Einsteigen - Verstehen - Beherrschen

DM 3,80 öS 30 sfr 3,80

computer kurs

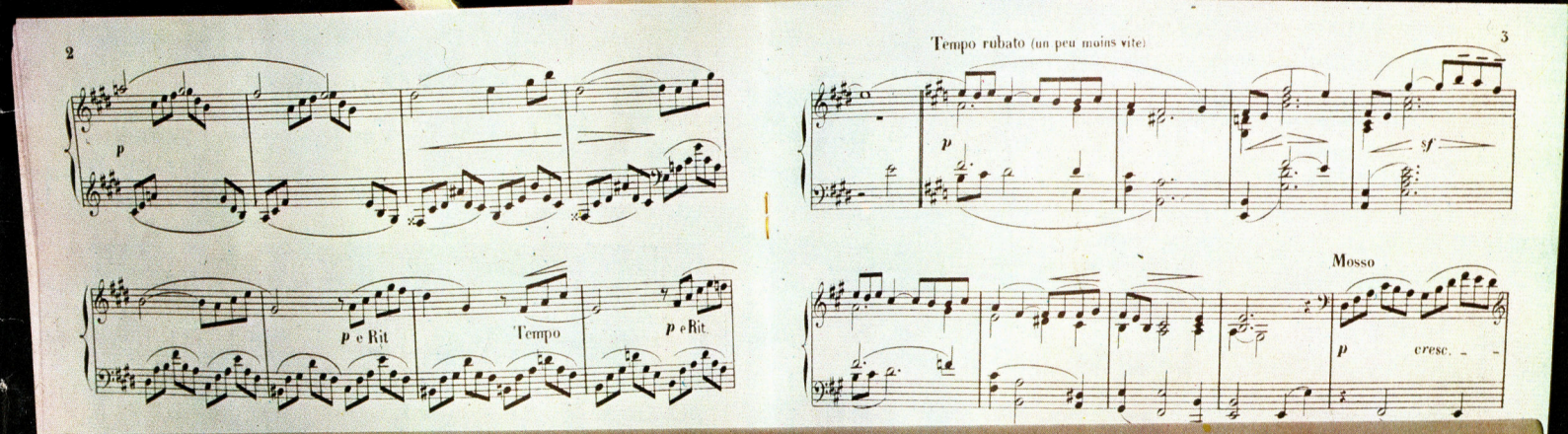
Heft 22

Texte und Zeichen

Praxis: Kontakte prüfen

Serie: Computer und Musik

Der MSX-Standard



computer kurs

Heft 22

Inhalt

WIE SIE JEDE WOCHE IHR HEFT BEKOMMEN

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

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

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

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

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

Abonnement

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

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

SAMMELORDNER

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

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

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

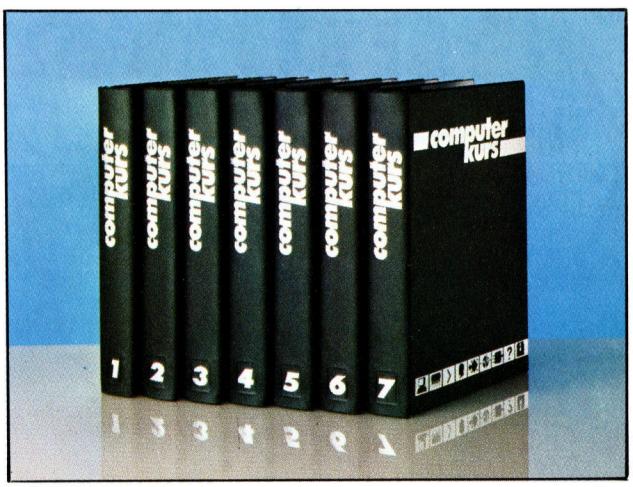
Schweiz: Der Sammelordner kostet sfr 15. Bitte wenden Sie sich an Ihren Kiosk; dort werden Sie jederzeit die gewünschten Exemplare erhalten

