PUSH	DE
	PHA		PUSH	BC
	TYA		PUSH	IX
SUBR1	PHA		PUSH	IY
SUBR2	STA	LOC2	LD	(LOC2),A
	LDA	#$00	LD	A,$00
SUBR3	PLA		POP	IY
	TAY		POP	IX
	PLA		POP	DE
	TAX		POP	BC
	PLA		POP	HL
SUBR4	PLP		POP	AF
	RTS		RET	

Die Befehle zwischen SUBR0 und SUBR1 schieben die aktuellen Registerinhalte auf den



Der Stack

Der Stack führt bei dem Aufruf von Unterroutinen (zum Beispiel durch den 6502-Befehl JSR) wesentliche Funktionen aus. Beim Laden des Befehls „Springe zum Unterprogramm" enthält der Programmzähler die Adresse der nächsten Programminstruktion (hier $E946 – die Adresse des Befehls CLC). Diese Adresse wird auf den Stack geschoben und dann der Stack Pointer auf das nächste freie Byte des Stacks gesetzt (hier $015D). Der Operand des Sprungbefehls wird nun als Adresse des nächsten Befehls in den Programmzähler geladen und so eine Verzweigung des Programms ausgelöst.

Stack, während die Instruktionen zwischen SUBR3 und SUBR4 sie wieder in die entsprechenden Register laden. Die beiden auf SUBR2 folgenden Befehle sind der eigentliche Inhalt der Routine, wobei der zweite Befehl gegenstandslos ist, da die darauf folgenden Instruktionen den Akkumulator wieder verändern.

Beachten Sie hierbei, daß die Z80-Befehle PUSH und POP jedes Registerpaar als Operanden enthalten können, während sich auf dem 6502 nur der Akkumulator (PHA und PLA) und das Prozessor-Status-Register (PHP und PLP) ansprechen lassen. Der 6502 braucht daher zusätzliche Befehle für die Datenübertragung zwischen Registern und Akkumulator (TXA, TAX, TYA, TAY). Beachten Sie weiterhin, daß wir in die Version des Z80 absichtlich einen Fehler eingebaut haben, indem wir nicht alle Register in umgekehrter Reihenfolge vom Stack herunterziehen. Hier wird deutlich, mit welcher Vorsicht Stackvorgänge programmiert werden müssen. Das Beispiel zeigt auch, wie der Stack den Inhalt eines Registers in ein anderes übertragen kann – eine aufwendige, aber zuweilen sehr praktische Methode, um zwischen Registern Daten auszutauschen.

In der nächsten Folge gehen wir auf die Funktionen und Einsatzmöglichkeiten der CPU-Register ein und schließen damit die allgemeine Einführung ab.

Lösung der Übungen unserer vorigen Folge

1) Diese Unterroutine speichert in absteigender Reihenfolge die Zahlen $0F bis $00 in einem Block von $10 Bytes, der von dem Pseudo-Op-code DS bei LABL1 reserviert wurde.

6502		Z80	
ORIGIN	ORG $7000	ORIGIN	ORG $C000
LABL1	DS $10	LABL1	DS $10
LABL2	DW $7100	LABL2	DW $C100
		OFFST	EQU $0F
BEGIN	LDY #$FF	BEGIN	LD IX,LABL1
	LDX #$10		LD B,OFFST
LOOP0	INY	LOOP0	LD (IX+0),B
	DEX		INC IX
	TXA	ENDLP0	DJNZ LOOP0
	STA LABL1,Y		LD (IX+0),B
ENDLP0	BNE LOOP0		RET
	RTS		

Der 6502 nimmt das Y-Register als Index der Adresse LABL1, während das X-Register als Schleifenzähler und als Ursprung der zu speichernden Daten dient. Beachten Sie, daß das X-Register bereits zwei Befehle vor dem BNE-Test bei ENDLP0 dekrementiert wird. Da die Befehle TXA („Übertrage Register X in den Akkumulator") und STA keinen Einfluß auf das Prozessor-Status-Register haben, verändert sich das Dekrementierungsergebnis dadurch nicht.

Die Z80-Version setzt als Zeiger der Speicheradresse das indirekt adressierte IX ein und nimmt das B-Register als Zähler und Datenquelle. Neben dem ENDLP0 der Z80-Version sehen wir den Befehl DJNZ LOOP0, der „Dekrementiere das Register B und führe einen relativen Sprung aus, falls das Ergebnis nicht Null ist" bedeutet.

2) Diese Routine kopiert die in LABL1 gespeicherte Meldung auf einen Block, der bei der in LABL2 gespeicherten Adresse beginnt. Der Wert $0D (der ASCII-Code für Return oder Enter) ist am Ende der Meldung als Abschlußzeichen gespeichert.

6502		Z80	
ORIGIN	ORG $7000	ORIGIN	ORG $C000
LABL1	DB 'THIS IS A MESSAGE'	LABL1	DB 'THIS IS A MESSAGE'
TERMN8	DB $0D	TERMN8	DB $0D
LABL2	DW $7100	LABL2	DW $C100
CR	EQU $0D	CR	EQU $0D
ZPLO	EQU $FB		
BEGIN	LDA LABL2	BEGIN	LD IX,LABL1
	STA ZPLO		LD IY,(LABL2)
	LDA LABL2+1	LOOP0	LD A,(IX+0)
	STA ZPLO+1		LD (IY+0),A
	LDY $FF		INC IX
LOOP0	INY		INC IY
	LDA LABL1,Y		CP CR
	STA (ZPLO),Y	ENDLP0	JR NZ,LOOP0
	CMP CR		RET
ENDLP0	BNE LOOP0		
	RTS		

Die Version des 6502 verwendet das Y-Register im nach-indizierten indirekten Modus als Index der indirekten Adresse ZPLO. Diese Adressierungsmethode ist nur mit dem Y-Register möglich und erfordert eine Operandenadresse, die auf der Seite Null liegt – daher die Initialisierung von ZPLO und ZPLO+1 mit der in LABL2 gespeicherten Adresse. Die Betriebssysteme von 6502er Maschinen setzen normalerweise fast alle Speicheradressen der Seite Null ein. Da auf dem Commodore 64 die Adressen von $FB bis $FF und auf dem Acorn B die Adressen von $70 bis $8F jedoch ungenutzt sind, wird ZPLO auf eine dieser Speicherstellen gelegt. Die Z80-Version verwendet IX im indizierten Modus und IY im indizierten indirekten Modus.

