Einsteigen - Verstehen - Beherrschen

DM 3,80 öS 30 sfr 3,80

computer kurs

Heft 49

Servomotoren

Prozessor 6809

Der Atari 520 ST

Japans MSX-Standard

Ein wöchentliches Sammelwerk

computer kurs

Heft 49

Inhalt

WIE SIE JEDE WOCHE IHR HEFT BEKOMMEN

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

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

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

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

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

Abonnement

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

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

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

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

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

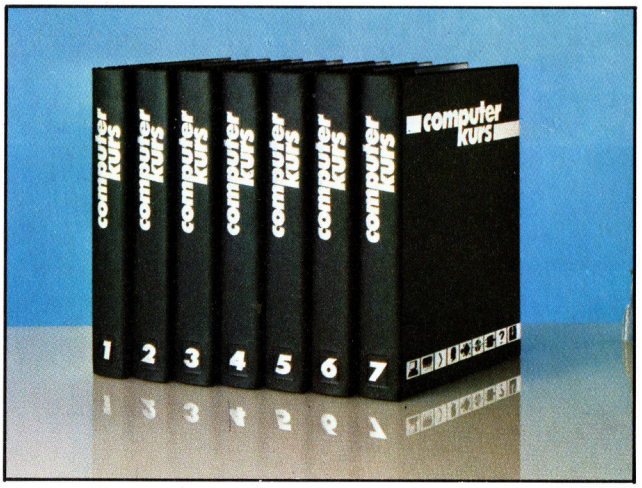
Schweiz: Der Sammelordner kostet sfr 15. Bitte wenden Sie sich an Ihren Kiosk; dort werden Sie jederzeit die gewünschten Exemplare erhalten.

INHALTSVERZEICHNIS

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

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

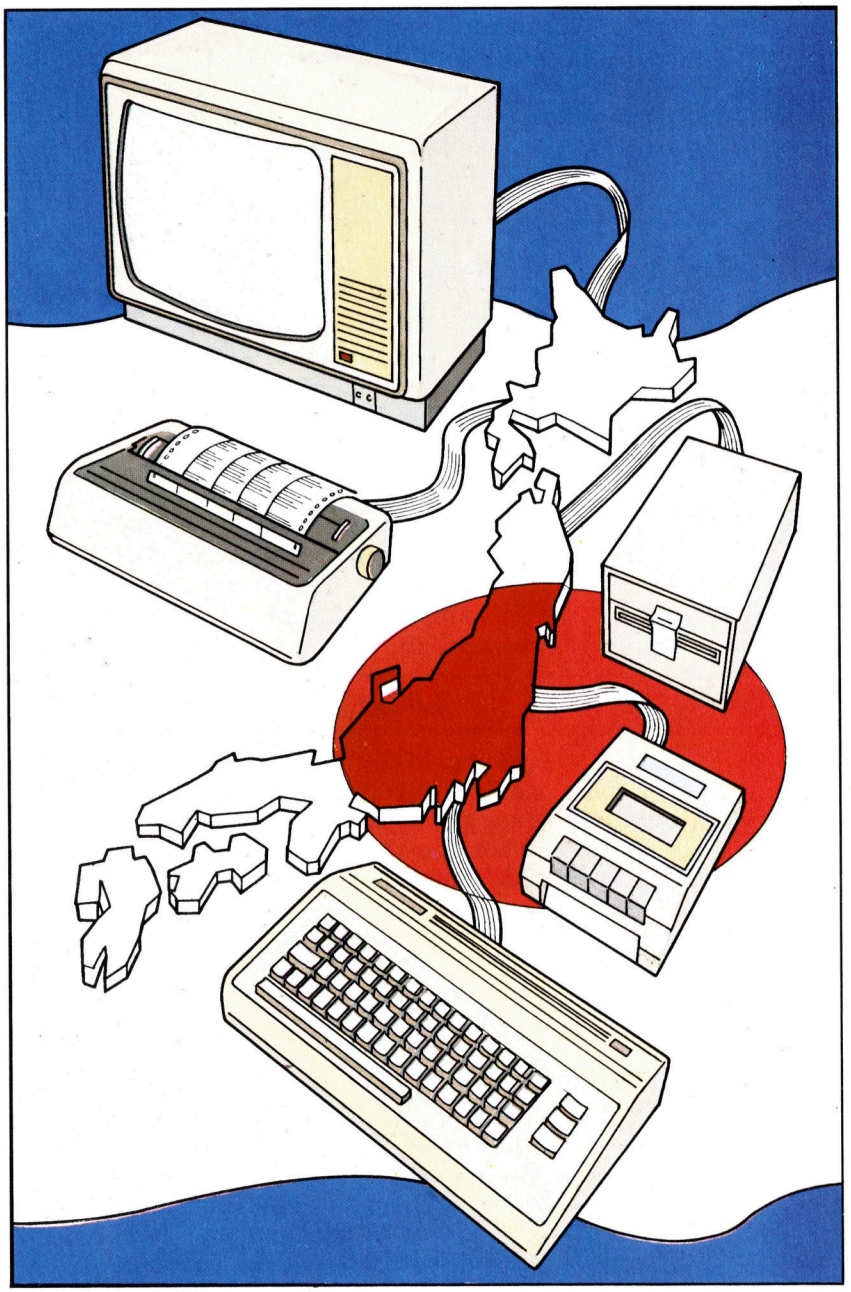
Vertrieb: Marshall Cavendish International Ltd., Heidenkampsweg 74, 2000 Hamburg 1



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

Normgerecht

Wer sich mit der Softwareentwicklung für Heimcomputer befaßt, steht ständig vor dem Problem der Kompatibilität. Mit dem MSX-Standard versucht eine Herstellergruppe, die generelle Austauschbarkeit von Hard- und Software zwischen verschiedenen Computern zu erreichen.



Um zu verstehen, weshalb die verschiedenen Computermodelle so starke Eigenheiten aufweisen, muß man sich die historische Entwicklung der Branche vergegenwärtigen. Die ersten Microprozessoren, die den Markt nachhaltig beeinflußten, waren der Intel 8080 und der Motorola 6800. Die Befehlssätze dieser Chips waren von Anfang an eindeutig festgelegt. Die Kompatibilitätsprobleme erwuchsen eigentlich erst aus der weitgehenden Freizügigkeit bei der Zuordnung der übrigen Systemkomponenten. Zum Beispiel erfordert die Ausgabe eines Datenbyte seitens der CPU bei allen Rechnern, die mit dem gleichen Prozessor arbeiten, denselben Befehl. Für die Festlegung der Adressen gibt es jedoch Hunderte oder Tausende von Möglichkeiten: Wenn für den Paralleldruckerausgang beim Hersteller A vielleicht gerade die Adresse 255 definiert wurde, so lag er bei der Firma B dagegen beispielsweise auf Adresse 254.

Probleme durch „Verbesserungen"

Das war schon lästig genug, aber das eigentliche Chaos brachten erst die vielfältigen Spezialchips wie Video-Steuerbausteine für die Bildschirmgrafik und Tongeneratoren für die Klangerzeugung. Ein Programm, das die individuellen Möglichkeiten solcher Schaltkreise ausschöpfte, lief nicht auf einem anderen Rechner.

Gleich zu Beginn der Computer-Ära gab es eine ganze Reihe etwa gleichstarker kleiner Firmen, die alle ihren eigenen „Standard" setzen wollten. Ihre Produkte unterschieden sich nicht nur äußerlich und im Schaltungskonzept, sondern auch in der Programmiersprache – von Anfang an gab es kein einheitliches BASIC. Schuld daran war auch, daß BASIC zumindest in den siebziger Jahren von den Profis kaum ernstgenommen und zur Sprache für Computer-Neulinge erklärt wurde.

Der Übergang von den siebziger zu den achtziger Jahren brachte einen gewaltigen Entwicklungsschub. Vorreiter wie Apple begannen, diverse Raffinessen, wie hochgezüchtete Farbgrafik, in ihre Rechner zu integrieren. Um solche Innovationen nutzbar zu machen, mußten die Hersteller ihre eigenen BASIC-Versionen entwickeln, was zu einer großen Anzahl von Dialekten führte.

Die heutige Sprachvielfalt ist für den Benutzer keinesfalls erfreulich, sondern genau wie für Software-Häuser eher frustrierend.

Wer als Rechnerhersteller vor der Einführung eines neuen Modells steht, weiß genau, daß er praktisch nicht mit Softwareunterstützung rechnen kann, bevor ein ausreichender Benutzerkreis existiert. Andererseits besteht

Der MSX-Standard soll Nippons Computerindustrie den Weg von den japanischen Inseln zum Weltmarkt freimachen. Wenn das Konzept Erfolg hat, könnte Japan auf dem Rechnersektor ähnlich marktbeherrschend werden wie im HiFi- und Kamera-Geschäft.

ohne großes Softwareangebot nur ein sehr begrenzter Kaufanreiz und damit die Gefahr, daß sich die Entwicklungskosten für das neue Gerät als Fehlinvestition erweisen.

Firmen, die von der Softwareerstellung leben, können ein für eine bestimmte Maschine geschriebenes Programm logischerweise bestenfalls in einer Auflage verkaufen, die der Anzahl der Rechnerbesitzer entspricht. Angenommen, ein Unternehmen hat gerade das neue Abenteuerspiel „Das Verlies von Rattenschreck" für den Spectrum fertiggestellt – ausgestattet mit Grottenschraten, Aaskriechern und andern frevelhaften Wesen, die dem Unachtsamen den Garaus machen wollen. Die Leute von der Marktforschung garantieren einen reißenden Absatz, wenn die Cassette für knapp zwanzig Mark angeboten wird. Leider scheint das Spiel aber doch nicht so zugkräftig, daß sich eine rentable Stückzahl allein mit den Spectrumbesitzern erreichen ließe. Der Rattenschreck müßte also für Commodore-, Atari- und Schneider-Computer umgeschrieben werden. Das würde den Ladenpreis jedoch hochtreiben, und damit wären kostendeckende Stückzahlen wieder nicht mehr zu erreichen. In einem ähnlichen Dilemma stecken viele Softwareanbieter.

Lösung der Probleme

Die Firma ASCII/Microsoft hat sich die Lösung des Kompatibilitätsproblems zum Ziel gesetzt. Beteiligt sind daran die American Microsoft Corporation und das japanische Verlagshaus ASCII, das für eine Reihe populärer Zeitschriften verantwortlich zeichnet. Nebenher vertreibt ASCII auch kommerzielle Software, und als Microsoft in den schwierigen japanischen Markt eindringen wollte, bot sich ASCII als Partner für dieses Unterfangen an. Microsoft hatte die fachliche Erfahrung, und ASCII kannte den Markt.

Die Firma Microsoft ist wesentlich durch ihr „MBASIC" bekannt geworden, das als eine Art Standard gilt und weltweit von Rechnerherstellern übernommen wurde. Trotzdem kann man nicht davon ausgehen, daß MBASIC-Programme auf jedem Rechner mit MBASIC-Interpreter laufen. Wenn der Rechner spezifische Extras aufzuweisen hat, fehlt nämlich häufig die Hardware-Kompatibilität.

Das Microsoft-BASIC wurde über die ASCII/Microsoft-Vertretungen erfolgreich an eine Reihe japanischer Computerfirmen verkauft, ohne daß damit die Frage der Kompatibilität vor allem auf der Hardwareseite gelöst gewesen wäre. Deshalb entschloß sich ASCII/Microsoft, in Zusammenarbeit mit führenden japanischen Rechnerherstellern umfassende Spezifikationen zu definieren, – in der Hoffnung auf internationale Anerkennung. Das Ergebnis ist der „MSX-Standard", der neben dem BASIC-Dialekt auch eine bestimmte Hardware-Grundausstattung vorschreibt, ausgehend vom Z80 als Prozessor und einigen Zusatzbausteinen.

Das MSX-BASIC (Microsoft Super Extended) ist dem MBASIC von Microsoft eng verwandt, hat aber etliche Verbesserungen aufzuweisen, die den heutigen Grafik- und Sound-Ansprüchen Rechnung tragen. Unter anderem sind folgende Sonderbefehle implementiert: SCREEN legt den Bildschirm-Modus fest, LOCATE lokalisiert Zeichen auf dem Schirm, COLOUR definiert eine der 16 Vorder- und Hintergrundfarben, PUT SPRITE positioniert ein Sprite auf dem Bildschirm, CIRCLE erzeugt Kreise und Ellipsen, LINE zieht Linien zwischen vorgegebenen Punktkoordinaten, mit PAINT lassen sich Figuren beliebig einfärben und mit KEY fünf Funktionstasten selbst definieren. Weitere Befehle betreffen die Datenübergabe an das Video-RAM (VPOKE), die Programmierung des Sound-Chips (SOUND) und die Steuerung des Cassettenlaufwerks (MOTOR).

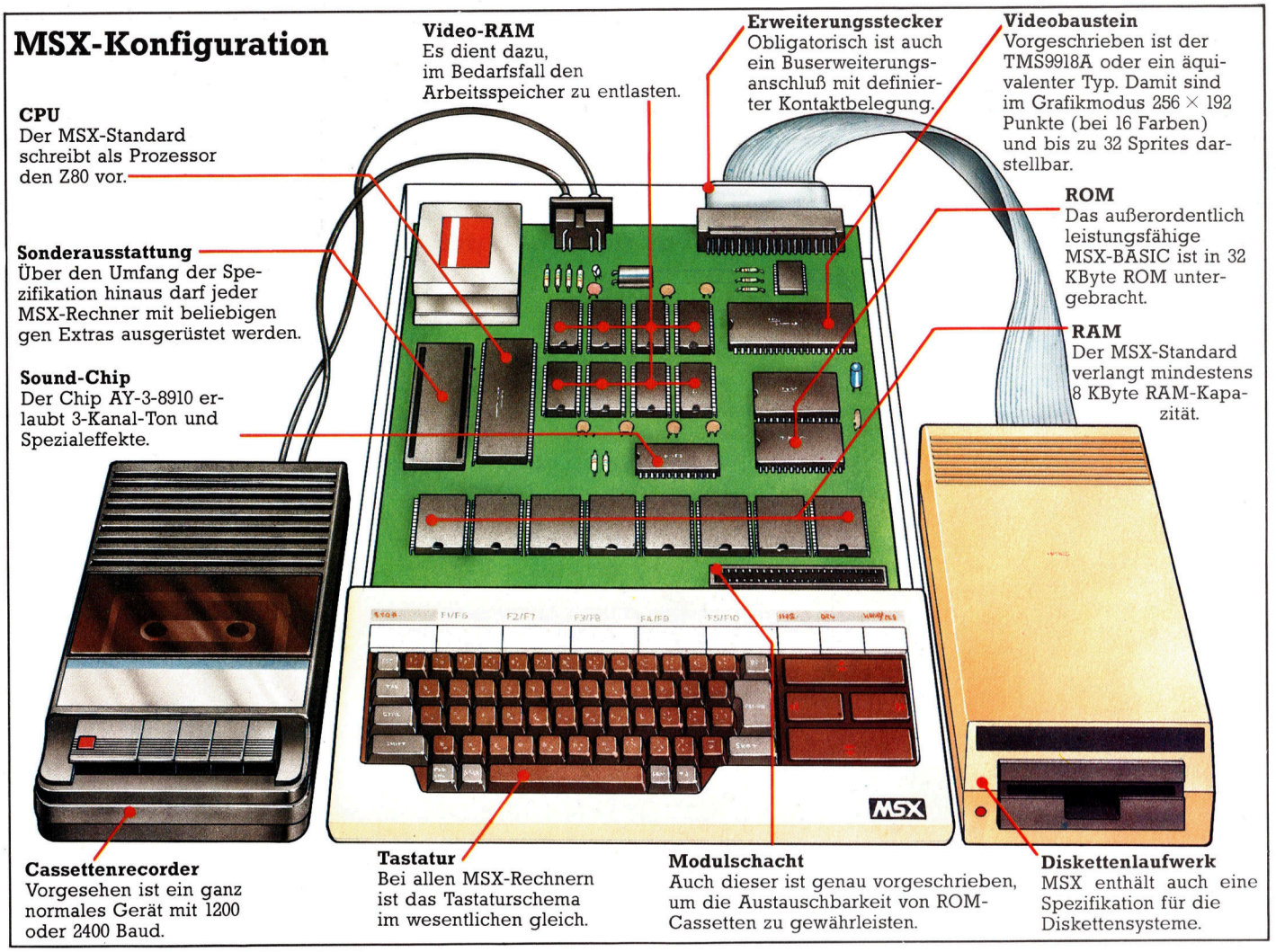
Der MSX-Standard betrifft aber nicht nur die Software. Als CPU wird ein Z80 mit 3,58 MHz-Takt spezifiziert. Mindestens 32 KByte ROM-Kapazität sind für die Unterbringung der MSX-Systemprogramme erforderlich, außerdem müssen wenigstens acht KByte RAM verfügbar sein. Nach oben sind ROM- und RAM-Kapazität nicht begrenzt. Ein MSX-Rechner muß mit dem TMS9918A von Texas Instruments oder einem gleichwertigen Video-Steuerbaustein ausgerüstet sein, und als Tongenerator ist der dreikanalige AY-3-8910 von General Instruments vorgesehen. Der Bildschirm muß für 24 Zeilen mit 32 oder 40 Zeichen eingerichtet sein. Eine 80-Zeichen-Darstellung ist derzeit nicht möglich. Im Grafikmodus beträgt die Auflösung 256 × 192 Punkte.

Wegen der Bedeutung von Cassetten beim Heimcomputer ist auch dafür eine Schnittstelle spezifiziert und zwar mit FSK-Codierung und einer Datenrate von 1200 oder 2400 Bit/s. Für die Tastatur ist mit Rücksicht auf die Japaner nicht nur die übliche Belegung (mit zusätzlichen Funktionstasten) vorgesehen, sondern auch das japanische Normalschrift-Alphabet „Katakana", das Kursiv-Alphabet „Hiragana" sowie standardisierte Grafikzeichen und auf Wunsch „Kanji" – das sind die chinesischen Schriftzeichen.

Für ROM-gespeicherte Software ist ein genormter Modulschacht vorgesehen, außerdem sind eine 50polige Bus-Schnittstelle für Ein- und Ausgabe und zwei Joystick-Ports im System spezifiziert.

Als Disketten-Betriebssystem wird das MSX-DOS verwendet, das in seiner Funktion dem MS-DOS entspricht und MS-DOS-Dateien lesen kann. Es soll mit dem verbreiteten CP/M2.2-DOS kompatibel sein.

Die Einhaltung der Spezifikationen garantiert, daß die für einen MSX-Rechner geschriebenen und auf Diskette oder Cassette gespeicherten Programme auch auf jedem anderen



Um die Kompatibilität von Hard- und Software durchgängig zu gewährleisten, ist für alle MSX-Rechner ein einheitliches Konzept vorgeschrieben. Über die in der Abbildung dargestellten Mindestanforderungen hinaus kann aber jeder Produzent seine Maschine mit weiteren Extras ausstatten, um sich einen Vorsprung gegenüber der Konkurrenz zu sichern.

MSX-Rechner laufen und seine Sound- und Grafik-Möglichkeiten voll nutzen. Die vielfältigen Vorteile für Hersteller wie Benutzer liegen auf der Hand.

Licht und Schatten

Das MSX-System hat aber auch seine Nachteile. Einmal führt jede Standardisierung dazu, daß von Neuerungen kein Gebrauch gemacht werden kann. Wenn zum Beispiel ein weiterentwickelter Videobaustein herauskommt, haben die MSX-Anhänger das Nachsehen, und für Konkurrenten mit einem Softwareangebot, das die Möglichkeiten des neuen Chips voll wahrnimmt, ergibt sich weltweit ein gewaltiger Marktvorteil.

Zum anderen ist die Lebenserwartung der 8-Bit-Prozessoren begrenzt, wenn auch der Z80 immer noch erfolgreich ist. Mit einer 8-Bit-CPU sind zwangsläufig nicht mehr als 64 KByte Arbeitsspeicher direkt adressierbar, und größere Zahlen als 256 können nicht parallel verarbeitet werden. Unter diesem Aspekt könnte man im MSX-Standard einen letzten Versuch sehen, um den Z80 am Leben zu halten – allerdings mit entsprechend geringen Aussichten auf langfristigen Erfolg.

Sie können den MSX-Standard aber auch als Fingerzeig für die Zukunft werten. Es ist kaum vorstellbar, daß der Microcomputermarkt in fünf oder zehn Jahren noch von den 8-Bit-Prozessoren bestimmt wird. Der MSX-Standard demonstriert der ganzen Branche jetzt nachhaltig, daß es für Absprachen über eine Vereinheitlichung auf dem Computersektor eigentlich nie zu früh sein kann. Auf die Dauer besteht der Effekt wahrscheinlich darin, daß die anderen Hersteller von der Unerläßlichkeit einer Standardisierung überzeugt werden. Bei den 16-Bit-Rechnern hat sich inzwischen wieder einmal der Spruch „Macht ist Gesetz" bewahrheitet, denn der IBM ist de facto zum Standard geworden. Hat die MSX-Norm bei den 8-Bit-Heimcomputern eine ähnliche Chance? Bisher haben sich ihr vor allem japanische Hersteller angeschlossen, darunter Yamaha, JVC, Hitachi, Sony, Sanyo, National, Pioneer, Canon, Fujitsu, Mitsubishi und Yashica, außerdem Spectravideo (USA), Daewoo (Korea) und auch Philips. Nur der langfristige Markterfolg kann zeigen, ob dies der richtige Weg ist oder ob unternehmerische Einzelgänger wie Sir Clive Sinclair mit ihren individuellen Innovationen letztlich mehr ausrichten. Der bewegliche Heimcomputermarkt ist allemal für Überraschungen gut.

Der Faktor Mensch

Ein wichtiger Aspekt beim Programmaufbau ist die „Schnittstelle Mensch-Maschine", also der Teil des Programms, das die Datenübertragung vom Anwender zur Maschine und von der Maschine zum Anwender steuert. In diesem Artikel untersuchen wir die Faktoren, die dabei berücksichtigt werden müssen.

Viele Jahre lang war die Computerprogrammierung ein beinahe „mystisches" Gebiet, das nur von Profis verstanden wurde. Vor dem Auftauchen des Microcomputers mit seiner Schreibmaschinentastatur wurden Programme oft byteweise über Schalter an der Vorderseite der Computer, per Lochstreifen oder über Fernschreiberkonsolen eingegeben.

Im Vergleich dazu wird der Anwender heute regelrecht verwöhnt. Die Computerbesitzer brauchen sich nicht mehr mit dem Maschinencode abzugeben, und der Begriff „benutzerfreundlich" besagt, daß sich die Micros unabhängig von Vorkenntnissen einsetzen und programmieren lassen. 1982 veröffentlichte das Alvey-Komitee in dem Bericht „Ein Programm für die fortgeschrittene Informationstechnologie", daß die Schnittstelle Mensch-Maschine als einer der vier Haupt-Forschungsbereiche angesehen werde. Die anderen drei Gebiete sind die Konstruktion von Software, integrierte VLSI-Schaltungen (Very Large Scale Integration) und Systeme zur Wissensverwaltung.

In fast jeder Anwendung spielt die Wechselwirkung bzw. der Datenaustausch zwischen Computer und Anwender eine wesentliche Rolle. Dieser „Dialog" wird über die Ein- und Ausgabekomponenten des Computers (E/A) geleitet, wobei die Tastatur die wichtigste Eingabequelle darstellt und der Bildschirm das wichtigste Ausgabeinstrument. Eingaben sind aber auch über Joysticks, Mäuse, Touch-Screens und andere Geräte möglich, während sich für die Ausgabe auch Drucker, Tongeneratoren (Sprache) und sogar Roboter einsetzen lassen.

Doch außer den eingesetzten E/A-Geräten bestimmt hauptsächlich die Software den Dialog zwischen Mensch und Computer. So steuert beispielsweise das Betriebssystem viele Funktionen von Tastatur und Bildschirm. Es bestimmt die Wiederholgeschwindigkeit einer gedrückten Taste, den Zeitraum zwischen den einzelnen Wiederholungen und bewahrt auch die Zeichen auf, die schneller eingegeben wurden, als sie der Bildschirm anzeigen kann. Einige Betriebssysteme können nur ein Zeichen zwischenspeichern, die meisten Heimgeräte haben aber Buffer für zehn oder mehr Zeichen.