INHALTSVERZEICHNIS

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

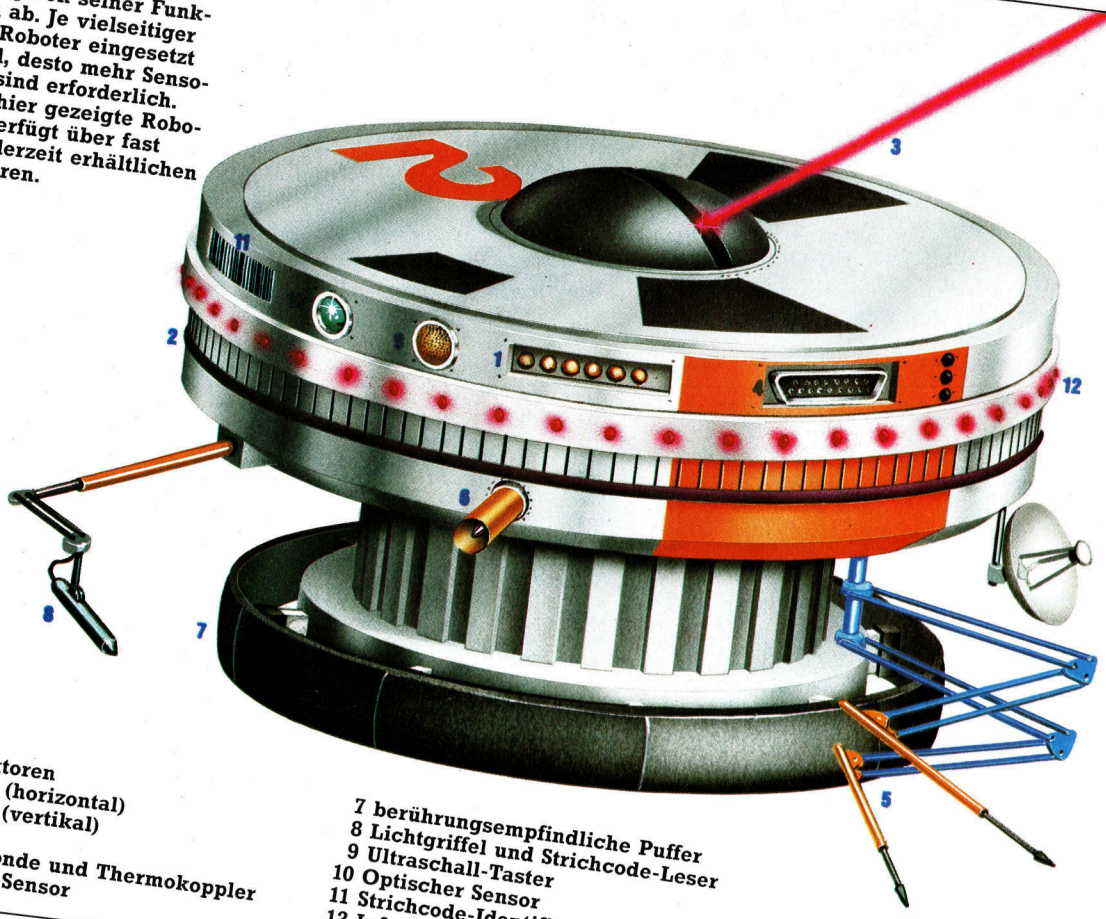
Redaktion: Winfried Schmidt (verantw. f. d. Inhalt), Joachim Seidel, Elke Leibinger, Susanne Brandt, Uta Brandl (Layout), Sammelwerk Redaktions-Service GmbH, Paulstraße 3, 2000 Hamburg 1

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



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

Die Sinnesausstattung eines Roboters hängt völlig von seiner Funktion ab. Je vielseitiger ein Roboter eingesetzt wird, desto mehr Sensoren sind erforderlich. Der hier gezeigte Roboter verfügt über fast alle derzeit erhältlichen Sensoren.