Beide Routinen verwenden einen Befehl zum Akkumulatorvergleich – CMP CR (6502) und CP CR (Z80) – bei dem die Flags des Prozessor-Status-Registers (PSR) durch eine Subtraktion des Operanden vom Akkumulatorinhalt beeinflußt werden. Wenn der Akkumulator $0D enthält, wird das Nullflag gesetzt. Dadurch ist bei ENDLP0 die Bedingung für eine Verzweigung nicht erfüllt, und die Steuerung wird an den Rücksprungbefehl übergeben.

Übungen

1) Ändern Sie die zweite Programmroutine in den Lösungen so, daß die Meldung von LABL1 in umgekehrter Reihenfolge wieder in LABL1 gespeichert wird:

LABL1 EGASSEM
A SI SIHT

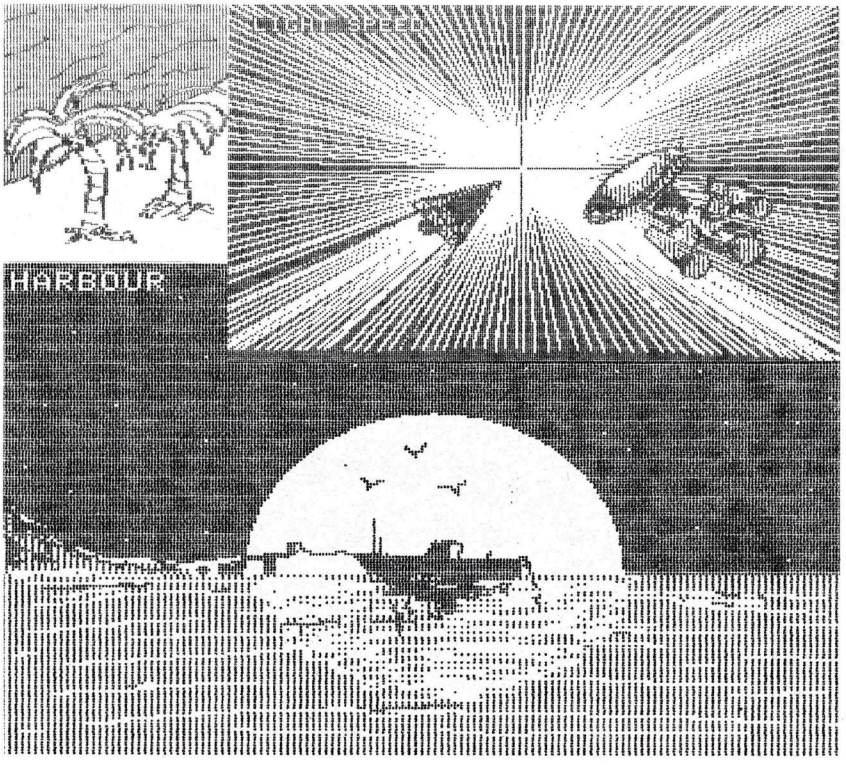
Verwenden Sie dazu den Stack.

2) Ändern Sie die Routine wiederum, so daß die Worte der Meldung in ihrer ursprünglichen Reihenfolge bleiben, die Zeichen jedes Wortes jedoch umgekehrt werden.

LABL1 SIHT SI
A EGASSEM

Drucker-Grafik

Die Fähigkeiten von Matrix-Druckern werden häufig nicht genügend beachtet. In diesem Beitrag wird gezeigt, wie man mit einem solchen Gerät interessante Bilder macht.



Diese Muster wurden mit einem Zeichentablett erzeugt. Anschließend wurden die Bilder auf einen Epson FX-80 übertragen. Ein Beispiel für die Grafikmöglichkeiten eines Matrix-Druckers.

Die meisten Heimcomputer verfügen über einen niedrigauflösenden Grafik-Modus, in dem Bilder aus Grafikzeichen aufgebaut werden, die alle die gleiche Größe haben wie ein normales Textzeichen. Die Zeichencodes dieser „Block"-Charaktere liegen über 127, da die Zahlen 0 bis 127 ausschließlich für den ASCII-Zeichensatz reserviert sind. Folglich würde PRINT CHR$(90) ein ASCII-Zeichen auf dem Bildschirm erzeugen – in unserem Fall ein „Z" – wogegen PRINT CHR$(128) ein Grafikzeichen darstellt.

Um den Buchstaben „Z" auszudrucken, müßte man LPRINT CHR$(90) eingeben. Man könnte davon ausgehen, daß der Befehl LPRINT CHR$(128) nun den Ausdruck eines bestimmten Grafikzeichens zur Folge hätte. Dies aber ist leider nicht der Fall. Grund dafür ist, daß die Zeichen mit einem Wert über 127 zwischen den einzelnen Rechnern erheblich variieren. Und die Hersteller von Druckern sind offensichtlich außerstande, spezielle Drucker für die im Angebot befindlichen Computer zu produzieren. Entweder versuchen sie, den Standard-ASCII-Zeichensatz in die Codierungen von 128 bis 255 zu übertragen oder programmieren alternativ ihre selbst erstellten Grafikzeichen.

Hochauflösende Computergrafiken werden aus kleinen Punkten zusammengesetzt. Im Druckerkopf eines Matrix-Druckers befinden sich mehrere Nadeln, die sich beim Druck über das Papier vertikal bewegen. Gewöhnlich erfolgt die Erzeugung von Zeichen durch ein Raster. Allerdings ist es auch möglich, Grafiken durch Steuerung der einzelnen Nadeln individuell zu erzeugen.