Zeichenbuffer können allerdings auch Probleme verursachen. Erfahrene Anwender, die mit menügesteuerten Systemen arbeiten, wissen beispielsweise, daß sie aus dem Hauptmenü die 2 wählen müssen, aus dem nächsten Menü 5, dann 3,4,6 etc. Sie sind mit dem System vertraut und geben diese Ziffern mit hoher Geschwindigkeit ein. Bei einem Buffer für zehn Zeichen entstehen keine Probleme, da der Computer sich alle Tasten in der richtigen Reihenfolge „merkt". Mit einem Ein-Zeichen-Buffer kann die Darstellung des zweiten Menüs jedoch mehr Zeit in Anspruch nehmen, als die gesamte Eingabe dauert. Da nur das letzte Zeichen im Buffer bleibt, wird statt der 5 und der 3 etc. nur die Eingabe 6 ausgeführt (als letztes vom Buffer gespeichert), und das Programm bricht ab.

Aber auch große Buffer sind nicht unproblematisch. Bei Menüprogrammen, die erst nach einiger Zeit auf Eingaben reagieren, kann der Anwender den Eindruck erhalten, daß überhaupt nichts passiert. Er versucht die Eingabe

CP/M

nochmals und drückt so lange auf die Tasten, bis etwas passiert. Wenn das Programm dann versucht, die im Buffer gespeicherten Anweisungen auszuführen, können unerwartete Ergebnisse entstehen.

Das „Räumen" des Arbeitsspeichers von nicht mehr benötigten Variablen stellt eine weitere Problemquelle dar. Da das Programm bei dieser Arbeit lange Zeit zu „hängen" scheint, sind viele Anwender versucht, diesen Zustand irgendwie zu beenden. Besonders große Pro-

gramme mit zahlreichen String-Variablen können für die Speicherbereinigung viel Zeit benötigen. In einigen BASIC-Versionen läßt sich in festen Zeitabständen eine Bereinigung durchführen. Dabei sollte eine Meldung anzeigen, daß der Computer beschäftigt ist.

Auch die Art, wie Programmiersprachen Ein- und Ausgaben handhaben, beeinflußt die Schnittstelle Mensch-Maschine. BASIC-Versionen mit eingebauter Cursorsteuerung und Grafikbefehlen gestalten Bildschirmeingaben einfacher. Überhaupt ist BASIC gut mit Ein- und Ausgabebefehlen ausgerüstet. So genügen für einfache Programme oft INPUT und PRINT, während die echte Eingabesteuerung (die beispielsweise auch Eingabefehler abfangen soll) mit GET$, INKEY$, INPUT$() und ähnlichen Befehlen programmiert werden kann. Weiterhin ist PRINT USING ein außerordentlich flexibler Befehl für die Ausgabeformatierung. Er kann Dezimalpunkte genau untereinandersetzen und Textspalten justieren.

Der Anwender ist unberechenbar

In jedem System Mensch-Maschine ist der Anwender das am wenigsten berechenbare Element. Wie die anderen Komponenten folgt auch seine Arbeit jedoch gewissen Mustern, die bei der Konstruktion der „Schnittstelle" daher unbedingt berücksichtigt werden müssen.

Micronet 800

Eine der grundlegenden Eigenschaften von Mensch und Computer ist die Fähigkeit, „Informationen verarbeiten zu können". Bei Menschen ist jedoch die Informationsmenge, die sie in ihren „Arbeitsspeicher" aufnehmen können, begrenzt. Man geht davon aus, daß von den meisten Informationsarten etwa sieben verschiedenen „Daten" gleichzeitig im Gedächtnis gespeichert werden. Die Menge der erinnerten Informationen hängt oft von der Struktur und Bedeutung dieser Merkmale ab. Sind die Informationen aus zufällig auftretenden Zeichen zusammengesetzt, dann besteht jede Erinnerung nur aus einzelnen Zeichen. Sind sie jedoch nicht zufällig, sondern bilden beispielsweise bekannte Nachnamen, dann kann jedes „Datum" durchaus einen ganzen Namen enthalten. Eine Verbesserung der Informationsstruktur kann dabei die Erinnerungsfähigkeit des Anwenders vergrößern.

Der Mensch kann auf mehrere Arten unterstützt werden, Informationen für den Einsatz eines Computers zu strukturieren. Eine dieser Methoden stellt eine Beziehung zwischen Daten und vertrauten und verstandenen Strukturen her. Der „Schreibtisch" des Apple Lisa arbeitet nach diesem Schema, und auch Kalkulationssysteme werden oft wie Bücher mit Seiten, Indizes etc. angelegt. Eine andere Methode macht den Anwender mit neuen Strukturen vertraut: Mit wiederholten Beispielen und ausführlichen Erklärungen bringen dabei die Programme selbst dem Anwender bei, wie er seine Informationen strukturieren soll. Dieses Training erfordert aber viel Zeit und Aufwand. Ausführliche Informationen, „Hilfsmenüs" und „Merker" können zwar eine direkte Hilfe bieten, sind aber nicht immer leicht zu bedienen.

Schließlich fördert schon die Anordnung von Informationen das Verständnis. Farbliche Gestaltung und ein spezieller Bildschirmaufbau helfen beim Auffinden der gewünschten Daten. Beispiele dafür sind die Farbcodes von Bildschirmtext, Prestel etc. Auf einer typischen Seite sind die „Überschrift" und die „Fußzeile" in Blöcken gleicher Farbe gehalten. Vor einer einheitlichen Hintergrundfarbe erscheint der Text in zwei weiteren Farben, die von Absatz zu Absatz wechseln. Wichtige Wörter werden in einer anderen Farbe markiert. Dieser Aufbau gibt dem Anwender die Möglichkeit, nur die gewünschte Information zu lesen und Bereiche zu ignorieren, die für ihn keinen unmittelbaren Wert haben. Zu viele Farben können aber auch Verwirrung stiften. Eine gute Faustregel ist daher: Man sollte niemals mehr als vier Farben gleichzeitig bei der Gestaltung des Bildschirms einsetzen.

Apple Macintosh

Die drei Fotografien zeigen Betriebssysteme von Microcomputern, die einen unterschiedlichen Grad an Benutzerfreundlichkeit haben. Auf dem ersten Bild versucht ein Anfänger, mit dem CP/M-Betriebssystem klarzukommen. Da CP/M keine eingebauten Hilfen besitzt, ist eine genaue Kenntnis der Befehle nötig, bevor das System richtig eingesetzt werden kann. Das zweite Beispiel zeigt ein Menüsystem – das „Log-on"-Menü für Micronet 800 auf dem Acorn B –, dessen Optionen mit Nummern versehen sind. Der Anwender braucht die gewünschte Funktion nur mit der entsprechenden Zahl aufzurufen. Da auch hier der Bildschirm nicht allzuviele Informationen bietet, muß der Anwender wissen, welche Funktionen sich hinter den Bearbeitungsmöglichkeiten verbergen. Unser letztes Bild zeigt das Betriebssystem des Apple Macintosh, das sichtbare Hilfen, grafische Darstellungen und einfache, leicht zu verstehende Menüs enthält. Mit der Maus wird jeweils eines der Piktogramme angesteuert und durch Knopfdruck gewählt.

Grafik-Routinen

Obwohl die meisten selbstprogrammierten Abenteuerspiele textorientiert sind, nutzen einige auch den Speicherplatz und die Farbgrafikmöglichkeiten, die auf den heutigen Heimcomputern zur Verfügung stehen. Der folgende Artikel ist der erste von dreien, in denen wir Beispielgrafiken und ihre Programmierung für unser zuvor entwickeltes Abenteuerspiel zeigen.

Das Digitaya-Programm, das wir entwickelt haben, ist ein textorientiertes Abenteuer. Das bedeutet, daß die vom Spieler besuchten Orte mit Textzeilen beschrieben werden. In einem solchen Abenteuer würde beispielsweise einfach die Meldung „Sie stehen vor einem Thron" dargestellt, wogegen in einem Grafikabenteuer der Thron zusätzlich auf den Bildschirm gezeichnet würde.

Die Grafiken, die wir hier entwerfen wollen, stellen zwei markante Orte von Digitaya dar: den Joystick-Port und die Arithmetik- und Logik-Einheit (ALU). Die Anzahl solcher Grafiken ist meistens durch den verfügbaren Speicherplatz begrenzt, da die zur Darstellung notwendigen Befehle Speicherplatz verbrauchen, der ansonsten für eine größere Komplexität des Einzelbildes verwendet werden könnte.

Bevor wir mit der Grafikerstellung auf dem Acorn B beginnen, müssen folgende Fragen geklärt werden:

1) Wieviel Speicherplatz steht zur Verfügung?
2) Wieviele Farben werden gebraucht?
3) Welche Auflösung ist notwendig?

Alle Punkte können mit einer Frage ausgedrückt werden: „Welchen Grafik-Modus soll man verwenden?" Höhere Auflösung und eine größere Farbauswahl bedeuten, daß wertvoller Speicherplatz verbraucht wird. Wir verwenden daher Modus 1, in dem vier Farben auf einem Bildschirm mittlerer Auflösung zur Verfügung stehen. Dieser Modus wird durch folgende Anweisung am Anfang des Programms festgelegt:

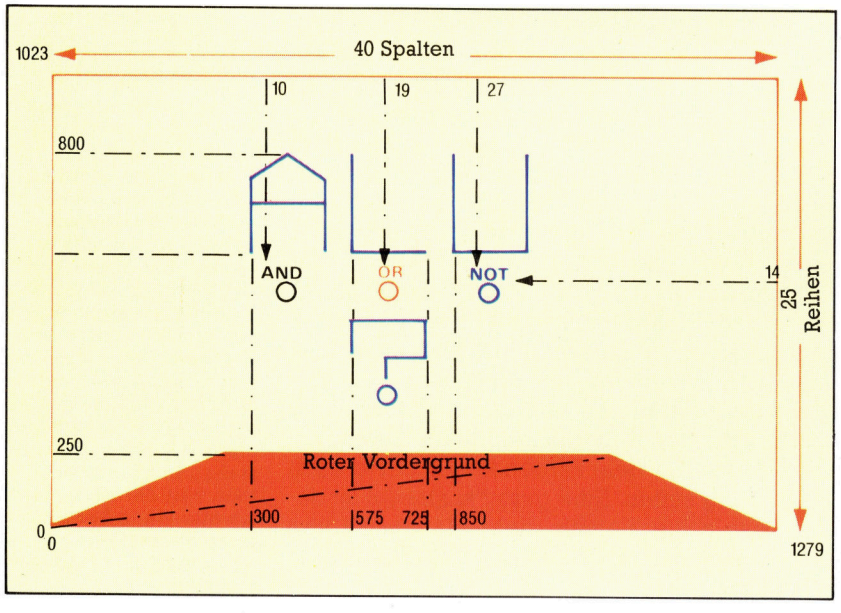
```
1095 MODE 1
```

Nach Festlegung des Modus kann die Grafik entworfen werden. Sie stellt die Buchstaben A, L und U auf dem Bildschirm dar. Der Spieler muß dann einen der drei Knöpfe – AND, OR und NOT – drücken. Auch diese Eingabe muß auf dem Bildschirm dargestellt werden. Schließlich bekommt die Grafik einen Rahmen, und der Vordergrund wird perspektivisch gestaltet. Den Rohentwurf sehen Sie im Bild links unten.

Zur Darstellung der Buchstaben wird mit MOVE der Startpunkt bestimmt und dann mit PLOT 1 das Objekt mit mehreren Linien gezeichnet, deren Positionen jeweils im Verhältnis zum letzten Punkt angegeben werden. Durch diese Methode können die Buchstaben einfach über den Bildschirm bewegt werden, indem der MOVE-Befehl geändert wird. Ein Buchstabe kann auch gelöscht werden, indem er an genau gleicher Position mit Hilfe von GCOL 3 neu gezeichnet wird.

Die Knöpfe werden mit einem umdefinierten Zeichen dargestellt. Für unser Beispiel wird CHR$(240) mit der Prozedur BUTTON umdefiniert. Beachten Sie, daß CHR$(240) zur Verwendung im Hauptprogramm der Variablen BUTTON$ zugeordnet wird. Die Knöpfe und Texte können einfach mit PRINT an durch TAB-Befehle bestimmte Stellen positioniert werden.

Zur Programmerstellung müssen verschiedene Entscheidungen getroffen werden. Hochauflösende Modi benötigen sehr viel Speicherplatz, bieten jedoch nur wenige Farben. Textmodi verbrauchen weniger Speicherplatz, bieten mehr Farben, unterstützen jedoch nur mittlere oder niedrige Auflösung. In unserem Programm bietet Modus 1 die notwendige Auflösung, allerdings auf Kosten eines 20-KByte-Bildschirmspeichers.



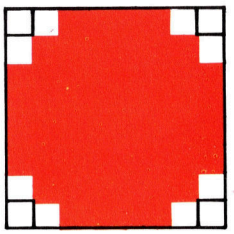
Der Vordergrund

Der Vordergrund wird mit Hilfe von PLOT 85 erstellt. Dieser Befehl verbindet den angegebenen Punkt mit den beiden zuletzt gedruckten Punkten und füllt das resultierende Dreieck mit Farbe. Zwei Dreiecke bilden ein Trapez.

Der Code für die Grafik repräsentiert eine Unterroutine innerhalb der speziellen Routine, die für den ALU-Ort des Spiels zuständig ist. Der Befehl A$=GET$ in Zeile 7560 wartet auf einen Tastendruck, bevor die Original-Vordergrundfarbe wiederhergestellt, der Bildschirm gelöscht und zur ALU-Hauptroutine zurückge-

Grafik-Programm

```
7000 REM **** ALU SCREEN S/R ****
7010 CLS
7015 REM ** CURSOR OFF **
7017 VDU23,1,0;0;0;0;
7020 REM ** BORDER **
7030 GCOL 0,1
7040 MOVE 0,0
7050 DRAW 0,1023
7060 DRAW 1279,1023
7070 DRAW 1279,0
7080 DRAW 0,0
7090 :
7100 REM ** PATH **
7110 MOVE 1279,0
7120 PLOT 85,500,250
7130 PLOT 85,800,250
7140 REM ** LETTER A **
7150 GCOL 3,2
7160 FORX=0 TO 300 STEP 10
7170 FOR I=1 TO 2
7180 PROCletter_a
7190 NEXT I,X
7200 PROCletter_a
7210 :
7220 REM ** LETTER L **
7230 FOR Y=300 TO 590 STEP 10
7240 FOR I=1 TO 2
7250 PROCletter_l
7260 NEXT I,Y
7270 PROCletter_l
7280 :
7290 REM ** LETTER U **
7300 FOR X=1280 TO 850 STEP -10
7310 FOR I=1 TO 2
7320 PROCletter_u
7330 NEXT I,X
7340 PROCletter_u
7350 :
7360 REM ** BUTTONS **
7370 PROCbutton
7380 COLOUR3
7390 PRINT TAB(11,15)button$
7400 COLOUR1
7410 PRINT TAB(20,15)button$
7420 COLOUR2
7430 PRINT TAB(28,15)button$
7440 REM ** COMMANDS **
7450 COLOUR3
7460 PRINT TAB(10,14)"AND"
7470 COLOUR1
7480 PRINT TAB(19,14)"OR"
7490 COLOUR2
7500 PRINT TAB(27,14)"NOT"
7510 :
7520 MOVE 575,400
7530 PROCq_mark
7540 :
7550 REM ** WAIT FOR KEY **
7560 A$=GET$
7562 REM ** CURSOR ON **
7564 VDU23,1,1;0;0;0;
7566 COLOUR3:CLS
7570 RETURN
7580 :
7590 DEF PROCletter_a
7600 MOVE X,600
7610 PLOT 1,0,150
7620 PLOT 1,75,50
7630 PLOT 1,75,-50
7640 PLOT 1,0,-150
7650 PLOT 0,0,80
7660 PLOT 1,-150,0
7670 ENDPROC
7680 :
7690 DEF PROCletter_l
7700 MOVE 725,Y
7710 PLOT 1,-150,0
7720 PLOT 1,0,200
7730 ENDPROC
7740 :
7750 DEF PROCletter_u
7760 MOVE X,800
7770 PLOT 1,0,-200
7780 PLOT 1,150,0
7790 PLOT 1,0,200
7800 ENDPROC
7810 :
7820 DEF PROCbutton
7830 VDU 23,240,60,126,255,255,255,255,126,60
7840 button$=CHR$(240)
7850 ENDPROC
7860 :
7870 DEF PROCq_mark
7880 PLOT 1,0,60
7890 PLOT 1,150,0
7900 PLOT 1,0,-70
7910 PLOT 1,-75,0
7920 PLOT 1,0,-50
7930 PLOT 0,-8,-30
7940 PLOT 1,16,0
7950 PLOT 1,0,-16
7960 PLOT 1,-16,0
7970 PLOT 1,0,16
7980 ENDPROC
8000 REM **** JOYSTICK PORT PICTURE ****
8010 :
8020 REM ** CURSOR OFF **
8030 VDU23,1,0;0;0;0;
8040 CLS
8050 REM ** BORDER **
8060 GCOL 0,1
8070 MOVE 0,0
8080 DRAW 0,1023
8090 DRAW 1279,1023
8100 DRAW 1279,0
8110 DRAW 0,0
8120 :
8130 REM ** HORIZON **
8140 PLOT 85,1279,319
8150 PLOT 85,0,319
8160 GCOL0,2
8170 FOR X= 0 TO 1280 STEP 32
8180 MOVE 562+X/8,320
8190 DRAW X,0
8200 NEXT X
8210 :
8220 REM ** JOYSTICK **
8230 COLOUR 2
8240 PRINTTAB(23,2)"JOYSTICK PORT"
8250 PRINTTAB(25,4)". . . . ."
8260 PRINTTAB(25,6)" . . . ."
8270 GCOL 0,2
8280 MOVE 796,893
8290 PLOT 1,288,0
8300 PLOT 1,4,-4
8310 PLOT 1,4,-4
8320 PLOT 1,0,-4
8330 PLOT 1,-4,-4
8340 PLOT 1,-30,-81
8350 PLOT 1,-4,-4
8360 PLOT 1,-4,-4
8370 PLOT 1,-218,0
8380 PLOT 1,-4,4
8390 PLOT 1,-4,4
8400 PLOT 1,-30,81
8410 PLOT 1,-4,4
8420 PLOT 1,0,4
8430 PLOT 1,4,4
8440 :
8450 REM ** SHOOT **
8460 REPEAT
8470 A$=INKEY$(10)
8480 X=RND(1279):Y=320
8490 GCOL 3,RND(3)
8500 FOR I=1 TO 2
8510 MOVE 940,836
8520 DRAW X,Y
8530 NEXT I
8540 UNTIL A$<>"":REM WAIT FOR KEYPRESS
8550 :
8560 REM ** CURSOR BACK ON **
8570 VDU23,1,1;0;0;0;
8575 COLOUR3:CLS
```



In der ALU-Grafik für den Acorn B wird das Bild eines Knopfes benötigt, um die drei Auswahl-Möglichkeiten AND, OR und NOT darzustellen. Dafür muß ein existierendes Zeichen umdefiniert werden. Unter Verwendung eines 8 × 8-Rasters kann ein Objekt entworfen und mit acht Dezimalzahlen dargestellt werden. Danach läßt sich CHR$(240) mit Hilfe des VDU-23-Befehls umdefinieren:

VDU 23,240,60,126, 255,255,255,255,126, 60

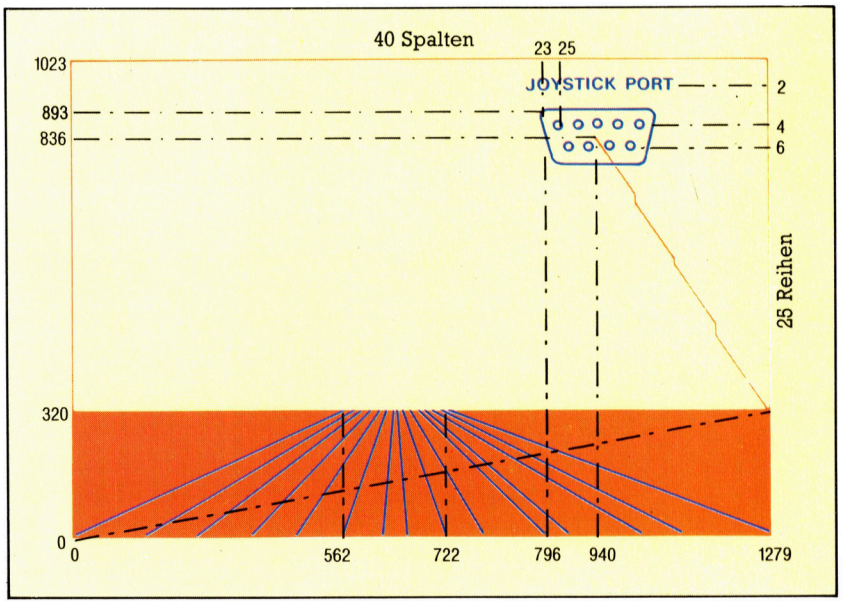
sprungen wird. Um die Grafik-Unterroutine aufzurufen, muß die folgende Zeile in das Hauptprogramm integriert werden:

4565 GOSUB 7000: REM ALU BILD U/R

Versucht ein Spieler bei Digitaya, in den Joystick-Port zu gelangen, so läuft er Gefahr, von einem Laserstrahl getroffen zu werden. Daher müssen Sie auf dem Bildschirm eine Anschlußbuchse zeichnen, aus deren Zentrum Laserstrahlen herausschießen. Der Anschluß wird durch mehrere Punkte in der oberen rechten Ecke des Bildschirms dargestellt, die mit PLOT-Anweisungen eingerahmt werden.

Beachten Sie, daß nach Festlegung der Startposition mit MOVE alle nachfolgenden PLOT-Anweisungen PLOT-l-Befehle sind – das bedeutet, sie werden relativ zum zuletzt dargestellten Punkt gezeichnet. Dies ist sehr praktisch, da man jederzeit das gesamte Objekt an einer anderen Position darstellen kann; das wird erreicht, indem man nur die erste MOVE-Anweisung ändert.

Der Vordergrund besteht aus einem rechtekkigen Farbblock, der ebenfalls aus den gefüllten Dreiecken erstellt wird. Um den Eindruck der Tiefe zu erhalten, wird mit einer FOR. . .NEXT-Schleife eine Serie in der Mitte zulaufender Linien über diesen Farbblock gezeichnet. Die Schleife setzt Werte für X und O bis 1280 – die Breite des Bildschirms in Grafikeinheiten. Die Linien werden von Startpunkten am Horizont zum unteren Rand des Bildschirmes hin gezeichnet. Dabei muß die Schrittweite von 32, die für die Endpunkte der Linien verwendet wird, im MOVE-Befehl durch 8 dividiert werden.



Der Effekt der Laserstrahlen wird durch Zeichnen einer Linie vom Zentrum der Anschlußbuchse zu einem zufälligen Punkt auf dem Horizont mit einer zufälligen Farbe erreicht. Die Linie wird dann wieder gelöscht, indem sie mit Hilfe von GCOL 3 noch einmal gezeichnet wird. Das Zeichnen und Löschen wird in einer REPEAT. . .UNTIL-Schleife parallel zur Tastaturabfrage ausgeführt. Die Schleife wird abgebrochen, sobald man eine Taste drückt. Dann wird der Bildschirm gelöscht, die Original-Textfarbe wiederhergestellt und die Programmkontrolle wieder zur JOYSTICK-Hauptroutine abgegeben. Zum Aufruf dieser Grafikroutine muß die folgende Zeile in das Adventureprogramm integriert werden:

3845 GOSUB 8000: REM JOYSTICK-BILD

Sowohl bei der Joystick-Port- als auch bei der ALU-Grafik wird der große Vorteil der relativen Zeichenmöglichkeiten genutzt. Sie erlauben einfaches Löschen und Verschieben ganzer Grafik-Objekte. Eine andere Zeichenoption wird verwendet, um mit DRAW und FILL hochauflösende Farbblöcke darzustellen.

String-Vergleiche

Wir untersuchen, wie mit den Indexregistern einfache mathematische Operationen durchgeführt werden und wie Unterroutinen funktionieren. Außerdem wird ein Programm entwickelt, das Strings miteinander vergleicht.