1 Gas-Detektoren
2 Tast-Laser (horizontal)
3 Tast-Laser (vertikal)
4 Data Port
5 Mehrfachsonde und Thermokoppler
6 Gas-Dichte-Sensor
7 berührungsempfindliche Puffer
8 Lichtgriffel und Strichcode-Leser
9 Ultraschall-Taster
10 Optischer Sensor
11 Strichcode-Identifikationseinheit
12 Infrarot-Detektoren

Verstand und Gefühl

In vorhergehenden Beiträgen haben wir uns mit den verschiedensten Methoden von Robotersteuerung und der Konstruktion mechanischer „Arme" und „Hände" beschäftigt. Eine andere grundlegende Frage soll in dieser Folge beantwortet werden: Wie kann ein Roboter „fühlen", was in der Umwelt geschieht?

Die Sinne des Menschen nehmen wir als gegeben hin, als selbstverständlich, und vergessen oft dabei, daß ein Mensch ohne Sinnesorgane völlig hilflos ist. Ohne das Sehen würde man gegen jedes Hindernis laufen. Und wäre nicht der Tastsinn, wüßten Sie nicht einmal, daß Sie sich gestoßen haben.

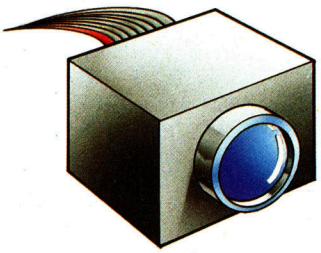
Die Art und Weise der Roboterbewegung wurde bereits dargelegt. Bevor ein Roboter sich aber unabhängig bewegen kann, ist ein Sensorsystem, eine Art „Sinnes"-System, erforderlich. Damit könnte ein Roboter so gebaut werden, daß er über alle menschlichen Sinne verfügt. Im Augenblick jedoch ist das unmöglich. Wir werden uns mit sehr einfachen Seh- und Hörmöglichkeiten beschäftigen, die jedoch mit den Fähigkeiten des Menschen nicht im entferntesten vergleichbar sind.

„Sehen" kann ein Roboter, indem er mit einem Lichtsensor, üblicherweise einer Fotozelle, ausgestattet wird. Sie erzeugt eine elektrische Spannung, die von dem auf sie wirkenden Licht abhängt. Das Prinzip ist zwar einfach, aber diese Art von Sensor ist schon recht effektiv in der Anwendung. So kann beispielsweise ein damit ausgestatteter Roboter einen bestimmten Punkt ansteuern. Dieses Prinzip ließe sich z. B. einsetzen, um ihn zu einer Stromquelle zu führen, an der er seine Batterien aufladen kann.

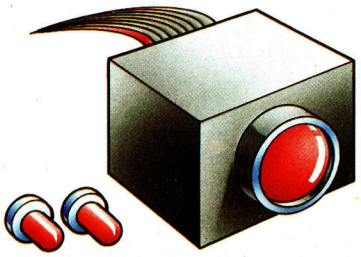
Eine einfache fotoelektronische Zelle gibt einem Roboter die Möglichkeit, sehr viele Aufgaben durchzuführen. So könnte ein am Fließband eingesetzter Roboter überprüfen, ob ein bestimmtes Bauelement vorhanden ist oder nicht, weil er einen Helligkeitsunterschied aufgrund des „Nichtvorhandenseins" des Gegenstandes registrieren würde. Die Aufgabe ließe sich noch vereinfachen, wenn durch entsprechenden Lichteinfluß diese Unterschiede intensiviert werden könnten.

Sensoren

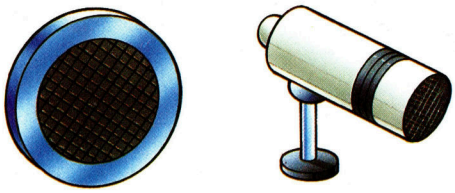
Der optische Sensor ist eine langsam tastende monochrome Fernsehkamera mit niedriger Auflösung. Sie erzeugt ein Bild in verschiedenen Grauwerten, das genug Informationen enthält, um einfache Aufgaben zu erfüllen, beispielsweise das Aufspüren von Ecken.