Der erste Schritt dazu ist das Umschalten des Druckers in den Grafik-Mode. Wie bei jedem anderen Druckbefehl ist das nichts weiter als das Aussenden eines „Escape Code", der den Eigenarten des verwendeten Druckers entspricht. Beim Epson FX-80 beispielsweise lautet die Anweisung:

```
LPRINT CHR$(27); "K";CHR$(N1);(N2);
```

Der Buchstabe "K" steht für den Grafik-Mode, und die Zahlen (N1) und (N2) geben die Linienbreite der Grafiken an. Anders gesagt: Sie geben an, wieviele Punkte tatsächlich auf eine Zeile passen.

Im Standard-Grafik-Modus kann der FX-80 maximal 480 Punkte pro Zeile drucken. Andere Modi erlauben eine Auflösung von 576 bis zu 1920 Punkte pro Zeile. Soll die gesamte Breite genutzt werden, muß man 480 eingeben. Im vorliegenden Code sind beide Zahlen zur Festlegung der Breite erforderlich, da der Wert von N jeweils maximal nur 255 betragen kann. Die zweite Zahl (N2) wird deshalb mit 256 multipliziert und zur ersten addiert (N1). Um 480 zu erzeugen, benötigen wir also die Zahlen 1 und 224 (480=224+256x1). Daher wird für den Epson FX-80 folgende Anweisung eingegeben:

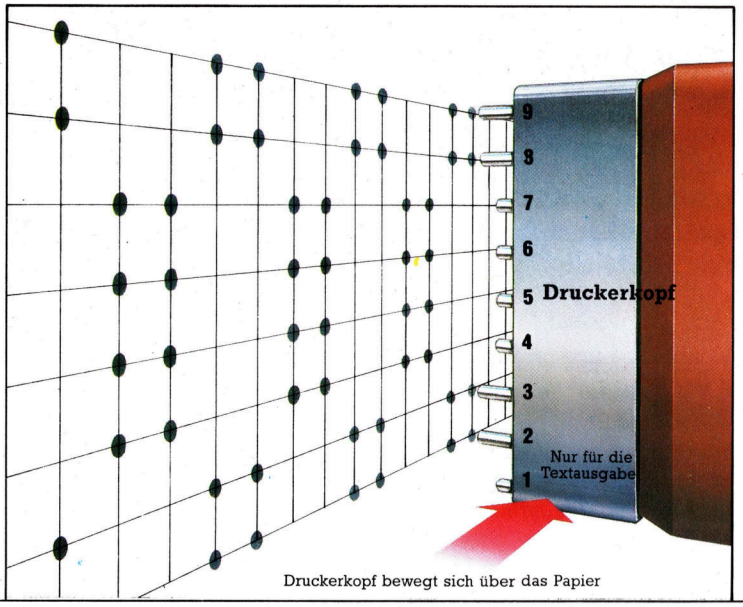
```
LPRINT CHR$(27);"K";CHR$(224);CHR$(1);
```

Nach der Programmierung des Druckers mit der Zeilenlänge müssen die Grafik-Datas gesendet werden. Obwohl der Druckerkopf des Epson FX-80 über neun Nadeln verfügt, können in den meisten Grafik-Modi nur die oberen acht verwendet werden. Mit der unteren Nadel beginnend, werden diese mit 1,2,4,8,16,32,64 und 128 numeriert. Die Datas für die acht Nadeln können dann durch eine einzelne Zahl zwischen 0 und 255 dargestellt werden, die mit der Anweisung LPRINT CHR$(X) an den Drucker gesendet wird. X ist in diesem Fall die Zahl. Soll also nur die untere Nadel aktiviert werden, würden wir CHR$(1) an den Drucker senden. Um nur den oberen Pin aus-

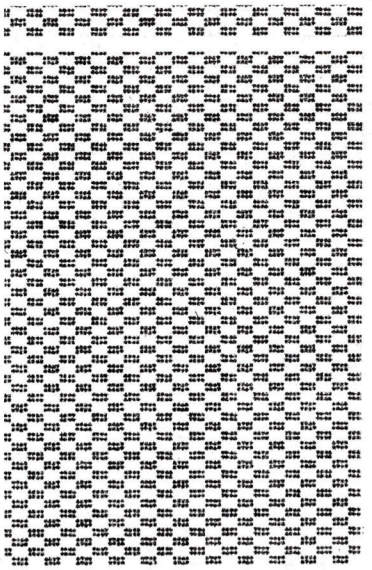
Punktmuster
Das Muster wurde erzeugt, indem abwechselnd Paare der Dezimalzahlen 195 und 60 an den Druckerkopf geschickt wurden. Die Tabelle verdeutlicht, wie diese Zahlen durch die Nadeln binär interpretiert werden. Durch den Papierzug überdruckt die folgende Zeile die Lücke von Nadel 1.

Nadelnummer	Nadelcode		
9	128	•	○
8	64	•	○
7	32	○	•
6	16	○	•
5	8	○	•
4	4	○	•
3	2	•	○
2	1	•	○
		195	60

• Nadel druckt
○ Nadel druckt nicht



Druckerkopf bewegt sich über das Papier



zulösen, lautete der Befehl CHR$(128). Wollen wir die Nadeln kombinieren, sind lediglich die Zahlen jeder Nadel hinzuzufügen.

In der Abbildung (oben) werden zwei Muster gezeigt: CHR$(195) und CHR$(60). Um die ersten vier Spalten des Linienmusters drucken zu können, wird eingegeben:

```
LPRINT CHR$(195);CHR$(195);CHR$(60);
    CHR$(60);
```

Nach vier Spalten wiederholt sich das Muster. Der Rest der Zeile wird also durch eine FOR . . . NEXT-Schleife gefüllt.