In der vorigen Folge erhielten Sie einen ersten Einblick in die Arbeitsweise der indizierten Adressierung des 6809. Dabei wird die aktuelle Adresse – beispielsweise OFFSET,X – aus dem Offset (Konstante oder Speicherstelleninhalt) und dem gegenwärtig im Indexregister gespeicherten Wert errechnet. Ein Offset von Null kann mit ,X angegeben werden (0,X ist ebenfalls richtig), und auch die Akkumulatoren A, B oder D lassen sich für diese Aufgabe einsetzen (zum Beispiel B,X). Beim Abarbeiten von Tabellen kann die automatische In- und Dekrementierung viel Arbeit ersparen. Dabei wird das Register entweder nach Ausführung eines Befehls um Eins oder Zwei erhöht (,X+ und ,X++) oder zuvor um Eins oder Zwei vermindert (,–Y und ,––Y).

Der Befehl LEA (Lade Adresse) kann bei der indizierten Adressierung einfache mathematische Operationen mit den Werten des Indexregisters ausführen. Da normale mathematische Befehle nur mit Akkumulatorwerten und nicht mit anderen Registern arbeiten, läßt sich zwar der Inhalt des Indexregisters in den Akkumulator D übertragen, dort in- oder dekrementieren und wieder zurückladen, doch dieser Vorgang ist sehr langsam und umständlich. LEA arbeitet zwar nur mit den Registern X, Y, S und U, führt aber alle notwendigen Adreßberechnungen durch und lädt dann die aktuelle Adresse. Da andere Ladebefehle normalerweise den Inhalt der aktuellen Adresse laden, ist dies eine brauchbare Alternative.

Sehen wir uns ein Beispiel an:

```
LEAX  –1,X
```

berechnet die aktuelle Adresse aus der Summe von –1 und dem augenblicklichen Inhalt des X-Registers. Diese Adresse wird nach X zurückgeladen und dekrementiert so den in diesem Register gespeicherten Wert. Der Befehl hat aber noch andere Funktionen: Damit Adressen nicht mehrfach berechnet werden müssen, kann er die Berechnung beispielsweise einmal ausführen und das Ergebnis speichern.

Mit dem Befehl ABX (Addiere den Inhalt des Akkumulators B auf das X-Register) kann auch das X-Register in Berechnungen einbezogen werden. Diese Anwendung ist jedoch nicht so universell einsetzbar wie LEA.

Subroutinen sind eigenständige Programmodule, die vom Hauptprogramm (oder anderen Subroutinen) aus aufgerufen werden und spezielle Aufgaben ausführen. Nach deren Beendigung übernimmt das aufrufende Modul wieder die Steuerung und führt nun den Befehl aus, der unmittelbar hinter dem Unterprogrammaufruf steht. Subroutinen werden hauptsächlich aus drei Gründen eingesetzt:

1) Der Code muß nur einmal geschrieben werden. Es ist praktisch, oft eingesetzte Funktionen als Subroutinen anzulegen, die bei Bedarf aufgerufen werden.

2) Es entsteht eine Sammlung universell verwendbarer Routinen, die sich in unterschiedlichen Programmen einsetzen lassen.

3) Programme, die in kleine Module aufgeteilt sind, lassen sich leichter handhaben.

Bei Unterprogrammen der Assemblersprache muß unbedingt beachtet werden, daß das aufrufende Programm und die Subroutine die gleichen Register einsetzen. In der Maschinencodeprogrammierung treten häufig Fehler auf, wenn nach Speicherung eines Registerwertes eine Subroutine aufgerufen wird, die diesen Wert verändert. Es sollten deshalb nicht nur alle von der Subroutine eingesetzten Register bekannt sein und dokumentiert werden, sondern auch der Registerinhalt sollte möglichst vor Aufruf der Routine gespeichert und nach Beendigung wieder geladen werden.

BSR- und JSR-Befehle

Wir werden uns später ansehen, wie der Stack (Stapelspeicher) diese Aufgabe ausführt und Werte an die Subroutinen übergeben kann. Für den Augenblick gehen wir jedoch davon aus, daß die Subroutine die gleichen Daten (globale Variablen) einsetzt wie das aufrufende Programm und daß alle weiteren Werte in den Registern abgelegt sind. Subroutinen werden mit folgenden Befehlen aufgerufen:

- BSR: Verzweige zum Unterprogramm
- JSR: Springe zum Unterprogramm

Der BSR-Befehl veranlaßt eine relative Verzweigung. Das heißt, er spricht eine Subroutine an, die eine bestimmte Anzahl Schritte vom augenblicklichen Wert des Befehlszählers entfernt ist. Dieser Befehl wird normalerweise für Subroutinen eingesetzt, die Bestandteile des Programms sind.

Dagegen spricht der JSR-Befehl Subroutinen

an, die auf festen Adressen liegen. JSR kann Unterprogramme des ROM oder Standardroutinen aufrufen, die im Speicher immer den gleichen Platz einnehmen, beispielsweise Teile des Diskettenbetriebssystems.

Wenn der Prozessor ein BSR oder JSR findet, „schiebt" er den aktuellen Wert des Befehlszählers unter Verwendung des S-Registers (Stack Pointer) auf den System-Stack. Sobald die Subroutine jedoch das S-Register für eigene Zwecke einsetzt, muß dieses Register bei der Rückkehr wieder in den ursprünglichen Zustand versetzt werden. BSR berechnet die Adresse der Subroutine und lädt sie in den Befehlszähler. Der nächste Befehl ist nun die erste Instruktion der Subroutine. Wichtig ist hierbei, daß die Subroutine mit einem Befehl und nicht mit einem Datenbyte beginnt.

Subroutinen müssen mit dem Befehl RTS (Rückkehr aus einem Unterprogramm) abschließen. Dabei wird der alte Wert des Befehlszählers wieder vom Stack „heruntergezogen" und das Programm veranlaßt, an der Stelle weiterzumachen, an der die Verzweigung eingeleitet wurde.

Unser Beispielprogramm ist komplizierter als die bislang gezeigten Module, kann aber durch den Einsatz einer Subroutine „handlicher" gestaltet werden. Das Programm durchsucht eine Tabelle mit Strings unterschiedlicher Länge und entnimmt ihr einen Wert, der einem bestimmten String zugeordnet ist. Das Format der Strings folgt dem bekannten Standard: Am Anfang steht ein Längenbyte, gefolgt von den Zeichen des Strings. Den Abschluß bildet eine 16-Bit-Adresse.

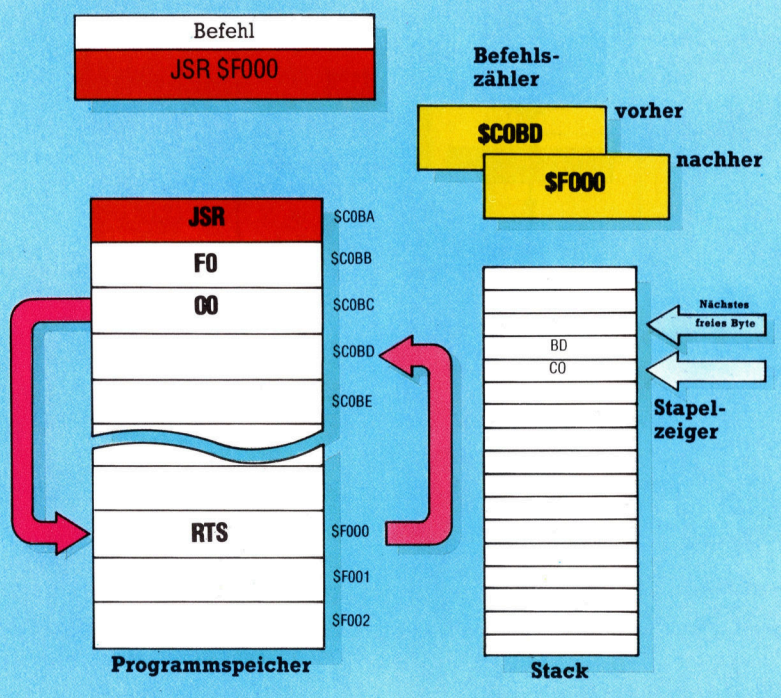
Das Ende der Tabelle wird durch einen String der Länge Null angezeigt. Das heißt, das entsprechende Längenbyte steht auf Null. In $10 ist die Anfangsadresse der Tabelle abgelegt und in $12 die Adresse des Strings, nach dem die Tabelle durchsucht werden soll. Findet sich eine Entsprechung, dann wird die Adresse in $14 gespeichert, wenn nicht, werden $12 und $14 auf Null gesetzt.

Übereinstimmungen

String-Vergleiche werden in der Programmierung oft gebraucht, speziell in der Verwaltung von String-Variablen durch BASIC-Interpreter: Alle Variablennamen werden durch Adressen ersetzt, an denen die Variablenwerte abgelegt wurden.

Unsere Programmieraufgabe läßt sich leicht in zwei Aufgabenbereiche unterteilen: Die Tabelle muß Schritt für Schritt durchsucht werden, bis entweder der String gefunden oder das Ende der Tabelle erreicht ist. Außerdem werden bei jedem Schritt zwei Strings miteinander verglichen (der gesuchte und der in der Tabelle gespeicherte).

Stacksprünge



Beim Aufruf und Rücksprung einer Subroutine (Jump and Return) wird der augenblickliche Wert des Befehlszählers gespeichert, durch die Adresse der Subroutine ersetzt und schließlich wieder in seinen ursprünglichen Zustand zurückversetzt. Der Stack ist ein Speicherbereich, in dem der Prozessor die Rücksprungadresse ablegt, der Stack Pointer (Stapelzeiger) dagegen ein 16-Bit-Register, das immer die Adresse des nächsten freien Stapelbytes enthält.

Wenn der Prozessor den Befehl JST beispielsweise an der Adresse $COBA findet, lädt er automatisch $COBD – die Adresse des auf JSR folgenden Befehls (ABX) – in den Befehlszähler. Danach wird die Subroutine ausgeführt, bis sie auf den RTS-Befehl (Rückkehr von der Subroutine) trifft. Nun wird die Rücksprungadresse $COBD wieder vom Stack „gezogen" und in den Befehlszähler eingesetzt.

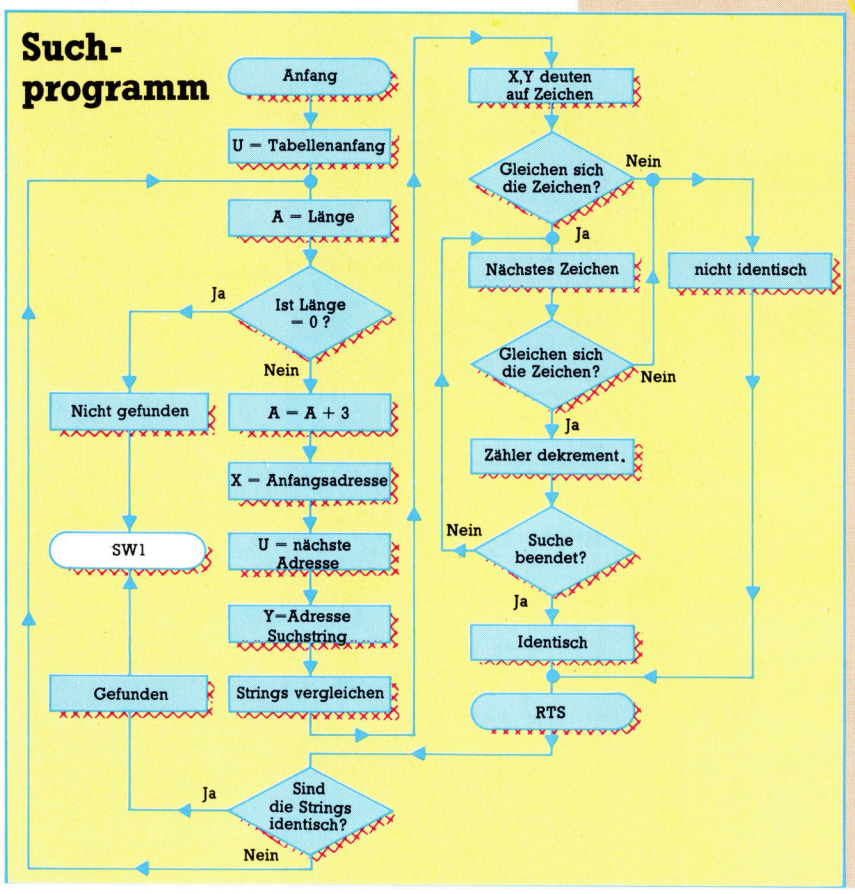
Dieser Vergleichsvorgang eignet sich ausgezeichnet für eine Subroutine – nicht nur weil er von dem Programm vielfach aufgerufen werden kann, sondern auch weil sich das Programm damit in recht handliche Aufgabenbereiche unterteilen läßt.

Die Subroutine benötigt vom Hauptprogramm zwei Informationen: die beiden Adressen der Strings, die miteinander verglichen werden sollen. Da die Subroutine die Strings byteweise bearbeitet, werden diese Werte am besten in die Indexregister X und Y gelegt. Die Unterroutine muß ebenfalls zwei Werte zurückgeben – einen Wert, der anzeigt, ob eine Entsprechung gefunden wurde oder nicht, und bei Übereinstimmung die Adresse des Strings.

Mit einem der Flags des Condition Code Registers kann auch ein Boolescher Parameter (wahr oder falsch) übergeben werden. Da dafür aber die Auswirkungen jedes einzelnen Flags bekannt sein müssen, zeigen wir dem Hauptprogramm mit $00 (nur Nullen) an, daß eine Entsprechung gefunden wurde, und mit $FF (nur Einsen) den umgekehrten Fall.

Um die Subroutine universell einsetzbar zu machen, wird beim Finden einer Entsprechung nicht die eigentliche String-Adresse zurückgegeben, sondern das X-Register auf eine Adresse gesetzt, die auf die gefundene Adresse zeigt. Da das X-Register diese Information beim schrittweisen Durchgehen der Tabelle automatisch erhält, sparen wir hierbei den Code für die Adressenspeicherung.

```
TABLE   EQU   $10
STRNG   EQU   $12
ADDRS   EQU   $14
        ORG   $1000          Anfang des Hauptprogramms
        LDU   TABLE          Tabellenanfang in U ablegen
LOOP1   LDA   ,U             Längenbyte holen
        BEQ   NTFND1         GOTO Tabellenende, falls Null
        ADDA  #3             Eins für das Längenbyte und Zwei für die Adresse zu der Stringlänge addieren
        TFR   U,X            Anfang des Tabellenstrings in X laden
        LEAU  A,U            U auf den nächsten Eintrag setzen
        LDY   STRNG          Y zeigt auf den Anfang des zu suchenden Strings
        BSR   COMPAR         Die beiden Strings vergleichen
        TSTA                 Prüfen, ob sie gleich sind
        BEQ   FOUND1         Wenn sie gleich sind, GOTO FOUND1
        BRA   LOOP1          ELSE nächsten Tabelleneintrag holen
FOUND1  LDD   ,X             Falls FOUND1, soll X auf die gewünschte Adresse zeigen
        BRA   FINSH1
NTFND1  LDD   #0             Falls nicht FOUND1, Adresse auf Null setzen
FINSH1  STD   ADDRS          Adresse speichern
        SWI                  Ende Hauptprogramm
COMPAR  LDB   ,X+            Anfang der Subroutine
        CMPB  ,Y+            Die Längenbytes holen und X und Y auf die ersten Zeichen stellen
        BNE   NOTEQ          Wenn Längen nicht identisch: GOTO NOTEQ
LOOP2   LDA   ,X+            Nächstes Zeichen des Tabellenstrings holen
        CMPA  ,Y+            Mit dem nächsten Zeichen des gesuchten Strings vergleichen
        BNE   NOTEQ          Stop, falls nicht identisch
        DECB                 Sonst Eins vom Positionszeiger abziehen
        BGT   LOOP2          Nächstes Zeichen holen
        CLRA                 A auf Null setzen, um anzuzeigen, daß die Strings identisch sind
        BRA   FINSH2         oder auf Eins, wenn sie nicht identisch sind
NOTEQ   LDA   #$FF
FINSH2  RTS                  Zurück zum aufrufenden Programm
        END
```



Nützliche Nebenwirkungen

Aufgrund ihrer Aussagenlogik kommen auch PROLOG-Prozeduren nicht ohne Kompromisse aus. In dieser Folge untersuchen wir einige praktische „Nebenwirkungen" von PROLOG und gehen auf „nicht-logische" Eigenschaften wie beispielsweise „cut" und „fail" ein.

PROLOG ist im wesentlichen eine logische Sprache. Einige Funktionen, wie das Schreiben und Lesen von Dateien oder die Ausführung mathematischer Berechnungen, lassen sich jedoch nur schwer in die Aussagenlogik einpassen, da sie anderen Regeln folgen. Für diese Aufgaben wurden in PROLOG daher einige Aussagen eingebaut, die als „Nebenwirkung" auch andere Funktionen ausführen: write(Begriff) beispielsweise läßt sich unter PROLOG immer beweisen, legt aber zusätzlich den Wert von Begriff auf den aktuellen Ausgabekanal.

Alle Ein- und Ausgaben von PROLOG werden über diese Nebenwirkungen gesteuert. Da ohne diese Funktionen die Sprache kaum brauchbar wäre, sind gewisse Kompromisse notwendig.

Das Schreiben in Dateien funktioniert wie das Schreiben auf den Bildschirm. Die Aussage tell (Dateiname) weist PROLOG an, von nun an alle Ausgaben in der angegebenen Datei zu speichern; told schließt die Datei. Nach dem gleichen Schema liest see(Dateiname) aus einer Datei, die seen wieder schließt. ASCII-Codes kann PROLOG über put(Code) senden und mit get lesen.

Für mathematische Aufgaben verfügt PROLOG über eine Reihe eingebauter Aussagen, die eigentlich Funktionen sind und vom Interpreter bewertet werden. Eine dieser „Bewertungsaussagen" ist:

```
C1 is C + 1
```

entspricht in BASIC

```
C1 = C + 1
```

Beachten Sie, daß unter PROLOG C is C + 1 immer fehlschlagen würde. In BASIC funktioniert C = C + 1 jedoch problemlos, da Sprachen wie BASIC, PASCAL, FORTRAN, ALGOL und C mit der „destruktiven" Zuordnung arbeiten. Das bedeutet, daß das Ergebnis des Ausdrucks (auf der rechten Seite) der Variablen (auf der linken Seite) zugewiesen wird und den Wert dieser Variablen damit überschreibt. Aus diesem Grund kann die gleiche Variable auf beiden Seiten des Ausdrucks erscheinen.

PROLOG verwendet jedoch keine Zuordnungen, sondern Vereinheitlichungen. Das bedeutet, daß beim Vorkommen einer Variablen jedes andere Vorkommen der gleichen Variablen dieses Satzes ebenfalls diesen Wert annimmt. Der Ausdruck C is C + 1 würde C jedoch mit zwei verschiedenen Werten belegen.

PROLOG-Dialekte

Der in dieser Serie verwandte Dialekt heißt DEC-10-PROLOG (so benannt nach dem Computer der Digital Equipment Corporation, auf dem die Sprache erstmals implementiert wurde) und gilt als Standard. Leider ist die weitverbreitete Version für Microcomputer – MICRO-PROLOG – einer der Dialekte, die sich von diesem Standard wesentlich unterscheiden. Die zweite wichtige Version – PROLOG-1 – und auch ihr Nachfolger – PROLOG-2 – kommen dem DEC-10-Standard sehr nahe.

In MICRO-PROLOG werden Programme – ähnlich wie unter LISP – ausschließlich als Listen geschrieben. Dieser Aufwand läßt sich jedoch mit dem Vorprogramm „Simple" abkürzen. Simple ist in MICRO-PROLOG geschrieben und bietet eine alternative Syntax.

Da zwischen MICRO-PROLOG und dem Standard viele kleine Unterschiede bestehen, sehen die beiden Dialekte recht unterschiedlich aus. In MICRO-PROLOG ist die Aussage das erste Element einer Liste. Hier ein Beispiel:

```
heimat_von(mars,marsmensch).
```

lautet in MICRO-PROLOG:

```
(heimat_von Mars Marsmensch)
```

In MICRO-PROLOG fehlen auch alle Verbindungselemente wie ":–" und Kommas, so daß der Satz:

```
heimat_von(Planet,Wesen):–
    geboren(Wesen,Stadt),
    in (Stadt,Planet).
```

in MICRO-PROLOG so aussieht:

```
((heimat_von x y) (geboren y z)
  (in z x))
```

Die Aussage läßt sich als eine Liste dreier Sublisten verstehen, von denen die erste Liste den Satzkopf darstellt. Durch das Simple-Programm läßt sich diese recht umständliche Syntax leichter durchschauen. Hier ein Beispiel:

```
x heimat_von y if y geboren z
  and z in x
```

MICRO-PROLOG verwendet außerdem für Listen runde statt eckiger Klammern. Argumente und Listenelemente werden durch Leerzeichen, nicht mit Kommas getrennt, und für cut wird "/" geschrieben. Eingebaute Aussagen erscheinen in Großbuchstaben. Ihre Namen und auch einige der Funktionen entsprechen nicht dem Standard.

MICRO-PROLOG (und auch fast alle anderen PROLOG-Versionen) besitzt eine Eigenschaft, die das DEC-10-PROLOG nicht hat – es kann mit Programmodulen arbeiten. Unter einer modularen Struktur lassen sich Programme in Funktionseinheiten schreiben. Die Module werden dann zusammengestellt, ohne daß man beachten muß, ob es gegensätzliche Aussagen gibt.

Die eingebaute Aussage not ist für die Abfrage negativer Informationen gedacht. Wie bei dem Booleschen NOT besagt sie, daß not(X) wahr ist, wenn X falsch ist, und umgekehrt. Die Tatsache, daß X falsch ist, zeigt PROLOG dabei auf recht einfache Weise: Eine Aussage ist falsch, wenn nicht gezeigt werden kann, daß sie wahr ist. Dazu ein Beispiel.
Standard-PROLOG:

```
sprache(pascal,schwer)
sprache(cobol,schwer)
sprache(basic,einfach)
sprache(prolog,einfach)
```

MICRO-PROLOG:

```
(sprache Pascal schwer)
(sprache Cobol schwer)
(sprache Basic einfach)
(sprache Prolog einfach)
```

enthält einige Tatsachen über die Eigenschaften verschiedener Programmiersprachen. Wenn wir fragen:

```
sprache(basic,ERLERNBARKEIT).
```

wobei ERLERNBARKEIT eine Variable ist, antwortet PROLOG:

```
ERLERNBARKEIT = einfach.
```

Wenn wir aber fragen:

```
sprache(basic,leicht).
```

gibt PROLOG die Antwort no, da das Ziel nicht bewiesen werden kann. Die Aussage ist nicht in der Datenbank gespeichert und läßt sich auch nicht aus anderen Tatsachen oder Regeln ableiten. PROLOG kann natürlich nicht wissen, daß einfach und leicht das gleiche bedeuten, und nimmt außerdem an, daß seine Datenbank eine „geschlossene Einheit" ist.

Beweismethoden

Diese Definition der „Negation als Fehlschlag" ist eine der Schwächen von PROLOG. Sie beruht auf der logischen Beweismethode des Interpreters. Diese „Auflösung" kann negative Informationen nicht direkt bearbeiten. Die folgende Regel:

```
not(A):— B, C, D.
```

(die besagt, daß X nicht eintritt, wenn B, C und D wahr sind) kann es in PROLOG nicht geben, da im Kopf eines Satzes keine negativen Aussagen vorkommen dürfen.

PROLOG ist keine reine erklärende (deklarative) Sprache. Wir haben bereits gezeigt, wie die Anordnung der Sätze des Quellentextes den Programmablauf beeinflussen kann (speziell bei rekursiven Prozeduren). Die Logik wird nun durch nichtlogische Eigenschaften wie fail und cut weiterhin eingeschränkt.