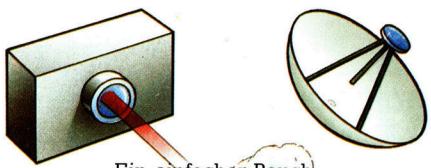
Die Infrarotkamera setzt das Bild ähnlich wie eine Fernsehkamera zusammen, registriert aber nur das Infrarotspektrum. Infrarot durchdringt Rauch und Nebel besser als Licht und kann auch die Temperatur von Gegenständen messen.



Ultraschall ist hochfrequenter Klang, der hier für Abstandsmessungen verwendet wird. Der Taster besteht aus einem Ultraschallsender und einem Richtmikrofon als Empfänger. Treffen die Ultraschallwellen auf ein Objekt, werden sie als Echowellen von seiner Oberfläche in einer ganz eindeutigen Form reflektiert und sind quasi eine Art unverwechselbares „Kennzeichen".

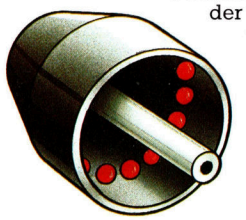


Niedrigfrequente Lasertaster werden für exakte Entfernungs- und Richtungsmessungen verwendet. Laserlicht läßt sich sehr fein bündeln. Das ermöglicht die genaue Untersuchung eines in der Nähe befindlichen Gegenstandes.

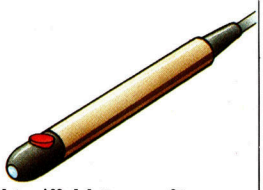


Ein einfacher Rauchdetektor besteht aus einem Lichtstrahl, der auf eine empfangende Fotodiode gerichtet ist. Sobald Rauch oder Nebel in den Strahlengang dringt, wird der ausgesandte Strahl unterbrochen.

Der Gasmesser besteht aus einem Ventil (Emitter) und einem Druck-Sensor. Aus diesem Ventil strömt ständig Gas in eine Kammer, wodurch der vorhandene Druck zunimmt. Befindet sich ein Gegenstand dicht vor dem Kammerzugang, beeinflußt dies den Druckzuwachs, der Gegenstand wird „erkennbar".



Vielfach-Meßsonden messen u. a. Widerstand, Stromstärke, Spannung und Kapazität. Als Temperaturtaster sind sie ebenfalls einsetzbar.



Lichtgriffel können für viele Arten digitaler Eingabe verwendet werden. Eines der bekanntesten Beispiele ist der Strichcode.

Die Ausstattung eines Roboters mit einem Mikrofon bietet die Möglichkeit, ihn akustische Signale „hören" zu lassen. Natürlich „versteht" der Roboter nicht, was er hört, doch das ist auch nicht erforderlich. Durch mehrfaches Wiederholen einer bestimmten Befehls- bzw. Signalreihe könnte der Roboter ein „Gerüst" von Klängen oder Geräuschen für jeden Befehl speichern, das ihm ermöglicht, neue Anordnungen mit bereits gehörten zu vergleichen.

Einfacher Tastsinn läßt sich ebenfalls in einen Roboter integrieren. Dazu baut man Mikroschalter in das Gehäuse. Diese schaffen immer dann, wenn Druck auf sie ausgeübt wird, eine elektronische Verbindung. Mit der Empfindlichkeit des menschlichen Tastsinnes ist das natürlich nicht vergleichbar. Aber Tastsensoren, die an den Kanten eines beweglichen Roboters angebracht wären, gäben ihm immerhin die Möglichkeit, intelligent zu reagieren, wenn er gegen Hindernisse stößt: Er könnte dem Hindernis ausweichen und sich einen Weg in anderer Richtung suchen.