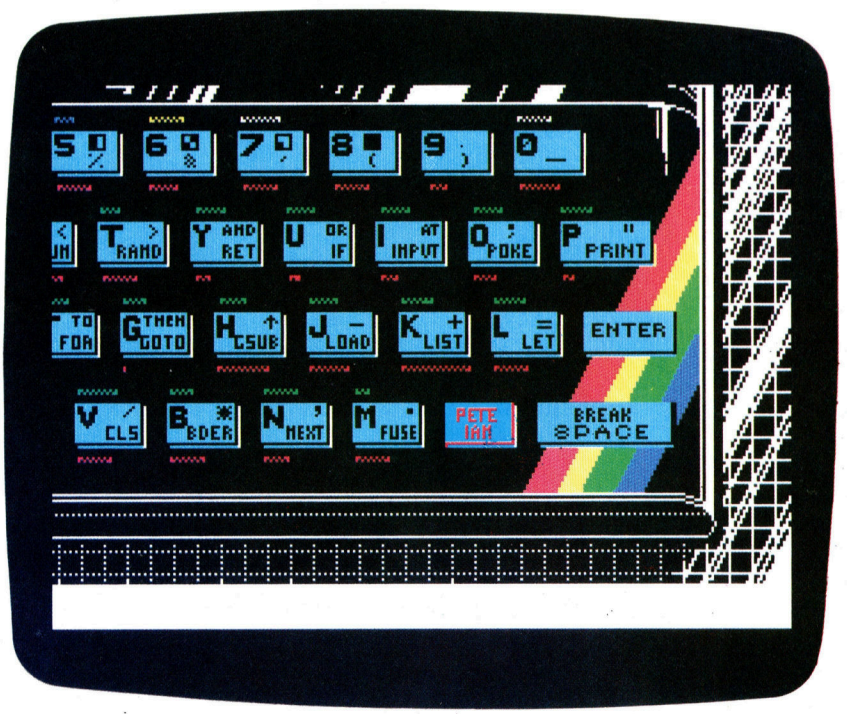
Diese Druckmethode, als „Bit Image Printing" bekannt, wird beim Epson FX-80 verwendet. Andere Drucker funktionieren ähnlich, weichen aber in den Details etwas ab. Das Erzeugen von Grafiken auf diese Weise ist recht arbeitsintensiv und nur zur Gestaltung von Mustern interessant. Eine weitaus bessere Möglichkeit zum Grafikdruck bietet das sogenannte „Screen Dump" (Bildschirm-Ausdruck): Mit Hilfe eines speziellen Programms wird auf Papier kopiert, was auf dem Bildschirm dargestellt wird.

Der Bildschirm-Ausdruck

Durch Längs- und Querabtastung der Bildschirmdarstellung überprüft das Programm, ob sich auf der abgefragten Position ein Punkt befindet. Ist dies der Fall, soll der Druckerkopf an der entsprechenden Stelle auf dem Papier drucken. Das Abtasten erfolgt durch Verwendung der POINT(x,y)-Funktion oder ähnlicher Befehle, über die die meisten Micros verfügen.

Wie aber wird ein „Screen Dump"-Programm mit farbiger Bildschirmdarstellung fertig? – Üblicherweise erfolgt die Problemlösung durch Verwendung unterschiedlicher Punktmuster für jede Farbe. Ein schwarzer Bildschirmpunkt kann durch Ausdruck von vier Punkten in Quadratform dargestellt werden. Ein roter Punkt durch zwei Punkte. Ein weißer Punkt käme ohne Farbe aus. Die POINT(x,y)-Funktion erzeugt unterschiedliche Zahlenwerte, abhängig von der Farbe des Punktes, und kann so immer eingesetzt werden.

„Screen Dump"-Programme werden gemeinhin als Subroutinen ans Ende eines Programms zur Bilderzeugung gehängt. Um das Bild auf den Drucker zu bekommen, drückt man die Taste „P", woraufhin das Programm in die Subroutine springt. In BASIC geschriebene „Screen Dump"-Programme sind recht langsam und benötigen etwa fünf Minuten zum Ausdruck eines kleinen Bildes. Maschinencode-Versionen sind erheblich schneller. Hat man alles jedoch erst einmal im Griff, kann die gedruckte Seite sehr attraktiv aussehen.

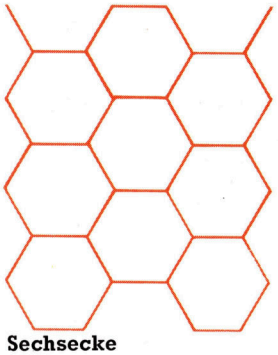


Dieses Bild der Spectrum-Tastatur wurde auf einem Farb-Tintenstrahldrucker erzeugt. Im Grunde handelt es sich dabei um einen Matrix-Drucker, in dem statt der Nadeln Tintenstrahldüsen verwendet werden.

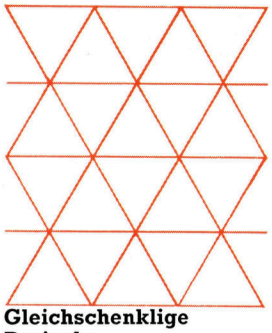
Mosaikmuster

Ein solches Muster entsteht durch Aneinanderfügen von gleichförmigen Elementen. In diesem letzten Teil des LOGO-Kurses wird gezeigt, wie man diese Muster mit Hilfe von verschiedenen Prozeduren erzeugt.

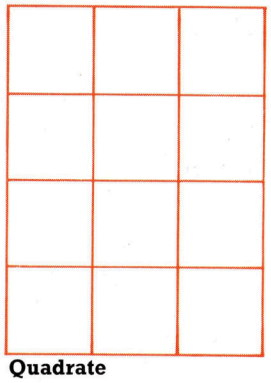
Drei Mosaikmuster



Sechsecke



Gleichschenklige Dreiecke



Quadrate

Aus den Polygonen Quadrat, gleichschenkliges Dreieck und Sechseck lassen sich Mosaikmuster zusammenstellen. Die restlichen Polygone eignen sich dafür jedoch nicht. Nehmen wir zum Beispiel das Fünfeck. Der Eckwinkel eines gleichmäßigen Fünfecks beträgt 108 Grad. Setzt man drei dieser Figuren an einem Punkt aneinander, ergibt sich ein Gesamtwinkel von 324 Grad – es bleibt also eine Lücke. Wird ein viertes Fünfeck zugefügt, beträgt die Summe der Winkel 432 Grad, wodurch eine Überlappung entsteht. Quadrate, gleichschenklige Dreiecke und Sechsecke dagegen lassen sich zu einem Mosaik zusammensetzen, da die Winkel an ihrer Senkrechten (90, 60 bzw. 120 Grad) exakt in 360 Grad aufgehen.