Wenn der Interpreter auf die Aussage fail trifft, wird ein „falsch" ausgelöst. Mit fail kann PROLOG Abläufe steuern. Auf die Frage an das Sprachprogramm:

```
sprache(Sprache,einfach).
```

antwortet PROLOG:

```
Sprache = basic
```

und hält an. Der Interpreter findet in der Datenbank den ersten Satz, der mit dem gesetzten Ziel übereinstimmt und sucht nicht weiter. Für eine Liste aller Sprachen, die einfach sind, müssen wir folgende Regel hinzufügen:
Standard-PROLOG:

```
einfachliste:— sprache(Sprache,
einfach),
write (Sprache), nl, fail.
```

MICRO-PROLOG:

```
((einfachliste) (sprache X einfach)
  (PX) PP FAIL)
```

Bei der Eingabe von einfachliste antwortet PROLOG:

```
basic
prolog
no
```

Backtracking

Hier versucht PROLOG, das Ziel einfachliste zu beweisen, indem es zunächst beweist, daß sprache(Sprache,einfach) wahr ist. Nachdem es die Tatsache sprache(basic,einfach) gefunden hat, wird die Variable Sprache auf den Wert basic gesetzt und so das erste Teilziel bewiesen. Als nächstes Ziel findet PROLOG nun write(basic). Da write(basic) automatisch wahr ist, findet PROLOG als letztes Teilziel nun fail. Dieses Ziel ist definitionsgemäß falsch, und so verfolgt PROLOG seinen Weg zurück bis zu write (backtracking). write (und auch die anderen bewertbaren Aussagen) lassen sich beim Backtracking jedoch nicht beweisen. PROLOG muß weiter zurück zu sprache(basic,einfach) und versuchen, diese Aussage auf einem anderen Weg zu beweisen.

Die Aussage cut (in Form eines Ausrufezeichens einzugeben) wird ebenfalls zur Steuerung des Backtrackings eingesetzt. Sie ist immer wahr, verhindert als Nebenwirkung jedoch, daß der Interpreter das Backtracking über sie hinaus fortsetzt. Damit werden alle Variablen auf die Werte fixiert, die sie in der Prozedur zu diesem Zeitpunkt angenommen haben. cut verhindert weiterhin, daß andere Sätze der Prozedur erforscht werden. Zusammen mit fail erreicht cut, daß ein mit fail abgebrochener Satz nicht nochmals angesprochen wird. Über ! und fail können wir die Aussage fehlt wie folgt definieren:
Standard-PROLOG:

```
fehlt(Satz):—Satz,!,fail.
fehlt(Satz).
```

MICRO-PROLOG:

```
((fehlt X) (?X)/FAIL)
(FEHLT X)
```

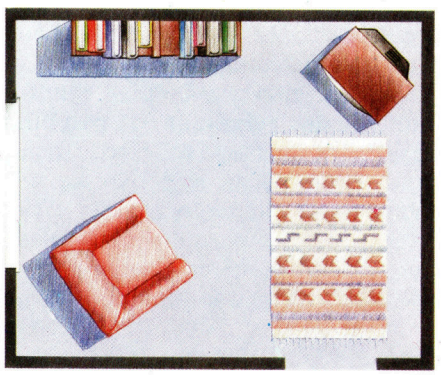
Die Aussage ist wahr, wenn der als Argument eingesetzte Satz in der Datenbank fehlt. In unserem Sprachprogramm können wir fragen:

```
fehlt(sprache(basic,einfach)).
```

Um fehlt(sprache(basic,einfach)) beweisen zu können, müssen wir sprache(basic,einfach) beweisen. Die Aussage ist wahr, da sie in der Datenbank tatsächlich vorkommt. Auch cut ist (definitionsgemäß) wahr, während fail veranlaßt, daß der gesamte Satz nicht erfolgreich ist.

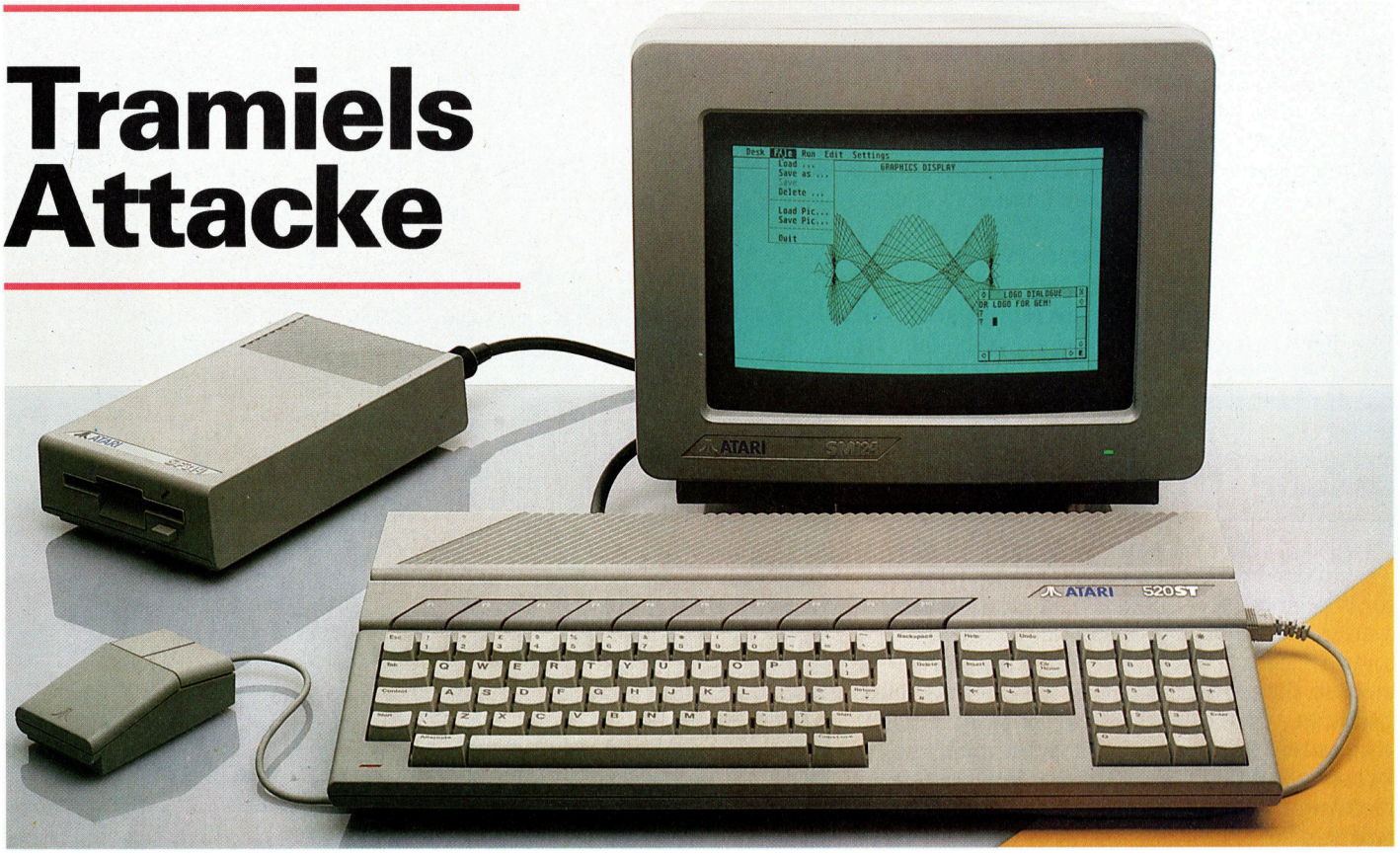
Ohne cut hätte PROLOG mit Hilfe des Backtracking versucht, einen anderen Weg zu finden, der Satz beweist. Da cut sich aber nicht umgehen läßt und es keinen weiteren Pfad gibt, kann das Hauptziel nicht bewiesen werden. Mit Hilfe dieser Methode läßt sich eine Suche nach weiteren – unergiebigen – Alternativen vermeiden.

Unser Beispiel zeigt eine wichtige Eigenschaft von PROLOG: Es kann seine eigenen Programme als Daten verwenden und eignet sich damit ausgezeichnet für die Programmierung Künstlicher Intelligenz. Dieses Thema und fortgeschrittene Programmiertechniken behandeln wir in der nächsten Folge.



Bewegung im Raum
In unserem letzten Artikel über PROLOG werden wir ein Programm entwickeln, daß die Bewegung eines Roboters durch den im Bild gezeigten Raum steuert. Die Elemente des Raumes haben dabei das Format platz(eingang) etc., während die Prozedur gehe(Platz1, Platz2) die Bewegung des Roboters darstellt. Dabei muß zuerst das Teilziel platz(Platz1) bewiesen werden – das heißt, es wird festgestellt, ob Platz1 überhaupt existiert und die Bewegung des Roboters von dort ausgehen kann.

Tramiels Attacke



Trotz eines finanziell entmutigenden Jahres für die Heimcomputer-Industrie präsentiert Atari eine Konfiguration, die die Industrie in ihren Grundfesten erschüttert. Für rund 3000 Mark gibt's den Atari 520 ST mit GEM-Betriebssystem, basierend auf dem Motorola-68000-Prozessor. Damit könnte Atari eine neue Ära für Micros einleiten.

Das Jahr 1985 wird als Jahr der Entscheidung in die Geschichte der Heimcomputerindustrie eingehen. Es ist das Jahr, in dem das exponierte Wachstum der vergangenen Jahre stoppte und die Industrie „erwachsen" wurde. Das schlägt sich in den vielfältig publizierten Problemen einer Reihe von Microcomputeranbietern nieder, die daraus resultierend Vorsicht walten lassen: Bei den meisten neueren Produkten handelt es sich um erweiterte Versionen erfolgreicher Rechner, zum Beispiel den C 128 und den Atari 130 XE.

1985 mag auch als das Jahr gewertet werden, in dem Atari das Hin und Her hinter sich ließ und sich wieder als bedeutender Heimcomputer-Hersteller behauptete. Das ist zumindest teilweise auf den Einfluß des neuen Inhabers Jack Tramiel zurückzuführen, der es verstand, als einer der „Macher" der Industrie die Aufmerksamkeit der Medien wachzurufen.

Dies allein wäre jedoch zuwenig gewesen, um Atari zu retten. Denn das Hauptproblem des Unternehmens war seine Produktlinie, die als überholt bezeichnet wurde. Die Einführung des Atari 520 ST änderte allerdings die Gesamtsituation von Grund auf. Der Computer basiert auf einem Motorola-68000-Microprozessor, einem 16-Biter – dem gleichen Chip, der auch im Macintosh von Apple verwendet wird. Der Chip gilt als am weitesten entwickelter, derzeit in großen Stückzahlen erhältlicher Chip mit interner 32-Bit-Struktur: 16 32-Bit-Register, ein 16-Bit-Datenbus und ein 24-Bit-Adreßbus.

Der 520 ST wurde auf dieselbe Art wie der 130 XE gestaltet, also mit einem stabilen grauen Kunststoffgehäuse. Die Tastatur ist in vier Sektionen geteilt, über denen sich zehn frei programmierbare Funktionstasten befinden – ebenfalls wie beim 130 XE angeordnet. Darunter liegt die Standardtastatur, die um die Alternate-Taste ergänzt wurde. Sie wird benutzt, wenn die Maus nicht angeschlossen ist, und steuert dann den Cursor.

Rechts liegt das Cursor-Steuerungsfeld, in dem sich auch die anderen Editiertasten wie „Clear" und „Insert" befinden. Darüber sind die „Help"- und „Undo"-Tasten angebracht. Ganz rechts außen steht ein numerisches Tastenfeld zur Verfügung.

Statt der früher geübten Praxis, eigene serielle Ports für Peripheriegeräte zu integrieren, hat Atari jetzt zahlreiche „Standard"-Schnittstellen eingebaut, so zum Beispiel eine Centronics-Parallel-Schnittstelle für die Druckeransteuerung und ein 25poliges RS232D-serielles Inter-

Der 520 ST ist Ataris Versuch, Marktanteile im Computermarkt zurückzugewinnen. Basierend auf dem Microprozessor Motorola-68000 zeichnet sich der Computer durch eine Fülle von Besonderheiten aus, so zum Beispiel 512 KByte RAM, MIDI-Ports und GEM-Betriebssystem. Bisher war GEM nur für teurere Rechner erhältlich.

face für den Anschluß eines Modems oder eines anderen seriellen Geräts.

Am interessantesten ist wohl die Ausstattung des Rechners mit einem MIDI-Interface. Dies ist in Form zweier DIN-Buchsen (für MIDI IN und MIDI OUT) angebracht. Somit kann der Computer den Betrieb einer Reihe von Synthesizern und anderer Musikinstrumente steuern.

Ebenfalls interessant ist das „Festplatten"-Interface. Atari hofft, als erster Hersteller einen CD-ROM-Spieler anbieten zu können. Dies ist eine Entwicklung auf Basis der Compact-Discs, die das Speichern von bis zu 800 MByte an Information auf einer einzigen Disk erlaubt. Gegenwärtig können Compact-Discs jedoch nur gelesen werden.

GEM und WIMP

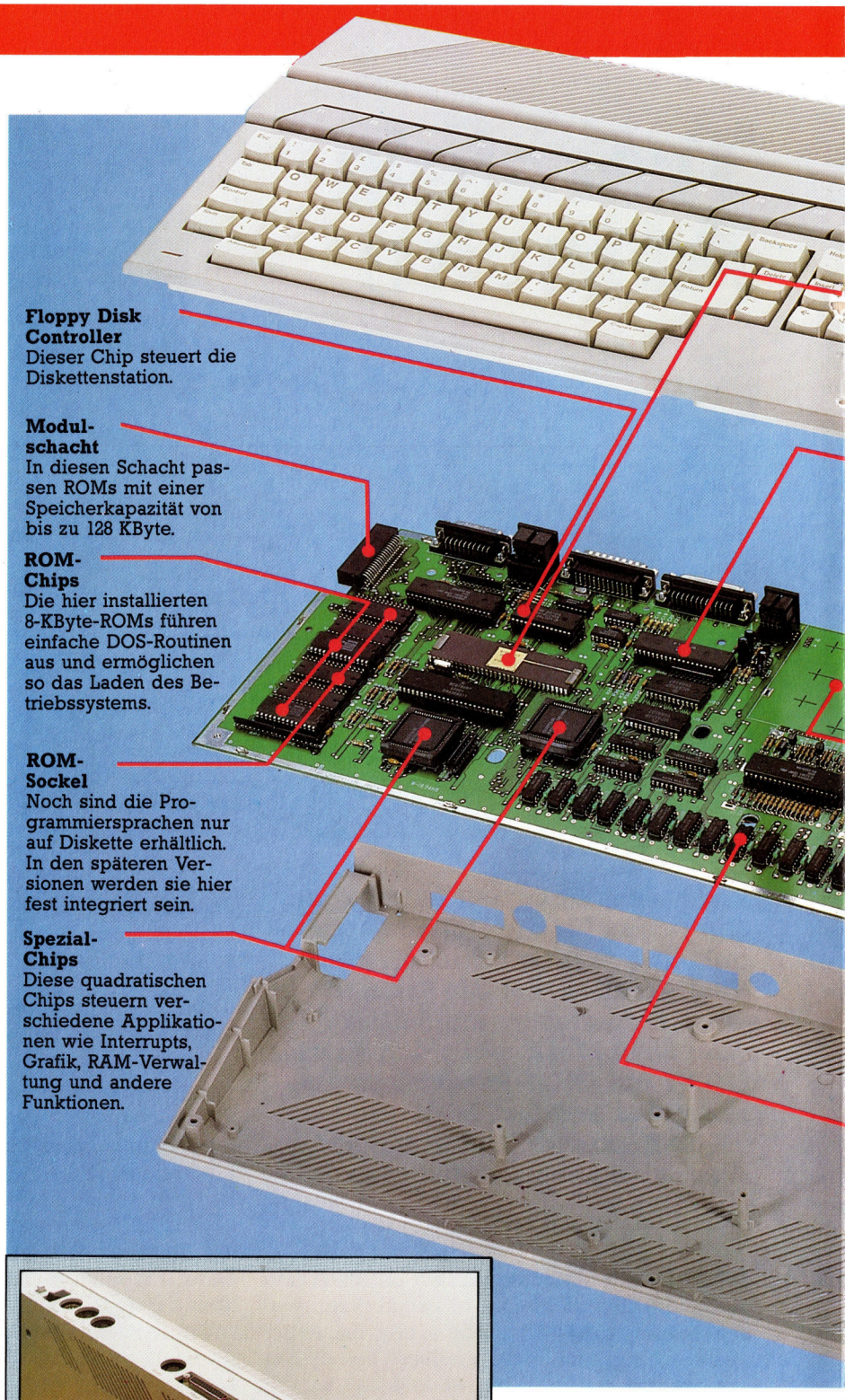
Im Preis des Atari enthalten ist eine Diskettenstation, die über das Einzellaufwerk-Interface angeschlossen wird. Bei Verwendung einer zweiten Diskettenstation wird diese an die erste gesteckt. Cassettenrecorderports sind nicht vorhanden. Atari hat sich für die 3 1/2-Zoll-Disketten von Sony entschieden. Die Diskettenlaufwerke werden als einseitige oder zweiseitige Version angeboten und speichern pro Seite 320 KByte.

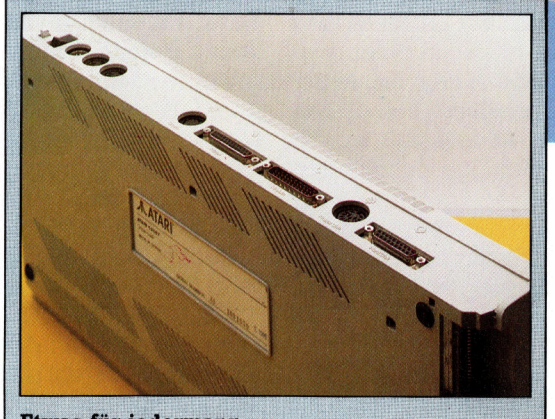
Die vollständige Übersicht der Schnittstellen: Stromversorgung, RGB-Monitor, RF-Ports (bei künftigen Versionen des Rechners), Steckmodul-Schacht, der ein 128-KByte-ROM aufnehmen kann, und rechts zwei Joystick-Anschlüsse, die allerdings in erster Linie für den Anschluß der Maus gedacht sind.

Der eigentliche Grund für den Wirbel, den das neue System verursacht hat, heißt GEM. Der 520 ST ist der erste preisgünstige Micro, der mit GEM als Standard-Betriebssystem ausgestattet wurde. Entwickelt von Digital Research, enthält GEM (Akronym für Graphics Environment Manager) ein WIMP-System (Windows-, Icons-, Mouse-Programm), das dem beim Apple Macintosh verwendeten entspricht. Tatsächlich ist denn auch die Übereinstimmung des beim Atari verwendeten GEM-Systems mit dem des Macintosh extrem groß.

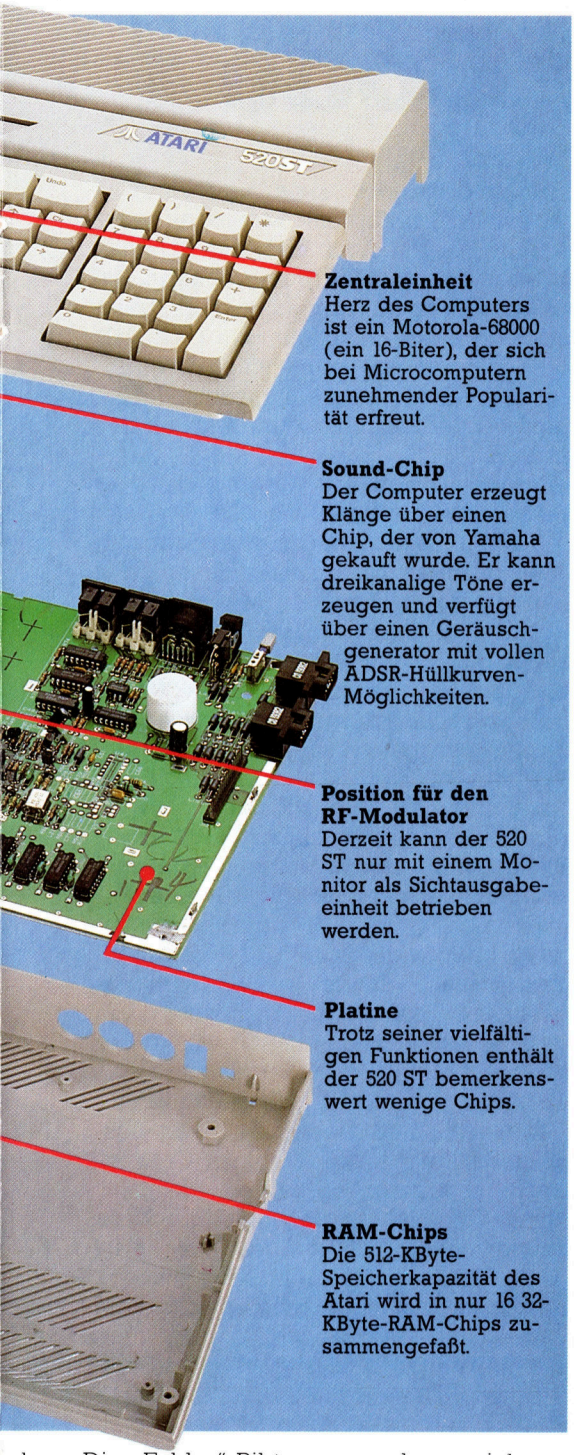
Kontrolle über das System erhält man durch einen pfeilförmigen Cursor (der sich beim Arbeiten des Rechners in eine „Biene" verwandelt), mit dem man eine Reihe von Piktogrammen ansteuert. Durch Druck auf die Maus wird das Piktogramm gewählt. Am oberen Bildschirmrand befinden sich außerdem mehrere Menüs, die als Fenster „heruntergezogen" werden können. Das Diskettenlaufwerk beispielsweise wird durch Wahl eines „Filing Cabinet"-Piktogramms in der oberen linken Ecke des Bildschirms aktiviert. Durch Herunterziehen des „Files"-Menüs können verschiedene File-Manipulationen ausgeführt werden. Nach Wahl von „Options" wird das Directory dargestellt.

Im GEM werden zwei File-Arten ausgege-





Etwas für jedermann
Atari hat seine Strategie geändert: Früher konnten Computer des Unternehmens (ohne zusätzliches Interface) nur mit Atari-eigener Peripherie ergänzt werden. Der 520 ST dagegen wurde mit mehreren „Standard"-Schnittstellen ausgestattet. Darunter sind RS232-, Centronics- und MIDI-Schnittstelle.



ben. Die „Folder"-Piktogramme kennzeichnen „Directories", wogegen die quadratischen Piktogramme Programm-Files darstellen. Will man ein File löschen, ist nur das entsprechende Piktogramm auf das „Papierkorb"-Piktogramm zu legen. Die Leichtigkeit, mit der selbst ein Anfänger den Umgang mit GEM erlernen kann, wird besonders denjenigen deutlich, die mit Betriebssystemen wie CP/M und MS-DOS vertraut sind.

Der 520 ST arbeitet in drei Grafik-Modi: hoch-, mittel- und niedrigauflösend. Der höchste Auflösungsmodus bringt 640 mal 400 Punkte, allerdings nur einfarbig, wodurch er mit dem des Mac identisch ist. In den niedrigeren Auflösungs-Modi werden die vielfältigen Farbmöglichkeiten des Rechners deutlich. Viele Anwender bevorzugen jedoch die monochrome Darstellung. Atari bietet für den 520 ST wahlweise einen monochromen und einen mehrfarbigen Monitor an.

Das GEM „Schreibtisch"-System (so benannt, da man die Piktogramme auf dieselbe Art bewegt, wie man Blätter auf einem Schreibtisch verschieben würde) läuft unter TOS – Tramiel Operating System. Hierbei handelt es sich um eine spezielle Version von Digital Researchs CP/M-68-Betriebssystem, das wiederum eine Weiterentwicklung des Digital Research-eigenen Acht-Bit-OS ist und für Rechner mit dem 68000-Prozessor umgeschrieben wurde.

Für den 520 ST steht eine Reihe mitgelieferter Programme zur Verfügung. Zunächst ist da GEM-Write, ein Textverarbeitungsprogramm, dazu GEM-Paint, das, wie der Name schon sagt, Grafikmöglichkeiten bietet. Ähnlich wie bei den Macintosh-Äquivalenten MacPaint und MacDraw stehen damit vielfältige „Pinsel"-Größen und andere Zeichenhilfen zur Verfügung. Zudem kann gezoomt werden, womit der Anwender auch kleinste Flächen auf dem Bildschirm exakt gestalten kann.