Geruchssinn ließe sich ebenfalls einbauen, z. B., indem man Rauch- oder Gasdetektoren verwendet. Gasdetektoren basieren auf einem einfachen Prinzip: Es handelt sich dabei ebenfalls um Sensoren-Elemente (meist Platindraht), die auf bestimmte Gase reagieren. Sind diese Gase vorhanden, erfährt der Stromfluß innerhalb des Elements eine Veränderung. Rauchdetektoren bestehen aus zwei Kammern – einer geschlossenen, die als Referenz oder „Kontrolle" verwendet wird, sowie einer geöffneten. In beiden Kammern befindet sich ionisiertes Helium. Ist Rauch vorhanden, verändert sich die Menge der Partikel in der offenen Kammer. Ein Detektor, der die Partikelmengen in beiden Kammern miteinander vergleicht, kann den Unterschied sofort feststellen.

Praktische Nutzanwendung

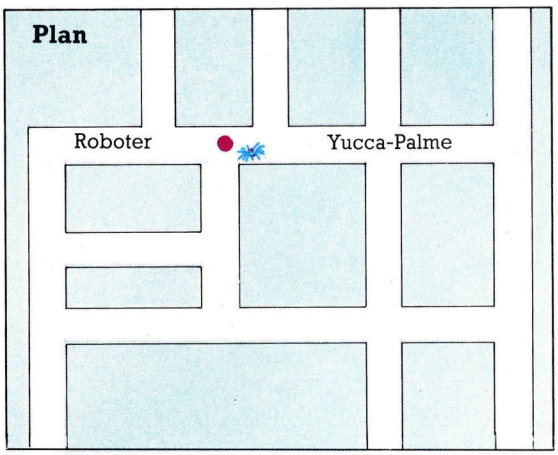
Bis heute hat man noch keinen Weg gefunden, einen Roboter mit Geschmackssinn auszustatten. Die vorgenannten Möglichkeiten versetzen ihn jedoch in die Lage, zu sehen, zu hören, zu fühlen und zu riechen. Das bedeutet: Er könnte ein Feuer in einem Haus registrieren, zur Brandstelle eilen, dabei Hindernissen ausweichen und, falls er einen Feuerlöscher in seinem „end effector" hat, den Brand mit Schaum bekämpfen.

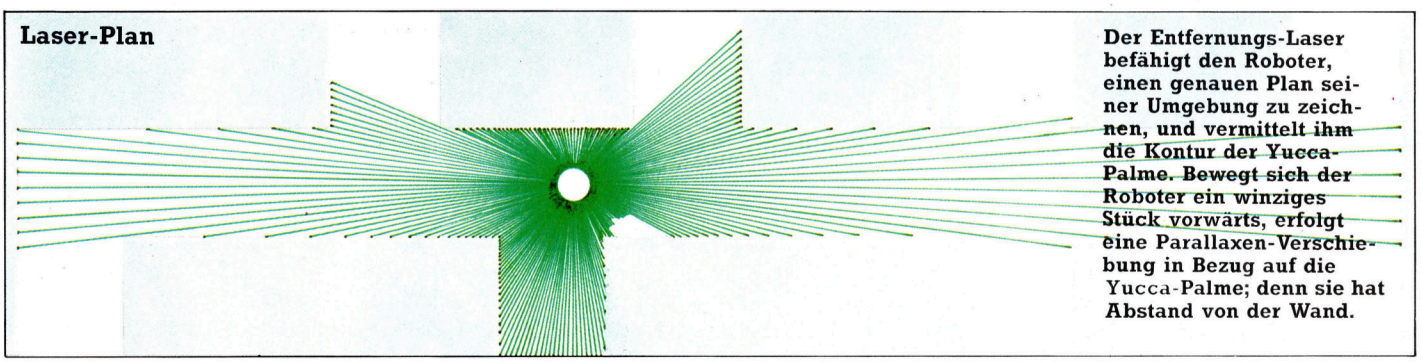
Einen Roboter lediglich mit den Sinnen des Menschen auszustatten, würde bedeuten, die in ihm steckenden Möglichkeiten ungenutzt zu lassen. Es gibt keinen Grund dafür, Roboter Dinge nur so erkennen zu lassen, wie wir es mit unseren Sinnen können. Mit welchen „Sinnen" sollte ein Roboter überhaupt ausgestattet sein, und gibt es dafür tatsächlich eine praktische Anwendung?