Um diese Mosaikmuster mit LOGO zu erzeugen, beginnen wir mit einem einfachen Motiv, wobei der Ausgangs- und Endpunkt der Turtle in der Mitte des Motivs liegen soll. Eine andere Prozedur steuert die Turtle danach lediglich in die Mitte des nächsten zu zeichnenden Motivs.

Die folgenden Prozeduren ergeben ein Mosaik aus Quadraten. Auf ähnliche Weise lassen sich Muster aus gleichschenkligen Dreiecken und Sechsecken erstellen.

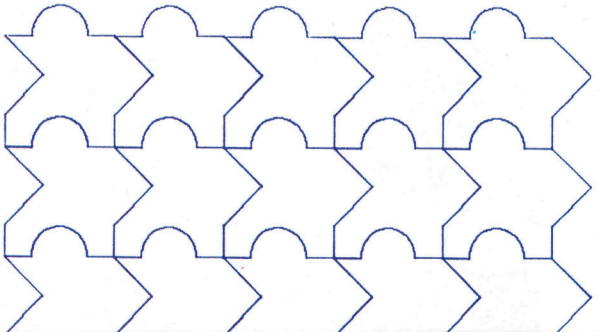
```
TO MOSAIK
  PENUP SETXY (—100) (90) PENDOWN
  REPEAT 5 [VERTIKAL SETXY (—100)
    (YCOR — 40)]
END

TO VERTIKAL
  REPEAT 5 [QU 40 SETX XCOR + 40]
END
TO QU :S
  PENUP LEFT 45
  FORWARD :S * (SQRT 2)/ 2
  RIGHT 135 PENDOWN
  FORWARD :S RIGHT 90 FORWARD :S
    RIGHT 90
  FORWARD :S RIGHT 90 FORWARD :S
    RIGHT 90
  PENUP LEFT 135
  BACK :S * (SQRT 2) / 2 RIGHT 45
END
```

Mosaikmuster lassen sich natürlich auch aus komplizierteren Motiven unterschiedlichster Art erstellen. Viele Zeichnungen von M. C. Escher zum Beispiel sind Variationen von Mosaikmustern.

Eine einfache Möglichkeit, interessantere Muster zu erstellen, bietet die Änderung von FORWARD in der QUADRAT-Prozedur. Dabei ist jedoch zu beachten, daß die Oberseite des einen und die Unterseite des nächsten Motivs genau aneinanderpassen müssen. Das gleiche gilt bei Modifikationen der rechten und linken Seiten.

Die Basisfigur wurde dahingehend geändert, daß sie zum Zeichnen der einzelnen Seiten Unterroutinen aufruft. Das Programm wird durch Eingabe von MOSAIK gestartet.

```
TO QU :S
  PENUP LEFT 45
  FORWARD :S * (SQRT 2) / 2
  RIGHT 135 PENDOWN
  SEITEA :S RIGHT 90 SEITEB :S RIGHT 90
  SEITEC :S RIGHT 90 SEITED :S RIGHT 90
  PENUP LEFT 135
  BACK :S * (SQRT 2) /2
  RIGHT 45
END
TO SEITEA :S
  FORWARD :S / 4 LEFT 90
  REPEAT 19 [FORWARD 2 * 3 1416 * :S /
    144 RIGHT 10]
  LEFT 100 FORWARD :S / 4
END
TO SEITEB :S
  LEFT 45 FORWARD :S / 2
  RIGHT 90 FORWARD :S / 2
  LEFT 45 FORWARD :S — :S * (SQRT 2) / 2
END
TO SEITEC :S
  FORWARD :S / 4 RIGHT 90
  REPEAT 19 [FORWARD 2 * 3 1416 * :S /
    144 LEFT 10]
  RIGHT 100 FORWARD :S / 4
END
TO SEITED :S
  FORWARD :S —  S * (SQRT 2) / 2 RIGHT 45
```



```
    FORWARD :S / 2 LEFT 90
    FORWARD :S / 2 RIGHT 45
  END
```

Die Zeichnungen von Escher sind natürlich viel interessanter als die hier entwickelten Muster. Unter anderem deshalb, weil sich die einzelnen Figuren im Gesamtbild des Mosaiks nach und nach verändern.

Harold Abelson und Andrea diSessa zeigen in ihrem Buch „Turtle-Geometrie" einen anderen Weg zur Erzeugung von Mosaikmustern auf. Ihre Überlegungen basieren auf folgendem: Zuerst wird ein Quadrat gezeichnet, das eine Figur enthält. Danach setzt bzw. „klebt" man dieses Quadrat mit drei weiteren zusammen und erhält so ein großeres Quadrat. Dieses Verfahren wird nun standig wiederholt.