Richtungsweisend

Zum Lieferumfang des Computers gehören die Programmiersprachen ST-BASIC und ST-LOGO – für die zuerst ausgelieferten Rechner auf Diskette, später aber als ROM-Modul. Wie TOS handelt es sich dabei um Weiterentwicklungen von anderen Digital-Research-Produkten. Das BASIC ist eine spezielle Version von Digital Researchs Microsoft-Basic, bekannt als Personal-BASIC, wogegen es sich beim LOGO um eine Version des populären DR. LOGO handelt, das jetzt im Lieferumfang vieler Rechner enthalten ist. Die Sprachen wurden so adaptiert, daß sich die „Fenster"-Möglichkeiten des GEM voll nutzen lassen.

Zweifellos ist der Atari 520 ST eine wichtige Maschine – nicht nur für die Zukunft der Atari Corporation selbst, sondern auch für den übrigen Computermarkt. Das komplette System enthält eine Diskettenstation, einen monochromen Monitor, eine Maus sowie ein Software-Paket. Somit werden die Wünsche des „ernsthaften" Heimanwenders weitestgehend erfüllt.

Mit den GEM-Programmen und 512 KByte Speicherkapazität erhält der Anwender eine Konfiguration, die der des erweiterten Mac kaum nachsteht und nicht einmal die Hälfte dessen kostet. Sollte eine „abgespeckte Version" des Computers mit 256 KByte erhältlich sein, wären die Verkaufschancen vielleicht größer. Eines aber ist sicher: Sollte der 520 ST in ausreichenden Mengen zur Verfügung stehen, wird sich der Computermarktbereich der preiswerten Geräte in eine völlig neue Richtung bewegen, und eine neue Ära wird eingeleitet sein.

Atari 520ST

ABMESSUNGEN
470 × 240 × 60 mm

ZENTRALEINHEIT
Motorola 68000, 8 MHz

SPEICHERKAPAZITÄT
512 KByte RAM

BILDSCHIRM-DARSTELLUNG
Drei Auflösungs-Modi, der höchste bei einfarbiger Darstellung mit 640 × 400 Punkten.

SCHNITTSTELLEN
MIDI (zwei DIN-Buchsen), Centronics-parallel-Schnittstelle, RS232-seriell, Diskettenstation-Schnittstelle, Stromversorgung, RGB-Monitor, RF-, Modul- und Joystick-Ports. Dazu steht ein „Festplatten"-Interface für künftige CD-ROM-Spieler zur Verfügung.

DISKETTENSTATION
Ein- und zweiseitig, Sony 3 ½-Zoll-Laufwerk. Jede Seite speichert 320 KByte.

BETRIEBSSYSTEM
TOS und GEM als Standard.

PROGRAMMIERSPRACHEN
ST-BASIC und ST-LOGO

TASTATUR
84 Tasten und zehn Funktionstasten.

DOKUMENTATION
Ausführliches Handbuch, sehr übersichtlich mit zahlreichen Diagrammen und Bildschirmfotos, die den umfangreichen Text ergänzen.

STÄRKEN
Der Computer bietet 16-Bit-Technik, große Speicherkapazität und ein benutzerfreundliches Betriebssystem, das bisher nur auf erheblich teureren Rechnern zur Verfügung stand.

SCHWÄCHEN
Trotz seiner beeindruckenden Spezifikationen muß sich das System erst einmal bewähren. Derzeit ist der Rechner noch zu teuer, um in größeren Stückzahlen verkauft zu werden. Und vom Verkauf hängt die künftige Softwareunterstützung ab.

Genetischer Code

Im zweiten Teil unserer Untersuchung von „Lernenden Rechnern“ beschäftigen wir uns mit einem Programm, das die Simulation des Evolutionsprozesses erlaubt.

Die Natur gibt der KI-Forschung grundlegende Anregungen für die Entwicklung selbstlernender Systeme: das Nervensystem, das Abwehrsystem (Immunsystem) und der Evolutionsprozeß selbst. Das Nervensystem, speziell das menschliche Gehirn, hat einen äußerst effektiven Lernmechanismus. Doch seine Funktionsweise ist noch immer unbekannt – viel ist zu erforschen, um seine Methoden nachvollziehen zu können. Auch über das Abwehrsystem des Körpers ist nur wenig bekannt. Es dient als Vorbild, da es Fremdkörper erkennen und zerstören kann. Im Laufe eines Lebens lernt es Abermillionen unterschiedlicher Proteine kennen. Ohne seine erstaunliche Adaptions- und Widerstandsfähigkeit würden wir bald sterben.

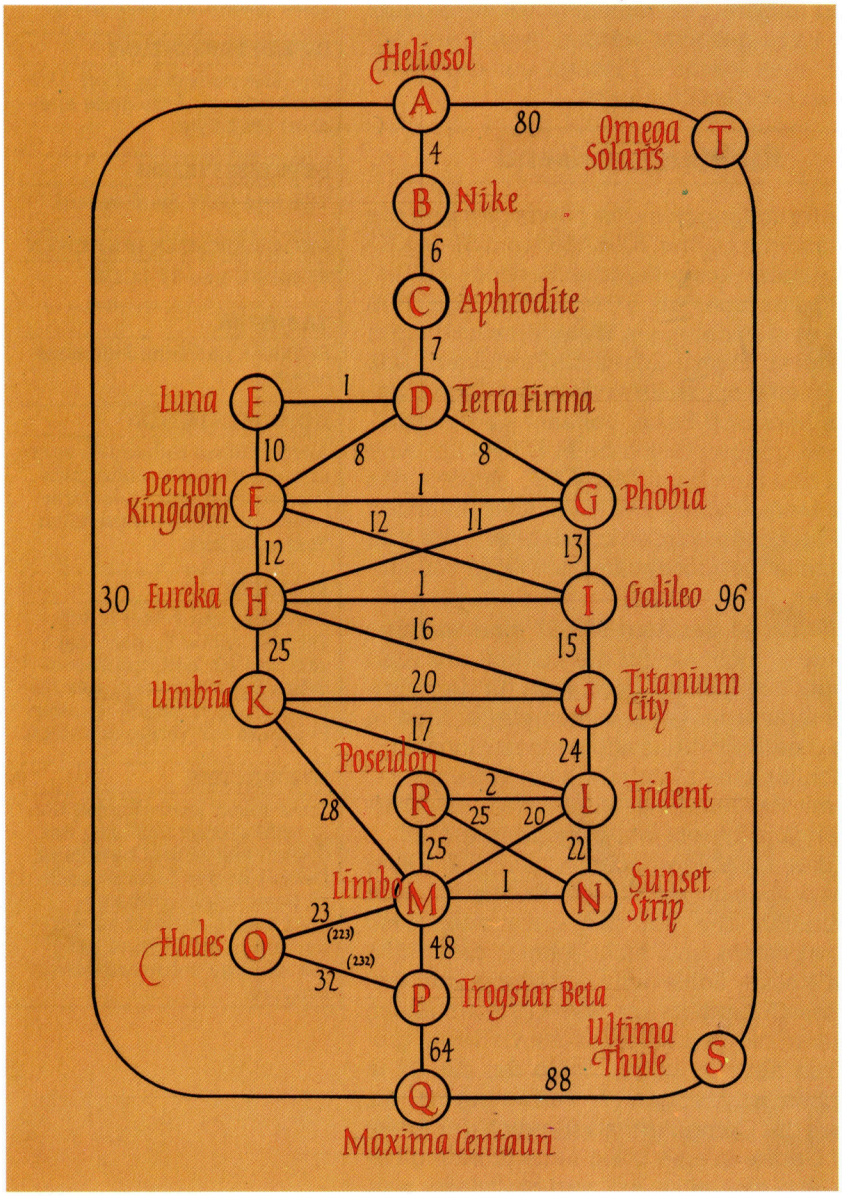
Der Evolutionsprozeß ist nützlich, denn er produziert immer weiterentwickelte Organismen. Er mag für unsere Zwecke etwas langsam sein, kann in einer Computersimulation aber beschleunigt werden. Vor allem ist er relativ gut verständlich und einfach genug, um mit Hoffnung auf Erfolg kopiert werden zu können. In der vorhergehenden Ausgabe beschäftigen wir uns mit der Maschinenlernmethode nach dem Darwinschen Prinzip und zeigten die theoretischen Vorteile auf. Hier sehen wir, was bei der Umsetzung in die Praxis geschieht.

Um das Maschinenlernen unter Verwendung eines Evolutions-Algorithmus zu verdeutlichen, wenden wir uns dem bekannten „Travelling Salesman Problem“ (kurz „TSP“) zu. Gemeinhin wird das Problem so dargestellt, daß die Hauptstädte der 48 Bundesstaaten der USA (ohne Hawaii und Alaska) besucht werden müssen. Dem Vertreter (Salesman) steht eine Übersicht der Entfernungen zwischen den Städten zur Verfügung. Er wird aufgefordert, jede Stadt nur einmal zu besuchen und zu seinem Ausgangspunkt zurückzukehren. Ziel ist es, die Gesamtreisestrecke so kurz wie möglich zu halten.

Das klingt vergleichsweise einfach, doch die Zahl der möglichen Strecken oder Touren ist (N-1)!, wobei N die Menge der Städte darstellt. Auf einer Rundreise von 48 Städten gibt es 47 erste Aufenthalte, gefolgt von 46 möglichen zweiten Unterbrechungen, gefolgt von 45 dritten usw. Und selbst wenn nur 20 Städte zu besuchen wären, betrüge die Zahl möglicher Routen über 120 Billiarden.

Es gibt verschiedene Lösungsansätze für diese Aufgabe. Beispielsweise kann sie als Suchproblem verstanden werden. Wir dagegen wenden eine recht unorthodoxe Lösungsmöglichkeit an: Wir betrachten es als ein Lernproblem. Wir suchen nach einer Methode, die die ungeheure Vielzahl möglicher Lösungen ökonomisch durchgeht. Wir erwarten keine optimale Lösung, aber eine gute.

Das nachfolgende GENE (General Evolutionary Network Explorer)-Programm ist ein Lernsystem, das speziell für die Untersuchung von Netzwerken entwickelt wurde und Gesetze errechnet, die immer ökonomischere Strecken in-



Die Intergalaktische Reisekarte des Partyfreundes zeigt die Hyperraumstrecken zwischen den Planeten und die jeweilige Reisezeit. Es ist möglich, zwischen nicht direkt miteinander verbundenen Planeten zu reisen, doch dauert dies 1000 Zeiteinheiten länger. Aufgabe des GENE-Programms ist es, eine schnelle Reiseroute durch das Netzwerk zu finden, so daß alle 20 Planeten nur einmal besucht werden.

nerhalb des Netzwerks ermittelt. Bevor wir das Programm im Detail beschreiben, ist die Entwicklung eines Netzwerks erforderlich, das es erforscht. Ein gutes Beispiel dafür ist „The Gatecrasher's Guide To The Galaxy", in dem unter anderem alle Planeten im Umkreis der Erde von 80 Lichtjahren aufgelistet sind, in denen samstags nachts Parties stattfinden. Mit Hilfe der Karte können wir das TSP in das PPCD (Planetary Party Crawl Dilemma) umwandeln, dessen Ziel es ist, innerhalb einer Nacht auf allen 20 Parties zu erscheinen und zum Ausgangspunkt zurückzukehren.

Galaktische Reiserouten

Die Karte links zeigt die Hyperraum-Routen unseres galaktischen Systems, ergänzt um die zur Reise zwischen den Aufenthalten erforderlichen Zeiten. Knoten, die nicht durch Hyperraum-Autobahnen miteinander verbunden sind, werden mit 1000 Zeiteinheiten für Passierkosten belegt.

In einem klassischen Evolutions-Lernschema gibt es eine Anzahl Strukturen, die als „Pseudo-Organismen" behandelt werden. Jede dieser Strukturen (die wir als „Regeln" bezeichnen) definiert eine mögliche Lösung des Problems. Sie werden auch zur Erzeugung neuer Strukturen verwendet (das sogenannte „Offspring"). Das geschieht derart, daß Merkmale biologischer Reproduktionen nachgeahmt werden – so etwa wie Eltern einige ihrer Merkmale ihren Kindern weitervererben.

Abhängig von ihrer Rolle innerhalb der Aufgabe werden Regeln ausgewählt, die am längsten zu überleben scheinen und die größte Fortpflanzungswahrscheinlichkeit haben. In GENE haben diese Organismen/Regeln eine sehr einfache Struktur. Im BASIC-Programm sind sie als Strings enthalten. So bezeichnet im Beispiel

ABJHMNCDKTSFRQEGILOP

jeder Buchstabe einen Planeten im Planeten-

Alphabetische Ingenieurskunst

Zunächst werden die beiden Elternstrukturen zufällig aus der Menge ausgesucht, die den Sortierprozeß überlebt haben (vermutlich sind daher die Eltern „besser" als der Durchschnitt). Eine davon wird als „Spender" bezeichnet (R1%), die andere ist der Rezipient (R2%) (Zeilen 3030 bis 3060).

Ein Stück des „genetischen Materials" (bei dem es sich lediglich um ein Substring handelt) wird dann per Zufall aus dem Spender genommen und in S$ plaziert. Darauf wird Subroutine 3300 aufgerufen, die das Stück des Spenders mit dem Rezipienten verbindet. Zeichen, die dann doppelt wären, werden fallengelassen. Die Reihenfolge der restlichen Zeichen im Offspring entspricht der Reihenfolge, die diese im Rezipienten hatten. Der „Paarungsprozeß" erfolgt also asymmetrisch. Das Paaren von X mit Y ergibt nicht dasselbe Ergebnis wie das Paaren von Y mit X, selbst wenn die Zufallspositionen P1% und P2% identisch sind.

Betrachten wir einmal ein kurzes, gerafftes Beispiel:
Zwei Eltern sind gegeben:

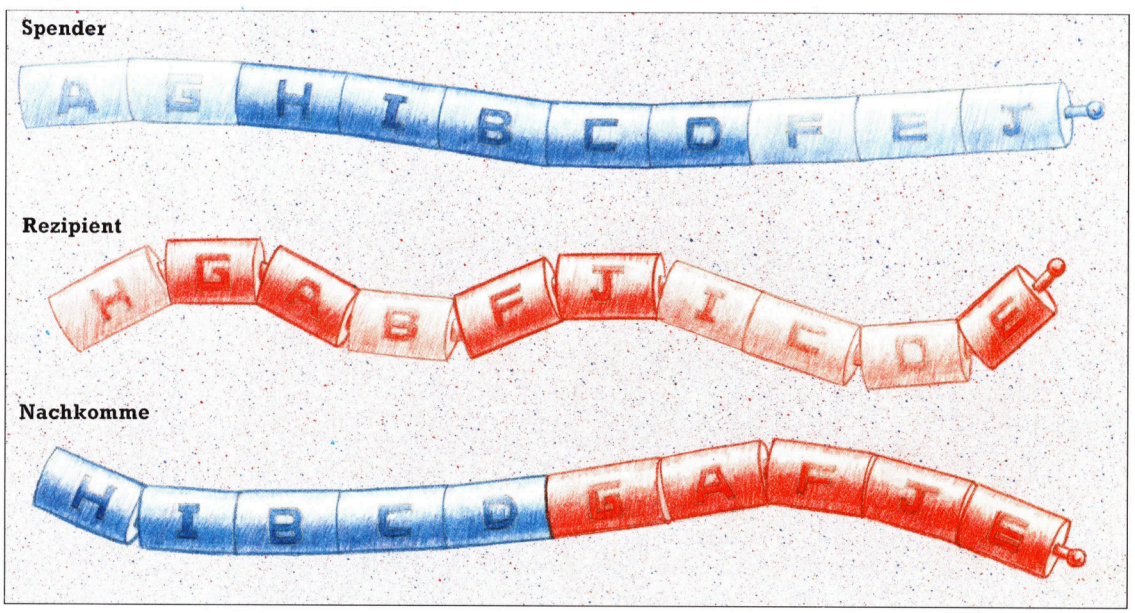
Spender: A G H I B C D F E J
Rezipient: H G A B F J I C D E
Wir können den Substring

H I B C D

als Beitrag des Spenders herausnehmen und ihn mit den verbleibenden Buchstaben des Rezipienten verknüpfen zu:

H I B C D G A F J E.

Dies ist nicht der einzige Weg zur Kreuzung eines Paares von Gesetzes-Strings. So wird jedoch sichergestellt, daß die Teile der elterlichen Information an die nächste Generation weitergegeben werden. Sichergestellt ist ebenfalls, daß jede Paarung einen „echten" gültigen Nachwuchs zur Folge hat.



GENE erzeugt neue Netzwerk-Routen, indem es zwei erfolgreiche Routen miteinander paart. Da diese als 20-Zeichen-BASIC-Strings dargestellt sind, erfolgt der Prozeß der Nachwuchszeugung durch einfache String-Manipulation. In unserem Beispiel wird der Substring HIBCD von einem Elternteil genommen (dem Spender) und mit dem anderen Elternteil verknüpft (dem Rezipienten). Dabei ist sichergestellt, daß kein Buchstabe doppelt vorhanden ist. Ferner ist wie bei realen genetischen Kreuzungen garantiert, daß die Merkmale der Eltern an den Nachwuchs weitergeleitet werden.

Netzwerk und taucht nur einmal in dem String auf. Deshalb ist jeder 20 Zeichen umfassende String eine Veränderung der Reihenfolge der zu besuchenden Planeten und definiert eine bestimmte Route innerhalb des Netzwerks.

Zur Evaluierung der Strings werden die benötigten Entfernungen addiert. Je kleiner die kombinierte Reisestrecke, desto besser ist die Gesamtroute.

Die Strings, die schlechter als der Durchschnitt sind, werden nach jeder „Generation" gelöscht. Woran sich die Frage knüpft, wie sie zu ersetzen sind. Folgen wir der biologischen Analogie, müßten wir etwas der geschlechtlichen Reproduktion Vergleichbares als Ersatz einführen. Doch diese Reproduktion ist nicht der einzige Weg zur Erzeugung neuer Strings. Etwa acht Prozent der überlebenden Strings entstehen durch Mutationen.

Die Mutations-Subroutine, beginnend bei Zeile 3500, führt eine bestimmte Menge von Zufalls-Austausch aus. Jedoch ist Mutation nicht der primäre genetische Operator, sondern ein Hintergrund-Operator, der sicherstellt, daß das System nicht in einem lokalen Optimum hängenbleibt.

Experimentiert man mit dem Programm, wird man feststellen, daß der durchschnittliche Standard der Gesetze mit der Zeit tatsächlich besser wird – auch wenn die besten Regeln „sterben". Dabei handelt es sich allerdings nicht um eine fortgesetzte Progression. Das Programm mogelt, indem es das beste Gesetz beibehält und es nur dann ersetzt, wenn ein besseres da ist.

Zufallslösungen

Man kann das System „feinabstimmen", indem man mit den Werten für die „Todesrate" in Zeile 2520 spielt. Das Zufallselement in dieser Zeile setzt die Überlebensrate der guten Gesetze von einer Generation zur nächsten auf 88 Prozent fest, kann aber geändert werden. Ebenfalls ist eine Änderung der Mutations-Rate in Zeile 3520 möglich. Zudem können alternative Mutations-Operatoren eingeführt werden, wie auch eine völlige Umkehrung eines Gesetzes-Strings möglich ist.

Es bleiben zwei Fragen:
1) Wie gut ist die Methode?
2) Warum funktioniert sie?

Die erste kann durch den Vergleich mit einer reinen Monte-Carlo-Lösung beantwortet werden. Alle genetischen Algorithmen sind letztlich modifizierte Monte-Carlo-Prozeduren, in denen Zufallslösungen generiert werden. Die beste Lösung bleibt erhalten. All dies geschieht meist innerhalb eines bestimmten Zeit-Limits.

Wenn Sie ein solches Programm schreiben, werden Sie feststellen, daß es schnell recht gute Lösungen finden kann, sich mit zunehmender Zeit aber weniger verbessert, als erwartet wird. Die beste Lösung nach 20 000 Versuchen ist nur minimal besser als die nach 10 000 Versuchen. Generell gilt, daß genetische Algorithmen nach längerer Laufzeit besser werden. Fragen wir uns nach dem „Warum?", kommen wir zur Beantwortung der zweiten Frage.

Eine reine Monte-Carlo-Methode ist im Prinzip eine blinde Suche. Andererseits benutzt ein genetischer Algorithmus das, was er findet, für seine weitere Suche. Bestimmte Muster, die zur guten Durchführung beitragen, werden gespeichert, durch die Wissensbasis fortgepflanzt und in minimal unterschiedlichen Zusammenhängen neu kombiniert. Die Suche wird vorrangig in Regionen des (multi-dimensionalen) „Problem-Raums" geleitet, in dem gute Resultate gefunden wurden.

Das GENE-Programm

Das Lernsystem (für den Acorn B) enthält folgende Datenstrukturen:

NAME$ (SIZE%)	Namen der Netzwerk-Knoten
LINK% (SIZE%,SIZE%)	Distanz zwischen den Knoten
NX% (SIZE%)	Wird in der „Paarungs-Routine" eingesetzt
R$ (NR%)	Regel-Strings
RV (NR%)	Werte der Regeln

NX% wird angewandt, um den Beitrag eines Rezipienten für ein neues Gesetz so einzuflechten (Routine 3300), daß jeder Knoten/Buchstabe nur einmal im „Offspring" auftaucht.

R$ enthält die derzeitige Population von Gesetzen als Strings der Länge SIZE%. Jede definiert eine bestimmte Route.

RV gibt den „Wert" jedes Gesetzes an, entsprechend der Länge der Strecke, die es darstellt. Wenn RV (R%)=0, ist Gesetz R% „tot" und muß in der nächsten Generation ersetzt werden. (Strecken zum Nulltarif sind offensichtlich nicht möglich.)

SIZE% kann geändert werden, wenn man verschiedene Kartennetze benutzen will. In diesem Fall sind die Zeilen 8000 und folgende zu ändern. NR% läßt sich verändern, um die Wirkung von Regeln in der Population zu beobachten.

```
  10 REM ************
  11 REM **  GENE  **
  13 REM ************
 100 GOSUB 1000  : REM set up map matrix
 101 G%=0: SN%=0
 105 B=SIZE%*1000: B$=""
 110 INPUT "How many generations ", MG%
 111 GOSUB 1700  : REM initial rules.
 115 IF SN%=0 THEN INPUT "Start Node is No. ", SN%
 120 REM **** MAIN PROGRAM LOOP ****
 130 G%=G%+1
 140 PRINT "Generation ";G%
 150 GOSUB 2000  : REM evaluate rules
 160 GOSUB 2500  : REM kill bad rules
 170 GOSUB 3000  : REM mate good ones
 180 GOSUB 3500  : REM mutations
 190 GOSUB 4000  : REM tidy up
 200 IF G%<MG% THEN 120
 220 GOSUB 5000  : REM dump new rules
 250 END
 999 :
1000 REM -- ROUTINE TO SET UP MAP-NET:
1001 SIZE%=20: NR%=24
1010 DIM NAME$(20), LINK%(20,20)
1011 DIM NX%(SIZE%)
1012 DIM R$(NR%), RV(NR%)
1013 REM rules and rule-values.
1015 FOR I%=1 TO SIZE%
1020 FOR J%=1 TO SIZE%
1022 LINK%(I%,J%)=1000  : REM default
1023 IF I%=J% THEN LINK%(I%,J%)=0
```

```
1025 NEXT : NEXT
1030 NC%=0 : L%=0
1032 FOR I%=1 TO NR%: RV(I%)=0: NEXT
1033 RESTORE
1040 REM **** READ NAME AND NODE NO. ****
1050 READ N$,ID%
1055 PRINT N$,ID%
1060 IF ID%<>0 THEN GOSUB 1500
1070 IF ID%<>0 THEN 1040
1080 PRINT NC%;" LOCATIONS READ IN."
1088 PRINT L%;" non-default links."
1090 RETURN
1100
1500 REM -- INDIVIDUAL NODE & CONNECTIONS:
1510 NC%=NC%+1
1520 IF ID%<>NC% THEN PRINT "WARNING: Node ";ID%;
" out of order."
1530 NAME$(NC%)=N$
1550 REM **** SET UP INTERNODE DIST ARRAY ****
1560 READ NI%,NT%
1570 LINK%(ID%,NI%)=NT%
1588 REM zero-links don't matter.
1590 L%=L%+1
1600 IF NI%>0 THEN 1550
1610 RETURN
1620 :
1700 REM -- Initial Dummy Rules:
1710 S$=LEFT$("ABCDEFGHIJKLMNOPQRSTUVWXYZ",SIZE%)
1715 GOSUB 1780 : REM read file if any
1720 FOR R%= 1 TO NR%
1730 R$(R%)=S$
1740 PRINT S$;R%
1750 I%=INT(RND(1)*SIZE%+1): J%=INT(RND(1)*SIZE%+1
1760 GOSUB 6000:REM SWAP
1770 NEXT
1775 RETURN
1777 :
1780 INPUT "Old Rule File (RETURN if none)", RF$
1790 IF RF$="" THEN RETURN
1800 F%=OPENUP(RF$)
1810 INPUT #F%, B$, B, SN%
1820 CLOSE #F%
1825 S$=B$
1830 REM only the top one.
1840 RETURN
1850 :
2000 REM -- Rule Evaluation Routine:
2010 T=0
2020 FOR R%=1 TO NR%
2030 IF RV(R%)<=0 THEN GOSUB 2200
2040 T=T+RV(R%)
2050 NEXT
2060 AV=T/NR% : REM average value
2070 PRINT "Average score = "; AV
2080 RETURN
2100 :
2200 REM -- Single Rule Evaluation:
2210 S$=R$(R%)
2220 P1%=SN% : REM start node
2230 GT%=0
2240 FOR S%=1 TO LEN(S$)
2250 P2%=ASC(MID$(S$,S%,1))-64
2260 IF P2%=SN% THEN GOTO 2290
2270 GT%=GT%+LINK%(P1%,P2%)
2280 P1%=P2%
2290 NEXT
2300 RV(R%)=GT%+LINK%(P2%,SN%)
2310 RETURN
2320 :
2500 REM -- Routine to Kill Bad Rules:
2510 FOR R%=1 TO NR%
2515 IF RV(R%) < B THEN B=RV(R%): B$=R$(R%)
2520 IF RV(R%) > AV OR INT(RND(1)*100+1)>88 THEN
R$(R%)="": RV(R%)=0
2530 NEXT
2540 RETURN
2550 REM -- NB smaller values are better.
2560 :
3000 REM -- Mating Routine:
3010 FOR R%=1 TO NR%
3020 IF RV(R%)>0 THEN GOTO 3120
3030 R1%=INT(RND(1)*NR%+1):IF RV(R1%)<=0 THEN 3030
3050 R2%=INT(RND(1)*NR%+1):IF RV(R2%)<=0 THEN 3050
3070 REM 'parents' chosen.
3075 P2%=INT(RND(1)*(SIZE%-1)+1)
3080 P1%=INT(RND(1)*SIZE%+1) : REM splice-points
3090 S$=MID$(R$(R1%),P1%,P2%)
3100 GOSUB 3300 : REM splice in rest
3110 R$(R%)=S$
3120 NEXT
3140 RETURN
3150 :
3300 REM -- Gene Splicing Routine!
3310 FOR S%=1 TO SIZE%
3320 NX%(S%)=0 : NEXT
3330 FOR S%=1 TO LEN(S$)
3340 SX%=ASC(MID$(S$,S%,1))-64
3350 NX%(SX%)=NX%(SX%)+1
3360 NEXT
3370 FOR S%=1 TO LEN(R$(R2%))
3380 SX%=ASC(MID$(R$(R2%),S%,1))-64
3390 IF NX%(SX%)>0 THEN GOTO 3420
3400 S$=S$+MID$(R$(R2%),S%,1)
3410 NX%(SX%)=NX%(SX%)+1
3420 NEXT
3440 RETURN
3450 :
3500 REM -- Mutation Routine:
3510 FOR R%=1 TO NR%
3520 IF RND(100) > 8 THEN GOTO 3580
3522 S$=R$(R%)
3525 FOR T%=1 TO 7
3530 R1%=INT(RND(1)*SIZE%+1)
3540 R2%=INT(RND(1)*SIZE%+1)
3560 I%=R1%:J%=R2%:GOSUB 6000:REM SWAP
3565 NEXT T%
3570 R$(R%)=S$
3575 RV(R%)=0
3580 NEXT
3590 REM only swapping at present.
3595 REM needs inversion too.
3600 RETURN
3620 :
4000 REM -- Tidying-up Routine:
4010 RETURN
4020 REM dummy at present.
4040 :
5000 REM -- O/P of results:
5010 PRINT'"Routes are:"
5020 BR=SIZE%*1000
5030 R%=0
5040 FOR I%=1 TO NR%
5050 IF RV(I%)=0 THEN 5100
5060 PRINT I%,RV(I%)
5070 PRINT R$(I%)
5080 IF RV(I%)<BR THEN BR=RV(I%): R%=I%
5100 NEXT
5105 A$=GET$
5110 PRINT
5120 PRINT "Best one is :"
5130 PRINT R$(R%),R%
5131 S$=R$(R%):GOSUB 6400
5133 PRINT"Distance  = ";RV(R%)
5134 A$=GET$
5135 PRINT'"Best ever :"
5136 PRINT B$
5140 S$=B$:GOSUB 6400
5143 PRINT"Distance = ";B
5144 A$=GET$
5145 PRINT
5148 GOSUB 5500 : REM file dump
5150 RETURN
5160 :
5500 REM -- Dump Routine:
5510 INPUT "New Rule File (RETURN if none)", RF$
5520 IF RF$="" THEN RETURN
5530 F%=OPENOUT(RF$)
5540 PRINT #F%, B$, B, SN%
5550 CLOSE#F%
5555 REM only the best.
5560 RETURN
5570 :
6000 REM **** SWAP TWO CHARS IN S$ ****
6040 IF I%>J% THEN T%=I%: I%=J%: J%=T%
6050 X$=MID$(S$,I%,1)
6060 Y$=MID$(S$,J%,1)
6070 S$=LEFT$(S$,I%-1)+Y$+MID$(S$,I%+1)
6080 S$=LEFT$(S$,J%-1)+X$+MID$(S$,J%+1)
6090 RETURN
6100 :
6400 REM **** TRIP ****
6430 PRINT " 0  ";NAME$(SN%)
6440 FOR I%=1 TO LEN(S$)
6450 N%=ASC(MID$(S$,I%,1))-64
6460 IF N%<>SN% THEN PRINT I%;" ";NAME$(N%)
6470 NEXT
6480 PRINT I%;" ";NAME$(SN%)
6490 RETURN
6500 :
8000 REM -- DATA FOR PLANETARY MAP:
8010 DATA HELIOSOL, 1
8020 DATA 2,4, 17,30, 20,80, 0,0
8030 DATA NIKE,2
8040 DATA 1,4, 3,6, 0,0
8050 DATA APHRODITE,3
8060 DATA 2,6, 5,7, 0,0
8070 DATA LUNA, 4
8080 DATA 5,1, 6,10, 0,0
8090 DATA TERRA FIRMA, 5
8100 DATA 3,7, 7,8, 6,8, 4,1, 0,0
8110 DATA DEMON KINGDOM, 6
8120 DATA 5,8, 7,1, 9,12, 8,12, 5,10, 0,0
8130 DATA PHOBIA, 7
8140 DATA 5,8, 9,13, 8,11, 6,1, 0,0
8150 DATA EUREKA, 8
8160 DATA 6,12, 7,11, 9,1, 10,16, 11,25, 0,0
8170 DATA GALILEO, 9
8180 DATA 10,15, 8,1, 6,12, 7,13, 0,0
8190 DATA TITANIUM CITY, 10
8200 DATA 9,15, 12,24, 11,20, 8,16, 0,0
8210 DATA UMBRIA, 11
8220 DATA 8,25, 10,20, 12,17, 13,28, 0,0
8230 DATA TRIDENT, 12
8240 DATA 10,24, 14,22, 13,20, 11,17, 18,2, 0,0
8250 DATA LIMBO, 13
8260 DATA 12,20, 14,1, 16,48, 15,23, 11,28, 0,0
8270 DATA SUNSET STRIP, 14
8280 DATA 12,22, 13,1, 18,25, 0,0
8290 DATA HADES, 15
8300 DATA 13,223, 16,232, 0,0
8310 DATA TROGSTAR BETA, 16
8320 DATA 13,48, 17,64, 15,32, 0,0
8330 DATA MAXIMA CENTAURI, 17
8340 DATA 16,64, 1,30, 19,88, 0,0
8350 DATA POSEIDON, 18
8360 DATA 12,2, 14,25, 13,25, 0,0
8370 DATA ULTIMA THULE, 19
8380 DATA 17,88, 20,96, 0,0
8390 DATA OMEGA SOLARIS, 20
8400 DATA 1,80, 19,95, 0,0
8410 DATA NOWHERE, 0
```

Menschliche Züge

In unserer Serie über vertikale Software sehen wir uns eine Programmfolge an, die Persönlichkeitsmerkmale bewertet und dem Geschäftsmann helfen soll, für den Umgang mit Kollegen und Kunden bessere Strategien zu entwickeln.

Das kalifornische Softwarehaus Human Edge Software hat einen interessanten Schritt in Richtung auf die Künstliche Intelligenz getan. Die Programme von Human Edge sind hochentwickelte kommerzielle Entscheidungshilfen, die große Datenmengen schnell nach bestimmten Kriterien durcharbeiten können, um Handlungsabläufe zu empfehlen. „The Human Edge" ist eine Reihe von vier Programmen – Communications Edge, Sales Edge, Management Edge und Negotiation Edge –, die auf dem IBM und kompatiblen Maschinen läuft. Dieses Paket soll in den vier im Programmnamen angegebenen Bereichen „das individuelle professionelle Können des Anwenders erhöhen". Für den Commodore 64, Apple II und Macintosh gibt es kleinere Versionen (Mind Prober), die mit den gleichen Techniken arbeiten. In diesem Artikel untersuchen wir jedoch das große Programmpaket.

Human Edge gibt an, daß die Entwicklung der Programme zehn Jahre gedauert hat, wobei die Erfahrungen von Verhaltensforschern und Geschäftsexperten eingeflossen sind und neue Techniken wie die Analyse des menschlichen Faktors, Expertensysteme und Entscheidungstheorie eingesetzt wurden.

Die Programme sind leicht zu bedienen und können schon nach etwa einer Stunde Eingewöhnung voll genutzt werden. Alle Module sind menügesteuert und gruppieren sich um ausführliche „Fragebögen", in denen der Anwender seine Zustimmung oder Ablehnung zu einer Reihe durchdacht formulierter Aussagen bekundet. Die Aussagen drehen sich um die wesentlichen Persönlichkeitsmerkmale des Anwenders, seine Verkaufspläne, Kunden, Untergebene, Vorgesetzte und die zu untersuchenden Bereiche.

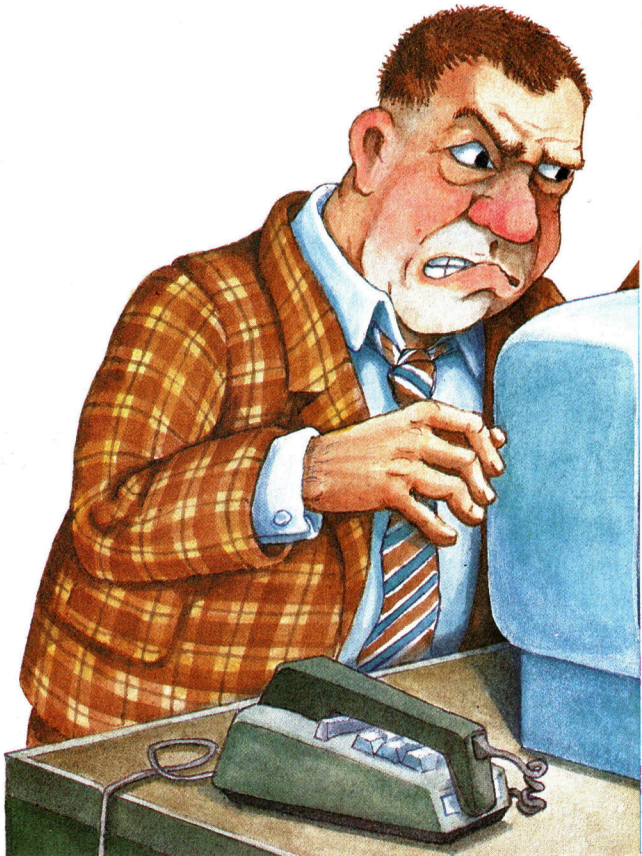
Nachdem das Programm die Antworten bewertet hat, wird ein ausführlicher Bericht mit Empfehlungen für die Handlungsweise vorbereitet. Diese Empfehlungen sagen beispielsweise, wie das Interesse eines neuen Kunden geweckt werden kann, wie die Verkaufsstrategie aussehen soll oder welche Verhandlungstechnik bei Angestellten oder Vorgesetzten am besten ankommt.

Jedes Programm beginnt mit einer Selbstbewertung, die folgende Aussagen enthalten kann: „In den meisten Konferenzen bin ich der bestimmende Faktor", „Ich streite mich mehr als andere", „Ich bin impulsiv" etc. Der Anwender entscheidet, ob die Aussage auf ihn zutrifft, und gibt die entsprechende Antwort ein. Die Selbstbewertung wird sehr gründlich durchgeführt und prüft durch überlappende Fragen die Richtigkeit der Selbsteinschätzung. So werden beispielsweise die Antworten auf „Ich bin impulsiv" und „Manchmal handele ich, ohne vorher nachzudenken" einander gegenübergestellt und auf Übereinstimmung geprüft. Nach Abschluß dieses Teils werden die Antworten auf der Diskette gespeichert. Sie lassen sich jederzeit abrufen und aktualisieren.

Ganz persönlich

Nach der Selbstbeurteilung wird die zu untersuchende Person mit einer Reihe von Eigenschaften bewertet. Wörter wie ängstlich, gesprächig, unabhängig, ehrgeizig, mutig, angeberisch und einfühlsam helfen dem Anwender, seinen Kunden, Vorgesetzten oder Untergebenen zu beschreiben.

Die Selbstbewertung muß nur einmal ausgefüllt werden und läßt sich dann zu jeder ande-



ren „gegnerischen" Eigenschaftenliste in Beziehung setzen. Auf einer Diskette können bis zu acht dieser „Gegnerlisten" untergebracht werden. Nach Abschluß der beiden Listen bewertet das Programm die Antworten und bereitet einen Bericht vor, der die Merkmale von Anwender und „Gegenspieler" miteinander verknüpft.

Der folgende Bericht wurde von Communication Edge erstellt. Die fragende und die untersuchte Person (der Sohn des Anwenders, im Bericht T. genannt) sind echt. In seinem Bericht spricht der Computer den Anwender direkt an:

„In der Kommunikation mit T. müssen Sie Ihre flexible und ruhige Umgangsweise mit Menschen zum Einsatz bringen. T. ist sehr eigen, er ist gern allein und hat nur wenig Interesse an oberflächlichen Gesprächen. Machen Sie sich auf eine zynische oder mißtrauische Haltung gefaßt, wenn Sie ihn auf seine Vorstellungen und Gefühle hin ansprechen. Gehen Sie bei Gesprächen nicht davon aus, daß er Sie sofort versteht. Seien Sie deutlich, knapp und direkt."

„Im Gegensatz zu Ihrer Ausgeglichenheit läßt sich T. leicht verärgern, er kann sogar schon vor Beginn des Gespräches einen ärgerlichen Eindruck machen. Er wird möglicherweise versuchen, Ihnen seine Ansichten aufzuzwingen. Bleiben Sie trotz dieser Haltung freundlich. Stellen Sie sich auf ein unvorhersehbares Verhalten ein. T. kann sich in einem Augenblick impulsiv äußern und im nächsten seine Worte mit großem Bedacht wählen. Versuchen Sie, den Verlauf des Treffens zu steuern. Wiederholen Sie seine Kommentare in Ihren eigenen Worten, um Klarheit und Übereinstimmung zu erreichen."

Obwohl Communication Edge keine Fragen über das Alter gestellt hat, werden Eltern in diesem Bericht leicht die Beschreibung eines Heranwachsenden erkennen.

Das Vokabular dieser Analyse hat Ähnlichkeit mit den Ratgeberspalten von Zeitungen und Persönlichkeitstests: Der Anwender wird mit angenehmen Eigenschaften belegt (ausgeglichen, flexibel, ruhig), während die untersuchte Person weniger positiv beschrieben ist (leicht zu verärgern, unvorhersehbares Verhalten, zynisch und mißtrauisch). Diese Technik soll vermutlich das Selbstvertrauen des Anwenders verstärken. Die Aussagen passen jedenfalls gut zum Trend des amerikanischen Managements, Geschäft als „Krieg" zu verstehen und den Kunden als „Gegner".

Verhandlungsbasis

In Negotiation Edge ist diese Tendenz noch deutlicher, wie diese empfohlenen Strategien zeigen:

SETZEN SIE DAS WISSEN VON T. ZU IHREM VORTEIL EIN
LEGEN SIE T. SCHON FRÜH AUF ZUGESTÄNDNISSE FEST
GEWÖHNEN SIE T. DARAN, "JA" ZU SAGEN
VERSCHLEIERN SIE IHRE DROHUNGEN
STELLEN SIE IHREN GEWINN ALS NIEDRIG DAR
FÜHREN SIE DAS GESPRÄCH IN EINEM POSITIVEN TON

Der Hersteller scheint davon auszugehen, daß die Anwender nur eins der vier Programme erwerben: In jedem muß eine neue Selbstbewertung ausgefüllt werden, deren fast identische Fragen auch fast die gleiche Reihenfolge haben. Für Firmen, die die Programme als Managementinstrumente einsetzen möchten, wäre es jedoch praktischer, wenn die Selbstbewertung gleich für alle vier Module gelten würde. Da die Programme sehr ähnlich sind, genügt es, das billigste – Communication Edge – zu kaufen und die Berichte auf die gewünschten Situationen einzustellen.

Leider haben alle Programme eine große Schwäche: Sie können nicht aus der Erfahrung lernen. Es wäre beispielsweise vorteilhaft, wenn der Anwender seine Erfahrungen mit einer vorgeschlagenen Strategie eingeben könnte und das Programm die neue Strategie dann diesen Daten anpassen würde. Dies wäre besonders für die Bewertung von Personen wichtig, über die anfangs nur wenig bekannt ist. Außerdem lassen sich viele Fragen nicht mit einem einfachen „Ja" oder „Nein" beantworten, und es scheint auch keine Baumstruktur zu geben, mit der eine tiefergehende Beantwortung und Überprüfung der Antworten möglich wäre.

Es hängt wohl von der Einstellung des einzelnen ab, ob er dieser Art der Gestaltung menschlicher Beziehungen zustimmt oder nicht. Die Programme eignen sich am besten zur Vorbereitung eines Verkaufsgesprächs. Solange Computer jedoch noch nicht wirklich „denken" können, sollte man den Ratschlägen nur mit Vorsicht folgen.

The Human Edge
Eine Folge von vier Programmen (auch einzeln erhältlich) für MS-DOS und PC-DOS-Maschinen

Vertrieb:
Thorn EMI Computer Software Distributors, 296 Farnborough Road, Farnborough, Hants GU14 7NF

Autoren:
Human Edge Software

Format:
Diskette

Mind Prober
Für den Commodore 64, Apple II und Macintosh

Vertrieb:
Siehe oben

Autoren:
Siehe oben

Format:
Diskette

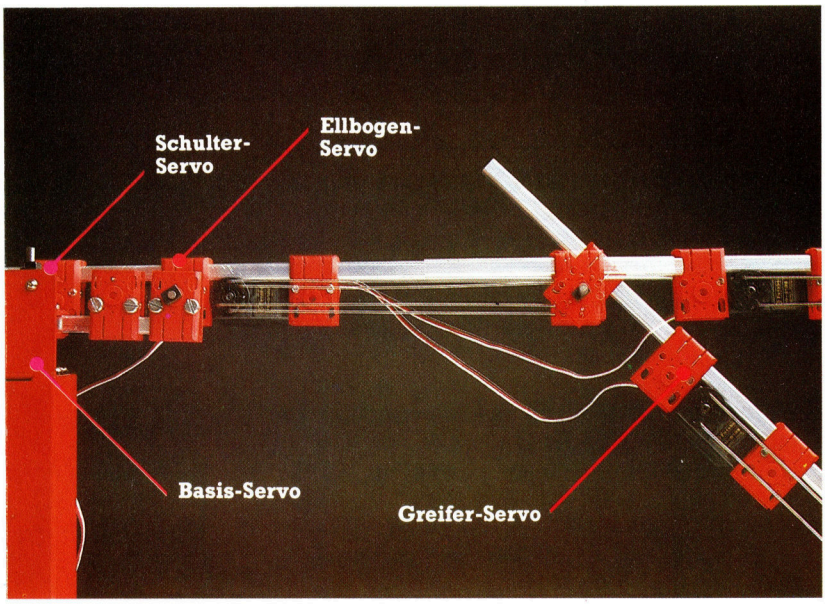
Kraftvoll

Sie können Ihrem Computer nach Anschluß eines Servomotors am User Port eine Vielzahl von Steuerungsaufgaben übertragen. Dabei kommt das in den früheren Kursabschnitten entwickelte Buffer-System wieder zum Einsatz.

Wir sprechen über drei Typen von Elektromotoren – Gleichstrom-, Schritt- und Servomotoren. Jeder normale Gleichstrommotor kann über den Computer angesteuert werden. Seine Drehzahl geht allerdings zurück, wenn irgend etwas seinen Lauf belastet. Durch solche Ungenauigkeiten läßt sich seine Position vom Computer her nicht exakt überwachen.

Einfacher ist das bei Schrittmotoren, die durch einen Impuls nur um einen festen Winkel drehen (zum Beispiel 7,5 Grad). Solange der Motor nicht durch Überlastung stehenbleibt, kennt der Computer auch die Achsenposition. Schrittmotoren werden häufig für Computersteuerungen verwendet – etwa bei Robotarmen, Drehbänken und Sortiermaschinen. Allerdings muß der Steuerimpuls gleichzeitig die Energie für den Motor liefern, der spezielle Treiber braucht und nicht gerade billig ist.

Digital-Servomotoren aus dem Modellbau-Laden sind im Vergleich preiswert und leichter erhältlich. Ihre Größe reicht von den Maßen einer halben Streichholzschachtel bis zu zehnmal voluminöseren Motorvarianten. Man bekommt sie zu Preisen ab etwa 25 Mark. Einige dieser Motoren sind recht kräftig – sie erreichen Drehmomente, die ein Mensch etwa mit einem kräftigen Schraubendreher erzeugen kann. Auch die billigsten und einfachsten Servomotoren lassen sich für den Selbstbau von kleinen Robot-Armen verwenden.



Der Robot-Arm „Beasty" bewegt sich mit Hilfe von drei Servomotoren (Basis, Schulter- und Ellbogenrotation). Zum Antrieb eines Greifers kann ein vierter Motor eingebaut werden. Servomotoren eignen sich für Robotarme, weil sie sehr exakt bewegt und in jeder Position fixiert werden können.

Kleine Servomotoren wie der Futuba FP-S126 oder ein Axoms AS-1 bestehen aus einem Rückkopplungs-Potentiometer und einer Art Hebel, der über mehrere Getrieberäder mit dem winzigen Gleichstrommotor verbunden ist. Die Motoren eignen sich sehr gut für die rückgekoppelte Verschaltung mit einem Computer, weil im Motorgehäuse die erforderliche Elektronik nebst einem integrierten Steuerchip bereits eingebaut ist.

Typische Digital-Servomotoren für den Modellbau werden mit fünf Volt betrieben. Der Drehwinkel des Antriebshebels wird mit einer Extra-Leitung gesteuert. Ein Stromimpuls von 1/1000 Sekunde dreht ihn um einen bestimmten Winkel. Ein doppelt so langer Impuls treibt den Hebel in die Gegenrichtung. Der Drehwinkel ist der Impulslänge proportional.

Die Stellung halten

Nach dem Impuls bleibt der Motor nur für weitere 20 Millisekunden aktiv, danach geht er auf die Ausgangsposition zurück. Um den Hebel in einer bestimmten Stellung zu halten, muß der Steuerimpuls mit einer Frequenz von circa 50 Hz wiederholt werden.