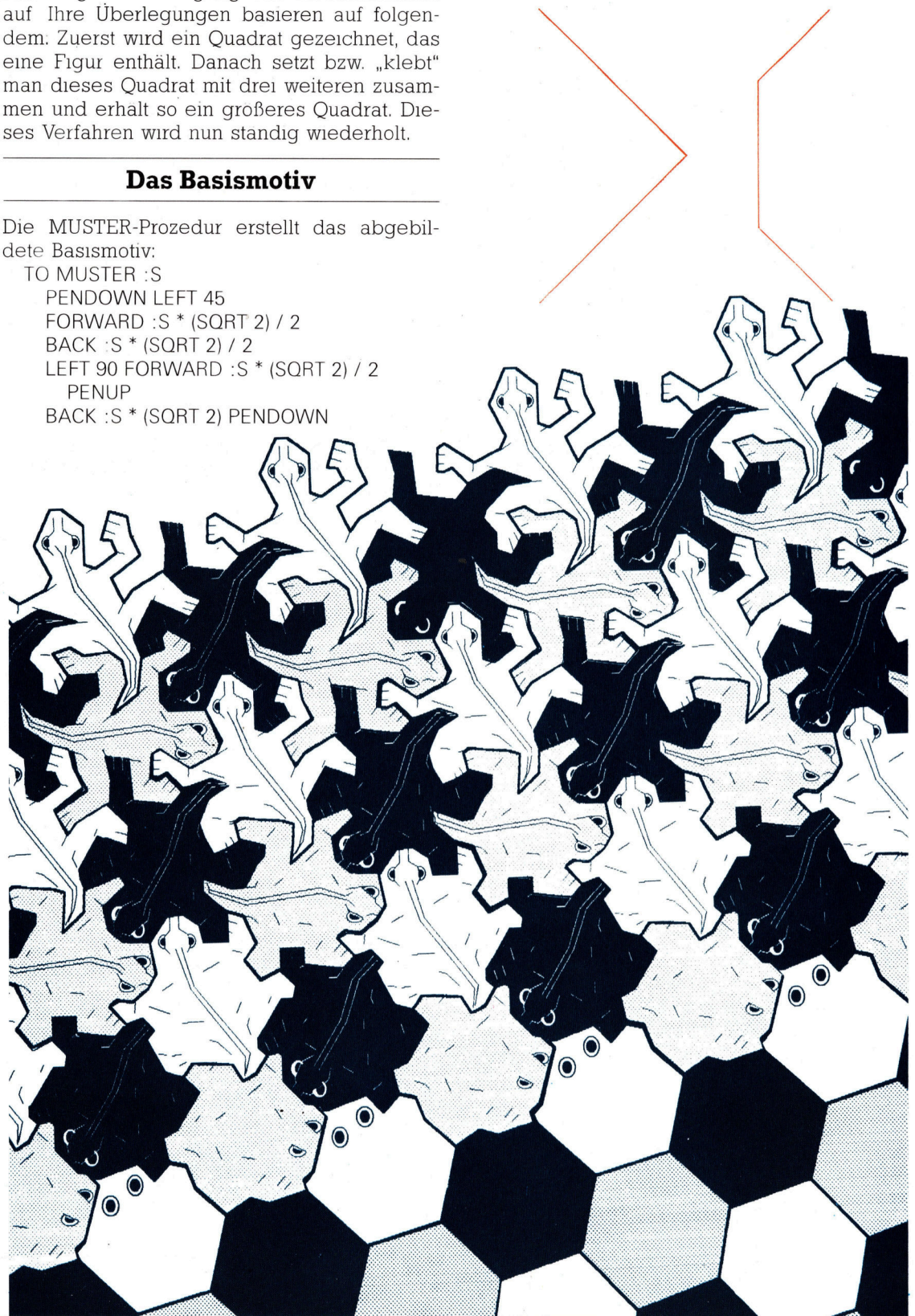
Das Basismotiv

Die MUSTER-Prozedur erstellt das abgebildete Basismotiv:

```
  TO MUSTER :S
    PENDOWN LEFT 45
    FORWARD :S * (SQRT 2) / 2
    BACK :S * (SQRT 2) / 2
    LEFT 90 FORWARD :S * (SQRT 2) / 2
      PENUP
    BACK :S * (SQRT 2) PENDOWN
    FORWARD S * (SQRT 2) / 4 LEFT 45
    FORWARD S / 2 LEFT 45
    FORWARD S * (SQRT 2) / 4 PENUP
    BACK S * (SQRT 2) / 2 LEFT 135
  END
```

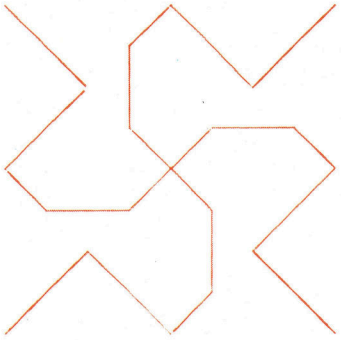
Mit der nachsten Prozedur und der Eingabe DRAW PENUP KLEBEN1 [MUSTER] 100 wird das folgende Muster gezeichnet:

```
  TO KLEBEN1 :PROZ :S
    QUADRAT.TEIL 0 :PROZ :S
```



Mosaikmuster setzen sich aus exakt ineinanderpassenden Figuren zusammen. „Metamorphosis II" von M. C. Escher (1898–1972) demonstriert, wie sich einfache Sechsecke modifizieren lassen, so daß wie in diesem Fall nach und nach aus diesen Figuren Eidechsen entstehen.

```
    QUADRAT.TEIL 0 :PROZ :S
    QUADRAT.TEIL 0 :PROZ :S
    QUADRAT.TEIL 0 :PROZ :S
END
```



QUADRAT.TEIL zeichnet eines der vier Quadrate, die zusammengefügt ein größeres ergeben. Dafür werden drei Eingaben benötigt: :A bestimmt die Zeichenrichtung, :PROZ repräsentiert den Namen der Prozedur, die das Motiv erstellt, und :S gibt die Seitenlänge an. Um zu verhindern, daß die Zeichnung aus dem Bildschirm „herausläuft", nimmt das Programm die Längeneingabe für das äußerste Quadrat und berechnet danach die Größe der innenliegenden:

```
TO QUADRAT.TEIL :A :PROZ :S
   FORWARD :S / 4 RIGHT 90
   FORWARD :S / 4 RIGHT 90 * :A
   RUN SENTENCE :PROZ :S / 2 LEFT 90 * :A
   BACK :S / 4 LEFT 90
   BACK :S / 4 RIGHT 90
END
```