Üblicherweise setzt man Servomotoren ein, um Hebel oder Gestänge zu bewegen. Aber auch andere Steuerungsarten sind möglich. Dazu wird das Potentiometer vom Getriebe abgekoppelt und auf Mittelstellung gebracht. Dann läuft der Motor mit kontinuierlicher, von der Impulslänge abhängiger Drehzahl.

Beim direkten Anschluß von Servomotoren am User Port können Fehler der Verkabelung zu Schäden am Computer führen. Sicherheit schafft hier das bereits früher im Selbstbau-Kurs entwickelte Buffersystem. Das Kabel für die Motor-Stromversorgung wird mit einer der positiven (roten) Buchsen des Niedervolt-Ausgangs verbunden. Die Masseleitung schließt man an die negative (schwarze) Buchse an.

Wenn die Steuerleitung des Motors mit der Datenleitung 0 des User Ports verbunden ist, können Sie über diesen Kanal den Motor mit entsprechenden Impulsen beliebig steuern. Für einen Impuls wird Kanal 0 auf fünf Volt gebracht, indem die binäre 0 durch eine 1 ersetzt wird. Eine Zählschleife sorgt dafür, daß nach der gewünschten Zeit wieder 0 Volt auf dem User-Port-Anschluß liegen.

Sowohl der Acorn B als auch der Commodore 64 haben eine veränderbare User-Port-Schaltung. Da der Port für die Ein- und Ausgabe dient, müssen wir den verwendeten Anschluß erst in die richtige Betriebsart (in diesem Fall Ausgabe) bringen. Bei beiden Rechnern wird das durch POKEn ins Datenrichtungs-Register bewerkstelligt.

Wie wird nun ein Impuls über die Datenleitung des User Ports geschickt? Der hexadezimale Wert 88 (entspricht dezimal 136 oder binär 10001000) wird zum User Port gePOKEt. Die Spannungen an den Pins sind dann entsprechend 5V,0V,0V0V,5V,0V,0V,0V. Dieser Zustand des User Ports bleibt bis zur nächsten Änderung erhalten. Ein einzelner Impuls würde also durch POKEn der Datenleitung 0 mit hex 00, hex 88, hex 00 entstehen.

Wir müssen die Maschinensprache einsetzen, damit der Impuls schnell genug produziert wird. Der Algorithmus für einen Impuls sieht im einzelnen so aus:

1) Lege den Winkel durch Speichern eines Wertes zwischen 0 und 255 fest. Wert vom BASIC aus in einem Byte (ANGLE) speichern.
2) Setze Datenleitung 0 auf High (5V), um den Impuls zu starten.
3) Verzögere eine Millisekunde und dekrementiere Zähler.
4) Zweite Schleife mit Länge zwischen 0 und 1 Millisekunde durchlaufen. Anfangswert des Zählindex (und damit die Anzahl der Schleifendurchläufe) ist diesmal der Wert ANGLE.
5) Datenleitung 0 auf Low (0 Volt) setzen, damit Impuls abschließen.

Wenn ANGLE=0 ist, dauert der Impuls eine Millisekunde, bei ANGLE=128 eineinhalb Millisekunden (Acorn B und C 64). Der Hebel am Motor bewegt sich dadurch in eine Mittelposition.

Zum Steuern eines Servomotors ist ein einziger Impuls natürlich nicht genug. Der Motor braucht circa alle 20 Millisekunden einen Impuls, damit er in der neuen Position stehenbleibt. Dafür gibt es zwei Verfahren:

1) Eine Warteschleife zwischen den beiden Impulsen – damit ist der Rechner allerdings für andere Tätigkeiten blockiert.
2) Durch „Interrupts", mit denen der Rechner mehrere Programme fast gleichzeitig ausführen kann. Ein zweites Programm kann beispielsweise die Bewegungsrichtung steuern.

Interrupt-Anschlüsse

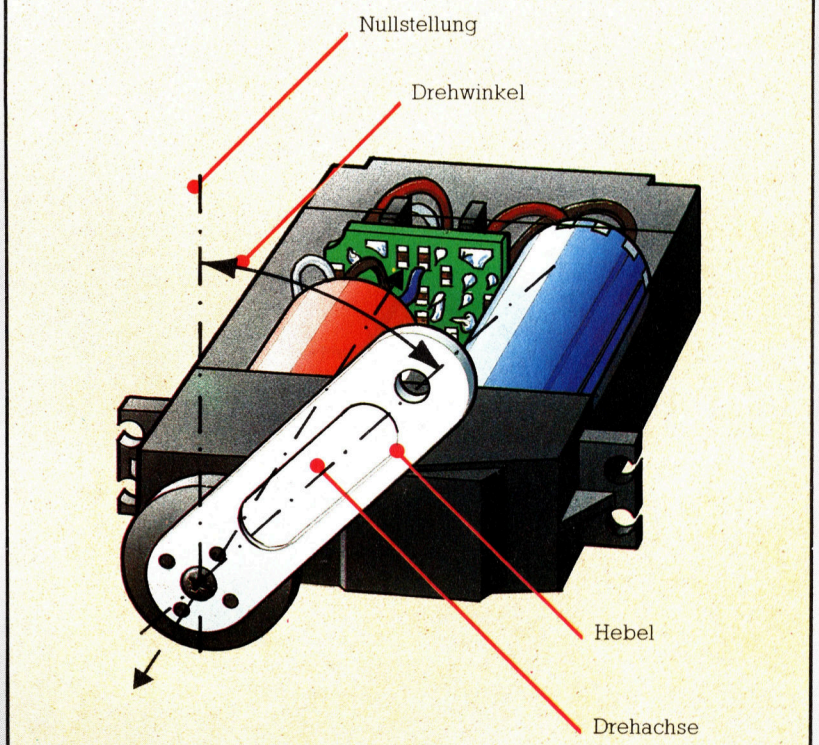
Der Acorn B und auch der Commodore C 64 sind mit Prozessoren der Serie 6500 ausgestattet, die über die Interrupt-Anschlüsse NMI und IRQ verfügen. Für unsere Zeitsteuerungen verwenden wir den IRQ. Wenn dort ein Impuls ankommt, unterbricht der Prozessor seine momentane Tätigkeit und verzweigt zu einem Interrupt-Steuerprogramm. Nach dessen Ausführung kehrt er wieder zu der vorherigen Aufgabe zurück (RTI).

Acorn B und C 64 nutzen Interrupts für das Betriebssystem. Der Acorn erzeugt pro Sekunde 100 Interrupts, beim C 64 sind es 60. Bei jedem dieser zeitgesteuerten Interrupts werden die Systemzähler auf den neuesten Stand gebracht und die Tastatur abgefragt.

Bei beiden Rechnern können wir die System-Interrupts für die Impulserzeugung nutzen. Im C 64 müssen die Interrupts durch Veränderung des Interrupt-Vektors modifiziert werden. Dieser Vektor umfaßt zwei Bytes und enthält die Adresse des Interrupt-Steuerprogramms. Wir können die Werte verändern und den Vektor auf den Anfang unserer Impuls-Routine zeigen lassen. An deren Ende wird der Prozessor zurückgeführt und erzeugt so bei jedem Interrupt einen Impuls – 60mal in der Sekunde.

Auf der folgenden Seite stehen die BASIC- und Assemblerversion eines Servomotor-Steuerprogramms für den C 64. Das entsprechende Programm für den Acorn B finden Sie im nächsten Heft, in dem wir auch auf die simultane Ansteuerung mehrerer Motoren eingehen.

Steuerungsprinzip



Wie weit sich der Hebel aus seiner Ausgangsposition bewegt, hängt von der Dauer des Steuerimpulses ab, den der angeschlossene Computer liefert. Wenn der Impuls etwa alle 20 Millisekunden wiederholt wird, bleibt der Hebel in seiner Position stehen. Diese Möglichkeit zum Festhalten einer Position ist ideal. Die meisten Heimcomputer erzeugen alle 10 bis 20 Millisekunden einen Interrupt. Die Impulswiederholung kann daher unabhängig vom laufenden BASIC-Programm durch den Einbau von Maschinenbefehlen ins Betriebssystem des Rechners ausgeführt werden. Ein großes Plus des Servomotors ist, daß für die Steuerung nur eine User-Port-Datenleitung gebraucht wird.

Servosteuerung für den Commodore 64

Der erste Teil des Quellprogramms für die Steuerung eines Servomotors mit dem C 64 ändert die Interrupt-Vektoren (Adresse 788 und 789). Während der Modifikation müssen eventuelle Interrupts unterbunden (SEI) werden, und mit CLI lassen sie sich reaktivieren. Der restliche Programmteil enthält die Interrupt-Steuerung für einen einzelnen Servomotor.

Das BASIC-Aufrufprogramm gibt an, wie die Maschinenbefehle eingegeben werden, steuert den User Port und POKEt die Werte auf Speicherplatz $3000 (12288), je nach Tastendruck (von 1—9). Der am User Port angeschlossene Servomotor sollte dann die der gewählten Ziffer entsprechende Position einnehmen. Mit E wird das Programm beendet.

Falls Sie über einen Assembler verfügen, geben Sie das Quellprogramm ein und assemblieren es in ein Objektfile, das stückweise mit dem Aufrufprogramm geladen wird. Alternativ können Sie auch das BASIC-Ladeprogramm für das Maschinenprogramm eingeben und den Code damit in den Speicher übertragen. Vor dem Laden des Aufrufprogramms muß NEW eingegeben werden. Wenn Sie das BASIC-Ladeprogramm einsetzen, fallen Zeile 30 und 40 weg.

Vorsicht: Beachten Sie, daß sich der Rechner bei Verwendung von Interrupts in einem Programm sehr leicht „aufhängt". Sicherheitshalber sollten Sie das Programm speichern, bevor es mit RUN gestartet wird!

Quellprogramm

```
;++++++++++++++++++++++
;++++++++++++++++++++++
;++                  ++
;++ CBM SINGLE SERVO ++
;++      DRIVER      ++
;++                  ++
;++++++++++++++++++++++
;++++++++++++++++++++++
;
PORT = 56579 ; USER PORT DATA REG
ANGLE=12288  ; ANGLE VALUE LOCATION
;
*=$0334
;
      SEI           ;INTERRUPTS OFF
      LDA $0314     ;EXXISTING IRQ VECOTOR
      LDX $03C4
      STA $03C4
      STA $0314
      LDA $0315
      LDX $03C5
      STA $03C5
      STX $0315

;     CLI           ;INTERRUPTS BACK ON
      RTS

;
;++++ EVENT HANDLER ++++
;
      PHP
      PHA           ;SAVE REGISTERS
      TYA           ;ON STACK
      PHA
      TXA
      PHA
      LDA #$FF
      STA PORT
      LDY #$FF

LOOP  DEY           ;DELAY LOOP
      BNE LOOP      ;APPROX 1MSEC

;     LDY ANGLE

LOOP1 DEY           ;COUNT OUT PULSE
      BNE LOOP1

;     LDA #$00
      STA PORT      ;ZERO DATA REGISTER

;
      PLA           ;RESTORE REGISTER
      TAX           ;VALUES
      PLA
      TAY
      PLA
      PLP

;     JMP $EA31
```

BASIC-Ladeprogramm

```
10 REM **** BASIC LOADER FOR ***
*
20 REM **** SINGLE SERVO PROG***
*
30 :
40 FOR I=820 TO 882
50 READ A:POKE I,A
60 CC=CC+A
70 NEXT I
80 READ CS:IF CC<>CS THENPRINT
"CHECKSUM ERROR":STOP
100 DATA120,173,20,3,174,196,3,1
41,196
110 DATA3,141,20,3,173,21,3,174,
197,3
120 DATA141,197,3,142,21,3,88,96
,8,72
130 DATA152,72,138,72,169,255,14
1,3
140 DATA221,160,255,136,208,253,
172,0
150 DATA48,136,208,253,169,0,141
,3,221
160 DATA104,170,104,168,104,40,7
6,49
170 DATA234
180 DATA7170:REM*CHECKSUM*
```

BASIC-Aufrufprogramm

```
10 REM **** SINGLE SERVO ****
20 :
30 DN=8:REM IF CASSETTE THEN
DN=1
40 IF A=0 THEN A=1:LOAD"SINGSERV
.HEX",8,1
50 POKE 964,79: POKE965,3:REM
POINT TO IRQ HANDLER
60 DDR=56577: POKE DDR,255:
REM ALL OUTPUT
70 MC=820:SYS MC: REM SET IRQ
VECTOR
80 POKE 53265,PEEK(53265)AND239:
REM BLANK SCREEN
90 :
100 GET K$:IF K$="" THEN100:REM
AWAIT KEYPRESS
110 REM ** ALTER MOTOR POSITION
**
120 IF ASC(K$)>48 AND ASC(K$)<58
 THEN POKE 12288,VAL(K$)*20
130 IF K$<>"E" THEN 80:REM 'E'
TO EXIT
140 END
```

Raketenmann

Mit „Jet Pac" debütierte das Software-Haus Ultimate, Play the Game. Mit diesem ursprünglich für den Spectrum geschriebenen Programm wurde damals ein neuer Standard für Sinclair-Rechner eingesetzt. Es avancierte prompt zum Bestseller. Diesem Erfolg schloß sich auch die Version für den VC 20 an.

Wie so oft, wird auf dem Cassetteneinleger ein Szenario dargestellt, das das Spiel spannender erscheinen läßt, als es in Wirklichkeit ist. In Ihrer Eigenschaft als Testpilot der „Acme Interstellar Transport Company" besteht Ihre Aufgabe darin, durch die Galaxis zu reisen und auf ausgesuchten Planeten Raumschiffe zusammenzubauen. Dabei sind gleichzeitig Gold und edles Gestein zu sammeln. Die Raumschiffmontage wird durch den „Hydrovac"-Düsenantrieb erleichtert, mit dem Gegenstände angehoben werden, so daß Sie auch schwere Dinge sehr leicht handhaben können. Zusätzlich ist man mit einem Quad Photon Laser Phaser bewaffnet, der zur Abwehr der Außerirdischen verwendet wird, die sich über die Umweltverschmutzung ihres Planeten bei Ihnen handgreiflich „beklagen" wollen.

Welten und Planeten

Die Spielbeschreibung erweckt den Eindruck, als würden Sie fantastische Reisen durch die unterschiedlichsten Welten erleben. Die Bildschirmwirklichkeit ist – wie stets – bescheidener. Die Planeten gleichen sich – vor allem bei der Landung – verblüffend. Doch die Außerirdischen machen das Manko wieder wett. Jeder Planet wird nur von einer Spezies bewohnt. Für andere wäre auch wenig Platz, da sich die Außerirdischen wie Kaninchen zu vermehren scheinen. Die Arten reichen von fliegenden Untertassen zu aufsteigenden Ballons. Eines haben sie alle gemeinsam: Eine einzige Berührung mit ihnen bedeutet sofortigen Tod.

Auf der Oberfläche der Planeten sind jeweils drei Raumschiffteile verteilt. Sie müssen zusammengefügt werden, bevor es weitergeht. Schraubenzieher und dergleichen sind für die Arbeit nicht erforderlich: Man setzt die Komponenten lediglich aufeinander, und nach und nach entsteht das Raumschiff.

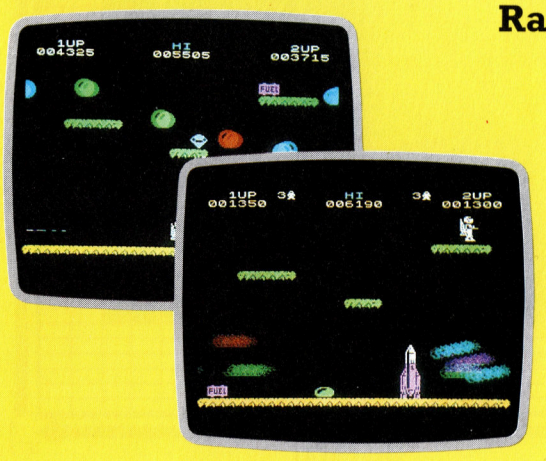
Erst die Außerirdischen verleihen dem Spiel den Reiz. Auf den ersten Blick wirken sie bedrohlich, da sie in Richtung des Bildschirm-Ich fliegen. Doch bald schon merkt man, daß sie auf vorgegebenen Bahnen fliegen, von zufälligen Startpunkten ausgehend. Die erste Welle der Angreifer sinkt langsam herab. – Reichlich Zeit für den überlegten Laser-Einsatz. Die zweite Welle besteht aus hüpfenden Bällen, die im Zickzack über den Bildschirm flitzen. Bei dieser Mischung aus festgelegten Wegen und Zufallsbewegung sind Überlegung und schnelle Reflexe gefordert.

Die Steuerung erfolgt mit dem Joystick oder über die Tastatur. Zwei Tasten auf der unteren Reihe werden für die Bewegung nach links und rechts benutzt. Die Tasten der zweiten Reihe lösen das Laserfeuer aus, nach Wunsch auch Dauerfeuer. Die Tastenreihe über dem „Auslöser" ermöglicht das „Aufsteigen", die darüber das „Schweben". Drückt man keine Taste, wird man durch die Schwerkraft auf den Planeten heruntergezogen.

Die Grafik ist ausgezeichnet. Der Testpilot wurde hervorragend gestaltet, ebenso der Hydrovac Jet Pac, der bei Inbetriebnahme Wolken als Zeichen der Aktivierung ausstößt.

Die Spectrum- und die VC-20-Versionen sind identisch, wenngleich das Bildschirmformat des VC 20 den Testpiloten etwas übergewichtig erscheinen läßt. In beiden Versionen ist Jet Pac ein Spiel mit hohem Unterhaltungswert.

Jet Pac: für Spectrum 16/48 K und für VC 20 (8 K)
Hersteller: Ashby Computers and Graphics Ltd., The Green, Ashby de la Zouch, Leicestershire LE6 5JU
Programm: Ultimate, Play The Game
Joysticks: Kempston, Competition-Pro (Spectrum), fast alle Commodore-kompatiblen.
Format: Cassette



Raketenmontage

Die Aufgabe des Jet-Pac-Piloten besteht darin, auf unterschiedlichen Planeten Raumschiffe zusammenzubauen. Dabei drohen Angriffe von Außerirdischen. Als Ausgleich werden Gold und Juwelen geboten, die vom Himmel fallen.

Jet Pac auf dem Spectrum

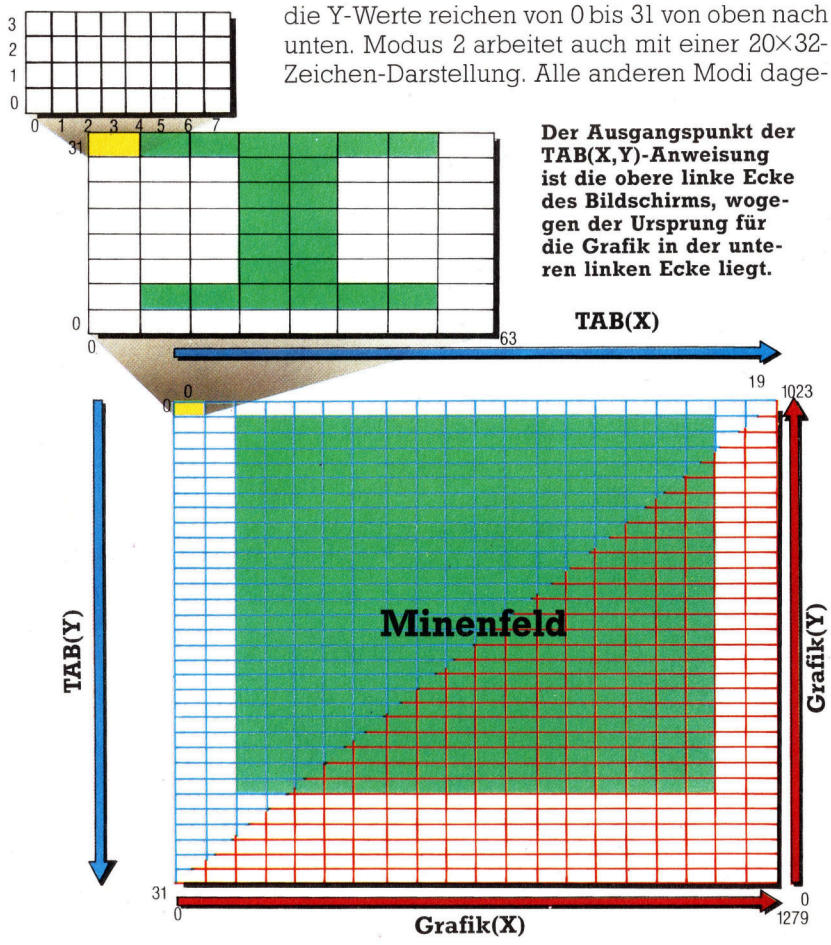
Im Blickfeld

Dies ist der zweite Teil eines Projekts zur Entwicklung eines Grafikspiels für den Acorn B und den Electron. Wir befassen uns hier mit hochauflösender Grafik, den internen Uhren des Acorn sowie den restlichen Prozeduren zur Darstellung des Spielfelds.

Im ersten Teil wurde im Modus 5 ein Gebiet als Minenfeld definiert. Auch die Umrisse für die Minen, das Suchgerät und den Assistenten wurden festgelegt. Danach entwickelten wir Routinen zum Auslegen der Minen und stellten das Suchgerät und den Assistenten an den entsprechenden Startpositionen dar. Um die Darstellung interessanter zu gestalten, kann man nun einen Rand um das Minenfeld zeichnen. Am einfachsten geht das mit den Befehlen MOVE und DRAW.

Wenn man auf dem Acorn hochauflösende Grafik mit Zeichen in geringerer Auflösung kombiniert, tritt ein Problem auf, da die verschiedenen Darstellungen mit unterschiedlichen Koordinaten arbeiten. Die Darstellung in geringer Auflösung wurde bereits erläutert. In diesem System ist der Ausgangspunkt (X und Y = 0) die obere linke Ecke. Die X-Werte erhöhen sich hier von 0 auf 19 von links nach rechts, und die Y-Werte reichen von 0 bis 31 von oben nach unten. Modus 2 arbeitet auch mit einer 20×32-Zeichen-Darstellung. Alle anderen Modi dagegen zeigen eine unterschiedliche Anzahl von Zeichen und verwenden deshalb verschiedene Koordinaten für jedes Zeichen.

Die Zeichen werden auf einem 8 × 8-Raster definiert. In Modus 5 entsteht ein Rechteck mit 64 hochauflösenden Punkten Breite und 32 Punkten Höhe. Jedes Zeichen muß somit aus acht Pixeln in der Breite und vier Pixeln in der Höhe aufgebaut sein. Spricht man mit den Befehlen PLOT oder DRAW einen Punkt eines Pixels an, wird das gesamte Pixel aktiviert – PLOT 7,3 entspricht PLOT 5,2.



Der Ausgangspunkt der TAB(X,Y)-Anweisung ist die obere linke Ecke des Bildschirms, wogegen der Ursprung für die Grafik in der unteren linken Ecke liegt.

Die acht Grafikmodi des Acorn bieten drei unterschiedliche Auflösungen (640 auf 256, 320 auf 256 und 160 auf 256) und nutzen dasselbe Koordinatensystem. Es behandelt den Bildschirm, als hätte er eine Auflösung von 1280 auf 1024 Pixel. Alle Koordinaten werden als Zahlen von 0 bis 1279 in der Horizontalen und 0 bis 1023 in der Vertikalen auf dem Bildschirm angegeben. Das BASIC wandelt diese Zahlen dann in die Werte des jeweils verwendeten Modus um.

Das Kombinieren von Zeichen und Grafiken ist etwas komplizierter, da der Ursprung der Grafik in der unteren linken und nicht – wie im Zeichen-Koordinatensystem – in der oberen rechten Ecke des Bildschirms liegt.

Die Auflösung von 160 × 256 Pixel im Modus 5 hat direkten Zusammenhang mit der Anzahl der darstellbaren Zeichen. Jedes Zeichen wird auf einem 8 × 8-Raster definiert. Da auf dem Bildschirm horizontal 20 Zeichen darstellbar sind, beträgt die Anzahl der Pixel 8 × 20=160. Vertikal sind es 32 × 8=256 Pixel. Bei hoher Auflösung liegen 1280 verschiedene Koordinaten auf der X-Achse. Dividiert man diese Zahl durch die Pixel-Anzahl auf der X-Achse, erhält man 1280/160=8. Entsprechend erhält man bei Division der hochauflösenden Koordinaten durch die Pixel auf der Y-Achse das Ergebnis 1024:256=4. Ein Pixel kann also eingeschaltet werden, indem an einer Position des 1280/1024-Koordinatensystems ein Punkt gezeichnet wird. Aus dem Bild ist zu ersehen, wie ein Koordinatenbereich verwendet werden kann, um ein Pixel ein- oder auszuschalten.

Wir können jetzt Zeichen-Positionen hochauflösenden Koordinaten zuordnen. Auf der horizontalen Achse entspricht eine Zeichenbreite 64 Einheiten (8 × 8). Vertikal sind vier Einheiten gleich einem Pixel, vorausgesetzt, daß die Höhe jedes Zeichens 32 Einheiten entspricht. Die Grenzen des Bildschirms können nun mit hochauflösenden Koordinaten für die MOVE- und DRAW-Befehle umschrieben werden. Betrachten Sie hierzu unser Bild.

Nun lassen sich auch die Koordinaten der unteren linken und oberen rechten Ecke des Minenfeldes berechnen (alle anderen Rand-Koordinaten stehen in Relation zu diesen Anfangspunkten).

Die nachfolgende Prozedur zeichnet einen Rand um den definierten Bereich. GCOL 0,1 setzt die logische Farbe, die für Grafiken verwendet wird. Die erste Zahl definiert die Zeichnungsart, die später näher besprochen wird. Die zweite Zahl gibt die Farbe vor. Im Modus 5 ist die logische Farbe 1 normalerweise Rot. Die MOVE-Befehle bewegen den Grafik-Cursor vom Ursprungspunkt in die untere linke Ecke des Randes. Die darauffolgenden DRAW-Befehle zeichnen gerade Linien von der derzeitigen Bildschirmposition bis zum jeweils spezifizierten Punkt.

```
2470DEF PROCdraw_border
2480GCOL 0,1
2490MOVE 120,188
2500DRAW 120,992
2510DRAW 1152,992
2520DRAW 1152,188
2530DRAW 120,188
2540ENDPROC
```

Die Zeit läuft

Die Acorn-Computer besitzen eine interne Uhr, auf die durch die reservierte Variable TIME zugegriffen werden kann. Sobald der Computer den Wert von TIME ausgeben soll, zeigt er eine Zahl an, die der Zeit seit Rücksetzung der Variablen auf Null in Hundertstelsekunden entspricht. Die Prozedur „set-time" gibt das Wort „TIME" sowie den Anfangswert aus und setzt die Variable TIME auf Null. Sie wird während der Set-Up-Routine aufgerufen und startet die Stoppuhr für das Spiel.

```
2640DEF PROCset_time
2650PRINTTAB(2,27)"Time      02:00"
2660TIME=0
2670ENDPROC
```

In der Hauptschleife des Programms muß die angezeigte Spielzeit ständig aktualisiert werden. Eine Anzeige in Sekunden wäre einfach: Man müßte lediglich die Variable TIME durch 100 dividieren, um den Wert in Sekunden umzuwandeln, und ihn dann darstellen lassen. Eine Anzeige in Minuten und Sekunden ist jedoch auch möglich. Hierzu werden die Befehle DIV und MOD verwendet. TIME DIV 100 würde die Anzahl der Sekunden als ganze Zahl ausgeben. (TIME DIV 100) MOD 60 zählt von 0 bis 59 und beginnt wieder bei 0. Der Grund dafür ist, daß der Befehl MOD 60 den Wert der Restzahl nach einer Division durch 60 ausgibt. 63/60 ergibt beispielsweise das Ergebnis 1 mit einem Restwert 3. Demzufolge resultiert aus (63/60) MOD 60 der Wert 3. Die Minuten können auf ähnliche Weise errechnet und dargestellt werden, wenn Sie (TIME DIV 6000) MOD 60 verwenden.

Mit der folgenden Prozedur kann die Zeitangabe ständig aktualisiert werden:

```
2900DEF PROCupdate_time
2910sec$=STR$(((12100-TIME) DIV 100)MOD 60)
2920min$=STR$(((12100-TIME) DIV 6000)MOD 60)
2930REM ** ADD LEADING ZEROS **
2940sec$=LEFT$(zero$,2-LEN(sec$))+sec$
2950min$=LEFT$(zero$,2-LEN(min$))+min$
2960time$=min$+":"+sec$
2970PRINTTAB(11,27);time$
2980ENDPROC
```

Wie aus dieser Prozedur ersichtlich, sind wir bereits einen Schritt weitergegangen. Die Zeitangabe wird in Minuten und Sekunden dargestellt und sogar von 2 Minuten auf 0 zurückgezählt. Zusätzlich wurde eine Routine eingebaut, die sicherstellt, daß die Anzeige für Minuten und Sekunden immer zweistellig erfolgt.

Für die Initialisierung sind zwei weitere Prozeduren notwendig. Im Spielverlauf hat jeder Teilnehmer vier Leben. Daher wird eine Anzeige am unteren Bildschirmrand benötigt, wie viele Leben noch zur Verfügung stehen. Anfangs werden drei Leben angegeben, dargestellt durch drei der bereits definierten „Assistenten"-Zeichen. Die count-Variable berechnet die Anzahl der bereits verlorenen Leben.

```
2690DEF PROCset_men
2700men$=CHR$(226)+CHR$(226)+CHR$(226)
2710count=1
2720COLOUR 1
2730PRINTTAB(2,30);men$
2740COLOUR 2
2750ENDPROC
```

Die fertige „set-up"-Prozedur initialisiert die Wertungen und zeigt diese auf dem Bildschirm an. Der „hiscore$"-Wert wird nicht in dieser Prozedur festgelegt, da sie beim Spielstart jedesmal neu aufgerufen wird. Statt dessen setzen wir diesen Anfangswert schon am Anfang des Programms.

```
2770DEF PROCset_score
2780score=0:score$="00000"
2790PRINTTAB(2,28)"Score     00000"
2800PRINTTAB(2,29)"Hi score ";hi_score$
2810ENDPROC
```

Da nun die Prozeduren zur Initialisierung des Programms entwickelt sind, können wir einen Schritt weitergehen und eine Prozedur schreiben, die alle anderen verwaltet. Im letzten Artikel hatten wir alle Prozeduren mittels eines kurzen Hauptprogramms aufgerufen. Löschen Sie nun diese Programmzeilen, und fügen Sie die folgenden ein:

```
1880DEF PROCsetup
1890COLOUR 2
1900end_flag=0
1910PROCinitialise_variables
1920PROCdefine_characters
1940PROClay_mines(40)
1950PROCdraw_border
1960PROCset_time
1970PROCset_score
1980PROCset_men
1990PROCposition_chars
2000ENDPROC
```

Nun sind wir an dem Punkt, an dem wir ein kurzes Hauptprogramm zum Aufruf der setup-Prozedur und zum Aktualisieren der Zeit mit einer REPEAT. . .UNTIL-Schleife schreiben können. Integrieren Sie die folgenden Zeilen in das Programm:

```
10 REM **** CALLING PROGRAM ****
20 hiscore$="00000"
30 PROCsetup
40 REPEAT
50 PROCupdate time
60 UNTIL TIME>12099
70 END
```

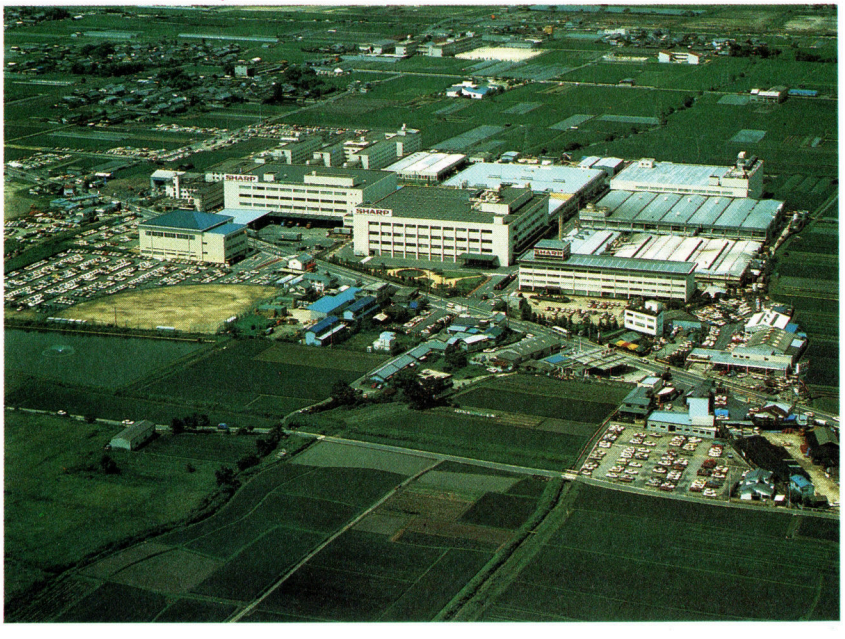
Die Weltfirma

Die japanische Firmengruppe Sharp zeichnet sich durch eine umfangreiche Produktpalette aus, die neben Transistoren, Computern und Industrierobotern auch Mikrowellenherde umfaßt. Im Jahre 1983 stieg die Firma auch in den europäischen Heimcomputermarkt ein.

Die Sharp Corporation beteiligte sich seit jeher an technologischen Zukunftsprojekten. Dabei wirkte ihr erstes erfolgreiches Produkt eher alltäglich – der „Ever Sharp"-Drehbleistift. Sein Erfinder Tokuji Hayakawa gründete die Firma 1915 ausschließlich für die Herstellung seines Schreibgeräts. Dabei blieb es allerdings nicht lange. 1925 brachte Sharp ein Detektorradio auf den Markt. Der Konsumgütermarkt – speziell Fernsehgeräte – wurde für das Unternehmen nach dem Zweiten Weltkrieg immer wichtiger. Mitte der 60er Jahre erfolgte mit einer Serie von Tischrechnern der Einstieg in den kommerziellen Bereich. Heute ist Sharp ein multinationaler Konzern mit sechs Unternehmensbereichen, 34 Produktionsstätten in 30 Ländern und circa 45 000 Mitarbeitern.

Der erste Sharp-Computer MZ80K wurde ab 1981 in Europa angeboten. Bereits ein Jahr später folgten die Rechner MZ80A und MZ80B. Die Geräte waren zwar für den kommerziellen Einsatz entwickelt, fanden aber auch unter den Heimcomputer-Freunden treue Anhänger. Alle Modelle hatten einen eingebauten Monitor und ein Cassettenlaufwerk. Die Sharp-Rechner waren „Clean Machines" – so genannt, weil sie keine Sprache im ROM hatten. Das war vorteilhaft, da sich so unterschiedliche Sprachen und Betriebssysteme (einschließlich CP/M) problemlos von Cassette laden lassen.

Die Ergebnisse der Forschung aus dem firmeneigenen Technologiezentrum werden an die Abteilung „Industriegeräte" im japanischen Yamato-Koriyama-shi bei Nara weitergegeben (Bild unten). Hier werden alle Sharp-Produkte – einschließlich Taschenrechner und Heimcomputer – zur Produktionsreife gebracht.



In den folgenden Jahren hat Sharp sein Angebot stark erweitert: Zuerst mit dem Profi-Rechner MZ-3541, später durch den Pocket-Computer PC-1500. Der Erfolg des PC-1500 führte zur Einführung des noch kompakteren PC-2500 – ebenfalls erfolgreich.

Bei den Heimcomputern ist Sharp mit den MZ700- und MZ800-Serien vertreten. Die in Europa verkauften Modelle stechen besonders durch ihre leistungsstarke Grafik heraus. Der dafür verwendete Speicherplatz enthält in der „Nippon-Version" lediglich den umfangreichen japanischen Zeichensatz. Die Geräte verfügen je nach Typ über eingebaute Cassettenlaufwerke und Vierfarb-Plotter.

Weiteres Wachstum

Im Mai 1984 erschien mit dem PC-1500A ein verbessertes Modell des PC-1500. Das Gerät ist mit 8,9 KByte RAM ausgestattet und kann bis 24 KByte aufgerüstet werden. Nicht lange danach erschien der PC-1350, ein Pocketcomputer mit einer vierzeiligen LCD-Anzeige und Grafikmöglichkeiten.

Auf die Frage nach weiteren Plänen sagt Rod Goodier, Verkaufsleiter in England: „Interessante Möglichkeiten sehen wir bei den Pocket-Computern. In diesem Bereich möchten wir sehr gern expandieren." Aber auch der Heimcomputer-Markt soll nicht zurückstehen: „Mit der MZ700-Serie haben wir einen guten Start gehabt. Wir werden auf jeden Fall in diesem Markt bleiben!" Der englische Unternehmenssprecher Peter Fletcher erklärt: „Unsere Abteilung für Büro- und Heimcomputer ist relativ jung. In England erzielt sie bisher nur 25% vom Umsatz, wir hoffen aber auf ein Wachstum des Anteils am Gesamtergebnis der Firma."

Sharp erweitert mittlerweile seine Produktionskapazität in Europa. In Wrexham/England entstand für mehr als 40 Millionen DM eine neue Fabrik, in der Videorecorder hergestellt werden. Das ehrgeizig gesteckte Ziel liegt für 1985 bei immerhin 60 000 Geräten, die in ganz Europa verkauft werden.

Auf die Frage, ob Sharp sich am MSX-Standard der anderen japanischen Hersteller beteiligen wird, antwortet Rod Goodier: „Natürlich haben auch wir ein MSX-Gerät entwickelt. Es steht aber noch nicht fest, ob und wann es auf den Markt kommt."

Fachwörter von A bis Z

Handshaking = Quittungsbetrieb
Ohne exakte zeitliche Abstimmung der Operationen funktioniert kein Rechner. Auch altgediente Microprozessoren wie der Motorola 6800 arbeiten schon mit einer Taktfrequenz von 1 MHz, das heißt, sie benötigen für einen Einzelschritt wie die Befehlsübernahme aus dem RAM etwa eine Mikrosekunde. Viele Peripheriegeräte sind dagegen wesentlich langsamer – auch ein schneller Matrixdrucker beispielsweise braucht mindestens 3000 Mikrosekunden pro Zeichen. Für den Datenverkehr zwischen Rechner und Peripherie sind daher strikte Vereinbarungen im sogenannten „Protokoll" erforderlich. Beim Handshake-Betrieb ist für die Verzahnung der Abläufe in der Form gesorgt, daß ein Gerät dem anderen zunächst nur seine Sende- oder Empfangsabsicht mitteilt und erst aktiv wird, nachdem der Partner seinerseits mit einem Ready-Signal geantwortet hat. Wenn die Übertragung nicht so geregelt wird, kommt es zu Fehlern, weil schnellere Empfänger Daten doppelt lesen oder zu langsame Empfänger nur die ersten Bits der eintreffenden Information mitbekommen.

Der Handshake-Betrieb läßt sich rein softwaremäßig, aber auch hardwaremäßig innerhalb einer Schnittstelle realisieren. Beim „Peripheral Interface Adaptor" (PIA) 6820 von Motorola zum Beispiel erfolgt die Kommunikation über ein Buffer-Register. Sobald dort Daten anstehen, wird im Statusregister des PIA ein Flag gesetzt, das erst nach dem Auslesen des Datenregisters wieder gelöscht wird. Damit ist ohne weiteres ein Handshake-Betrieb möglich: Wenn etwa die CPU ein Zeichen an den PIA gesendet hat, fährt sie zunächst mit der Programmbearbeitung fort, bis das Flag im PIA-Statusregister zurückgesetzt ist, das heißt, bis das angesprochene externe Gerät die Daten übernommen hat. Dann erst kann die CPU das nächste Zeichen abschicken.

Bildnachweise
1345: Tony Duncan-Smith
1347: Steve Cross
1348, 1349, 1366: Ian McKinnell
1350, 1351, 1353, 1356, 1361, 1367, 1370: Kevin Jones
1357, 1358, 1359: Chris Stevens
1360: Bernard Jennings
1364, 1365: David Higham
1369: Liz Heaney
U3: David Weeks

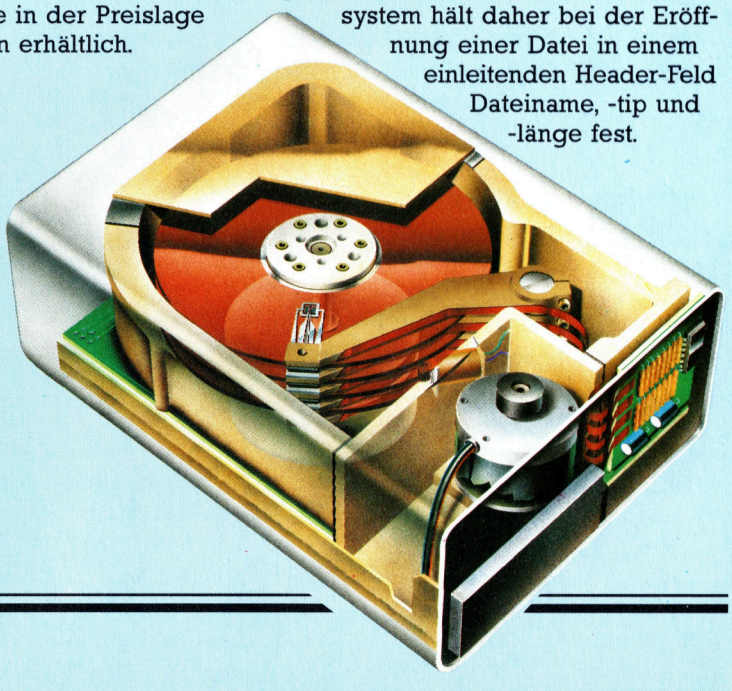
Hier werden einzelne Fachausdrücke eingehend behandelt. Da bei der Kommunikation mit dem Computer meist die englische Sprache verwendet wird, werden hier zunächst die englischen Begriffe genannt, dann die deutsche Übersetzung. In den Gesamtindex werden sowohl deutsche als auch englische Stichwörter aufgenommen,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

Hard Disk = Festplatte
Als Alternative zu den Diskettenlaufwerken, bei denen das Speichermedium zwar austauschbar ist, aber nur geringe Kapazität und Geschwindigkeit zuläßt, erfreuen sich die schnellen Festplatten mit ihrer großen Aufnahmefähigkeit wachsender Beliebtheit. Die starren Platten rotieren mit hoher Drehzahl in einem versiegelten Gehäuse. Das ermöglicht rund die zehnfache Aufzeichnungsdichte wie bei Disketten. Heute sind Festplatten mit Kapazitäten zwischen 10 und 30 Megabyte in der Preislage von Bürorechnern erhältlich.

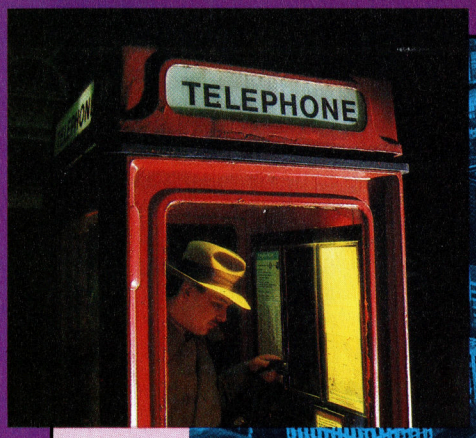
Der Umgang mit den Hard Disks erfordert einige Sorgfalt, speziell hinsichtlich der Sicherung von Betriebssoftware und Datenfiles.

Hashing = Hash-Codierung
Wenn große Datenmengen mit System in begrenztem Speicherraum untergebracht werden sollen, braucht man ein Zuordnungsverfahren, das an der Datenträgerstruktur orientiert ist. Eine Möglichkeit ist die Hash-Codierung, die eine rationelle Raumnutzung bei erträglichen Zugriffszeiten erlaubt. Angenommen, Sie möchten die Datensätze in einer Datei alphabetisch anordnen, wobei der Text im jeweils ersten Datensatzfeld wie bei einem Lexikon die Reihenfolge bestimmen soll. Vorteilhaft wäre es, unmittelbar die ASCII-Codes zur Adressierung der Datensätze heranzuziehen. Die Anzahl der möglichen Kombinationen aus nur zehn ASCII-Zeichen, von denen jedes 26 verschiedene Werte (von A–Z) annehmen kann, ist aber so groß, daß sie in keine Datei hineinpassen würde. Die Lösung besteht darin, die ASCII-Zeichenfolgen durch „Hashing" so zu verschlüsseln, daß sich die Anzahl von Kombinationsmöglichkeiten reduziert.

Header = Dateikopfsatz
Daten und Programme können als unterschiedlich formatierte Daten gespeichert werden. Das Betriebssystem hält daher bei der Eröffnung einer Datei in einem einleitenden Header-Feld Dateiname, -tip und -länge fest.



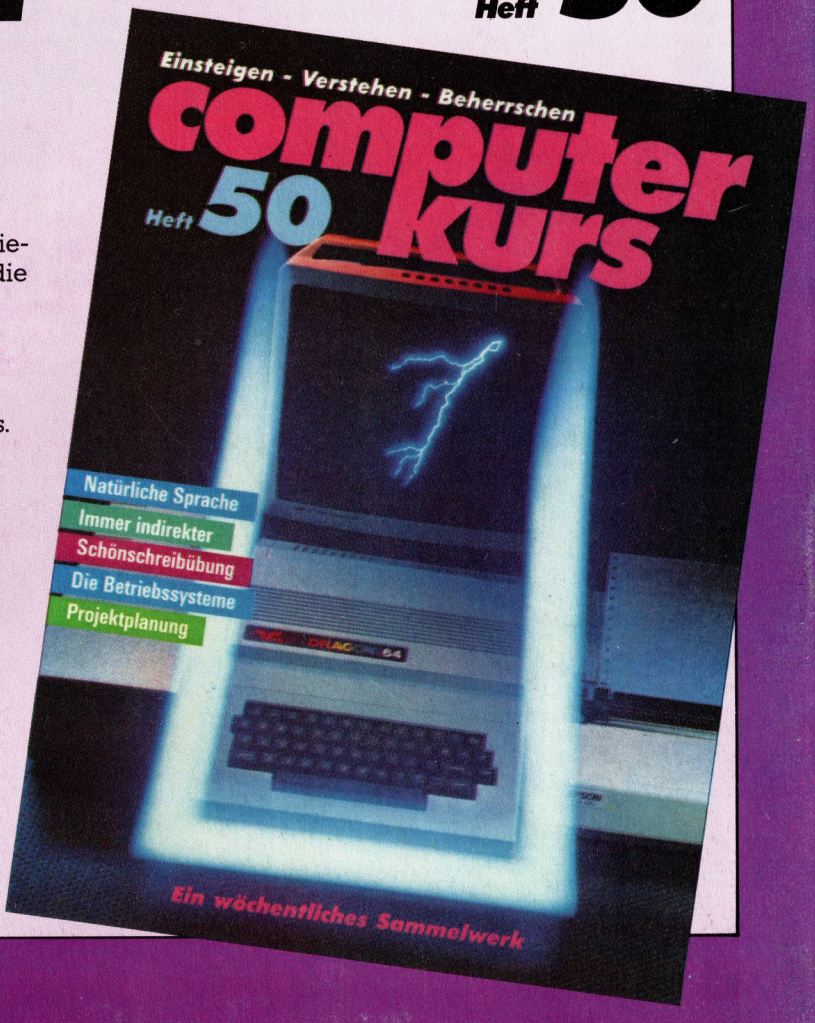
Heute ist es üblich, jede Festplatteneinheit „Winchester"-Laufwerk zu nennen – ursprünglich war das eine von IBM geprägte Bezeichnung. Erzählungen zufolge hießen dort die Festplatten in der Entwicklung wegen ihrer Speicherkapazität von 2 × 30 Megabyte zunächst 30/30er, was zufällig das Kaliber der historischen Winchesterbüchse war.



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

computer kurs

Heft 50



Große Sprünge

Große Sprünge mit der indizierten Adressierung. Aus der aktuellen Adresse werden die Daten endgültig geladen.

Simultansteuerung

Listings für die Simultansteuerung von maximal acht Motoren über die User-Ports.

Kollisionskurs

Ein Programm stellt Kollisionen zwischen Spielfiguren und vorhandenen Minen fest.

Kritischer Pfad

Damit sich niemand verrennt, ein Vorschlag für geordnete Projektpläne.

Allzeit bereit

Stets zu Diensten ist das Betriebssystem. Welche genauen Aufgaben es hat, wird hier beschrieben.