Weitere Kombinationen sind:

```
TO KLEBEN2 :PROZ :S
   QUADRAT.TEIL 0 :PROZ :S
   QUADRAT.TEIL 1 :PROZ :S
   QUADRAT.TEIL 2 :PROZ :S
   QUADRAT.TEIL 3 :PROZ :S
END

TO KLEBEN3 :PROZ :S
   QUADRAT.TEIL 0 :PROZ :S
   QUADRAT.TEIL 2 :PROZ :S
   QUADRAT.TEIL 3 :PROZ :S
   QUADRAT.TEIL 1 :PROZ :S
END

TO KLEBEN4 :PROZ :S
   QUADRAT.TEIL 3 :PROZ :S
   QUADRAT.TEIL 2 :PROZ :S
   QUADRAT.TEIL 1 :PROZ :S
   QUADRAT.TEIL 0  :PROZ :S
END

TO KLEBEN5 :PROZ :S
   QUADRAT.TEIL 2 :PROZ :S
   QUADRAT.TEIL 1 :PROZ :S
   QUADRAT.TEIL 0 :PROZ :S
   QUADRAT.TEIL 3 :PROZ :S
END
```

Zahllose Kombinationen

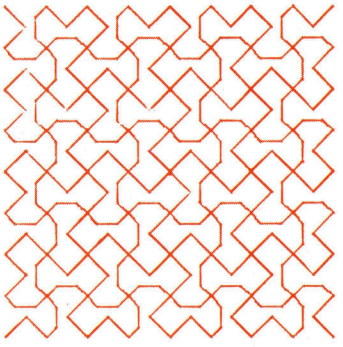
Die Prozedur MUSTER benötigt eine Eingabe, beispielsweise KLEBEN1 [MUSTER] 100. Da diese Eingabe wiederum als Befehl interpretiert wird, kann man mit [KLEBEN1[MUSTER]] auch eine andere KLEBEN-Prozedur aufrufen. Zum Beispiel:

```
KLEBEN2 [KLEBEN1 [MUSTER]] 100
```

Und weiter geht's mit

```
KLEBEN3 [KLEBEN2 [KLEBEN1 [MUSTER]]]
   100
```

Daraus ergeben sich schon 256x256x256 Kombinationen. Probieren Sie das auch mit den restlichen Prozeduren.

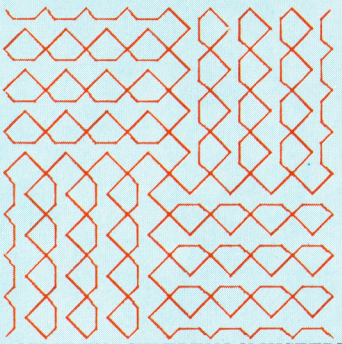


LOGO-Dialekte

In einigen LOGO-Versionen muß SETXY durch SETPOS ersetzt werden. Das Atari-LOGO verwendet SE statt SENTENCE.

Variationen

Probieren Sie folgende Kombinationen:



KLEBEN1 [KLEBEN4 [KLEBEN5 [MUSTER]]] 100

KLEBEN4 [KLEBEN3 [KLEBEN3 [MUSTER]]] 100

Fachwörter von A bis Z

Crash = Zusammenbruch
Bei einem Systemzusammenbruch verliert der Rechner sozusagen das Bewußtsein. In diesem Fall hilft nur Drücken des Reset-Knopfs oder Ziehen des Netzsteckers, womit im allgemeinen aber alle Programme und Daten im RAM verloren sind. Deshalb wird der Crash auch oft als „Fatal Error" (vernichtender Fehler) bezeichnet.

Für den Zusammenbruch eines Computersystems gibt es drei Hauptursachen. Häufig liegt die Ursache beim Netzteil. Vor allem die früheren Rechner nahmen schon geringe Spannungsschwankungen äußerst übel. Diese entstehen häufig durch einen lokalen Spannungsabfall bei Anschluß kräftiger Stromverbraucher nahe der Rechner-Steckdose. Bei den neueren Netzteilen ist die Stromversorgung besser geregelt, aber wenn auf Ihren Leitungen „Spikes" (hochfrequente Spannungsspitzen) herumgeistern oder wenn längeranhaltende Spannungseinbrüche auftreten (dabei flackert das Licht), dann sollten Sie sich den Kauf eines externen Netzspannungs-Stabilisators überlegen.

Eine andere Form von Crash wird dadurch ausgelöst, daß der Rechner sich in einer endlosen Programmschleife festläuft. In BASIC hilft dann einfaches Drücken der „Break"- oder „Stop"-Taste, aber Maschinencode-Programme sind nur bei wenigen Rechnern über die Tastatur abzubrechen, so daß eine JMP-Schleife ohne Ausgangsabfrage zwangsläufig zum Crash führt.

Bei den meisten Microprozessoren kommt es außerdem zum Crash, wenn im Befehlsregister ein Opcode auftaucht, der im Befehlsvorrat nicht vorgesehen ist. Dafür kann ein Assemblerfehler verantwortlich sein – vielleicht eine falsche JMP-Anweisung, derzufolge Daten als Befehle interpretiert werden.

Bildnachweise
841, 849, 867: Ian McKinnell
842, 846, 847, 853, 854, 855: Chris Stevens
850, 851, 852, 862: Kevin Jones
856, 857, 866: Liz Dixon
859: Digital Research Inc., Ian McKinnell
860: Digital Research Inc.
864: Simon Daton
865: Steve Cross, Ian McKinnell, Dimension Graphics

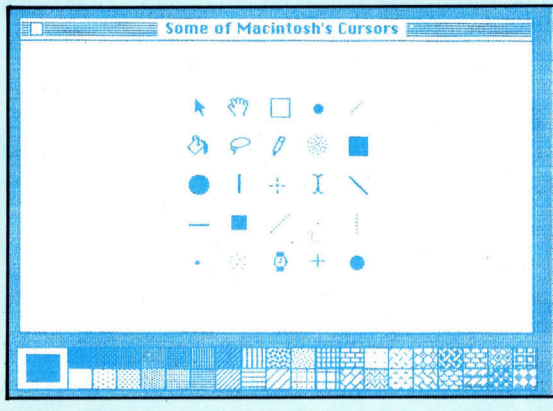
Hier werden einzelne Fachausdrücke eingehend behandelt. Da bei der Kommunikation mit dem Computer meist die englische Sprache verwendet wird, werden hier zunächst die englischen Begriffe genannt, dann die deutsche Übersetzung. In den Gesamtindex werden sowohl deutsche als auch englische Stichwörter aufgenommen,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

Cross-Assembler = Cross-Assembler
Ein Cross-Assembler ist ein Übersetzungsprogramm zur Umwandlung vom Source Code in den Object Code. Cross-Assembler erlauben den Entwurf von Programmen für unterschiedliche Microcomputer auf ein und demselben Entwicklungssystem, das mit viel mehr Komfort und Speicherplatz ausgestattet sein kann als ein Heimcomputer, natürlich auch schneller ist und über eine größere Auswahl an Editier- und Dienstprogrammen verfügt. Der Einsatz ist dann sinnvoll, wenn Sie zum Beispiel Ihren Lebensunterhalt damit verdienen wollen, Spiel- oder Geschäftsprogramme zu entwerfen und selbständig zu vertreiben.

Das Entwicklungssystem kann sogar einen anderen Prozessor haben als der Rechner, für den die Programme entwickelt werden. Der Cross-Assembler wird mit den unterschiedlichsten Befehlsstrukturen und auch mit Adressierungsproblemen fertig. Ein Cross-Compiler arbeitet ähnlich, dient jedoch zur Übersetzung von Programmen, die in einer Hochsprache und nicht in Assembler geschrieben sind.

Current Loop = Stromschleife
Die gängigste Norm für den seriellen Datenverkehr ist der RS232-Standard, der die Codierung der Datenbits (mit Start- und Stop-Bits) und das „Handshaking" für die Steuerung der Übertragung festlegt.

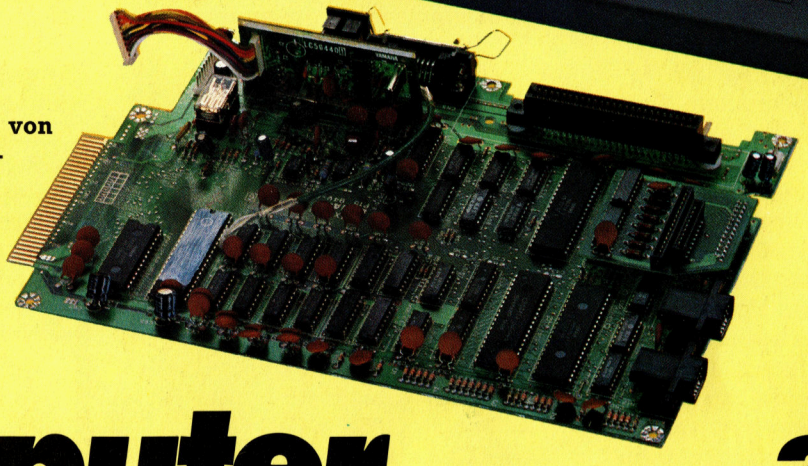
Der US-Norm RS232 entspricht die internationale CCITT-Norm V.24. Dabei werden die Datenbits durch einen Spannungspegel von -3 bis -15 V für die logische Eins und einen Pegel von +3 bis +15 V für die logische Null dargestellt. Neben dieser „Spannungsschnittstelle" gibt es noch die 20 mA-Stromschnittstelle (TTY), die ursprünglich zur Ansteuerung von Fernschreibern verwendet wurde. Dabei bilden die Peripheriegeräte mit Hin- und Rückleitung eine geschlossene Stromschleife (Current Loop), in der eine 1 durch einen Strom von 20 mA und eine 0 durch eine Stromstärke von Null Milliampere realisiert wird.



Cursor = Cursor
Der Cursor ist ein Bildschirmsymbol, das die nächste Zeichenposition andeutet. Das Wort (ursprünglich aus dem Lateinischen: cursor = der Läufer) bezeichnet im Englischen auch den Schieber des Rechenstabs oder den Stellenanzeiger bei mechanischen Addiermaschinen.

Traditionell wird als Cursor bei Computern ein blinkendes Unterstreichungszeichen oder ein Quadrat verwendet, manchmal auch einfach ein blinkender Buchstabe. Dagegen setzen Rechner wie der Lisa von Apple oder der Macintosh als Cursor nach Bedarf verschiedene Piktogramme ein: Eine Armbanduhr zeigt Ihnen beispielsweise, daß Sie warten müssen, und eine Hand deutet darauf hin, daß eine Objektbewegung erwartet wird.

+ Vorscha[illegible]hau +++ Vorschau +++ Vorschau +++ Vorschau +



Der MSX-Rechner CX5M von Yamaha ist ein Heimcomputer, der speziell zum Musizieren entwickelt wurde. So verfügt er z. B. über eine besondere Buchse zum Anschluß von Klaviertastaturen.

computer kurs

Heft 32

Perfekte Zukunft

Noch ist die reale Roboterwelt von „denkenden mechanischen Wesen" weit entfernt. Trotzdem erwarten wir in der Zukunft ganz bestimmte Fähigkeiten der Roboter.

Bildwiedergabe

Zwar kann man jeden Heimcomputer an ein Fernsehgerät anschließen, doch liefern Monitore ein besseres Bild. Kombinations-Geräte sind eine Alternative.

Schaltkasten

Im Selbstbau-Kurs geht es um ein Interface, das die Computersteuerung von Lampen und Motoren ermöglicht.

Schnelle Post

Wenn man ein Textverarbeitungsprogramm auf dem Heimcomputer einsetzt, läßt sich bei Serienbriefen Zeit sparen.

Pascal

Im nächsten Heft beginnt eine neue Serie: Die Sprache PASCAL, die als „streng diszipliniert" gilt.