Nehmen wir die Roboterarme. Angenommen, wir wollen, daß ein Roboter einen Gegen-

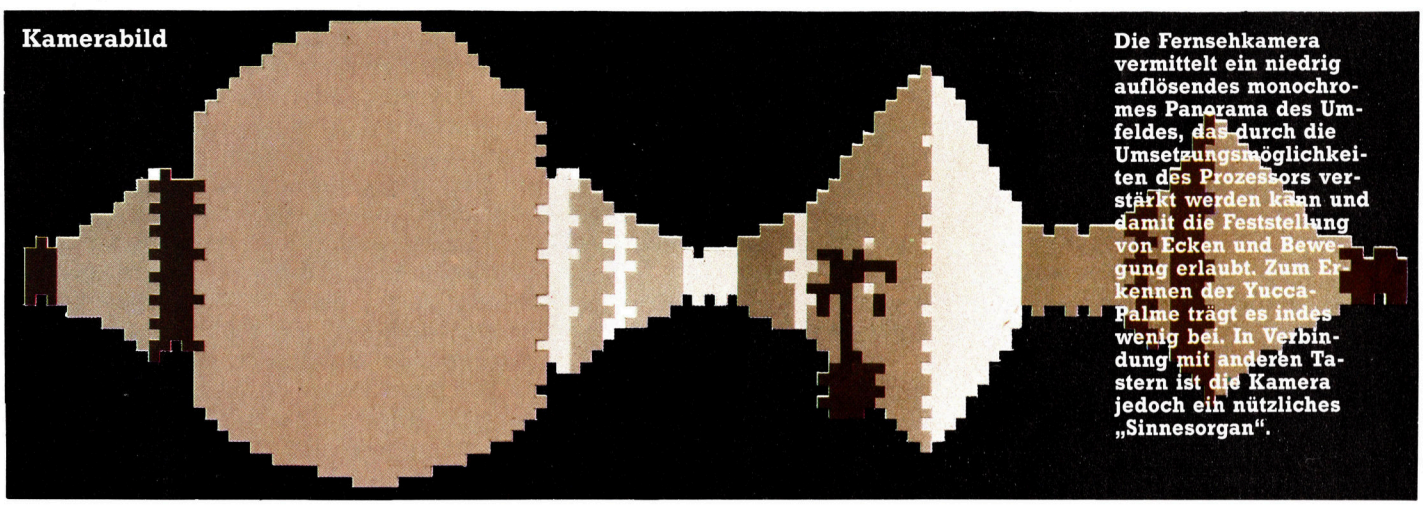
Sinne durch Sensoren

Das Erfassen der Außenwelt ist für den Roboter das größte Problem. Es nimmt mit der Reichweite und Komplexität seiner Sensoren sogar zu. Ein einzelner Sensor kann kein komplettes, informatives Bild vermitteln, und manche Sensoren scheinen anderen genau entgegenzuwirken. Der Umfang, in dem ein Roboter die durch seine verschiedenen Sensoren vermittelten Eindrücke integrieren und miteinander vergleichen kann, ist Maßeinheit für sein externes „Bewußtsein".
Nebenstehender Grundriß zeigt, daß der Roboter sich in einem Korridor befindet, dessen Wände weiß gemalt sind. Es gibt nur eine Lichtquelle, von deren Richtung die Beleuchtung der Wände abhängt. In der Nähe des Roboters steht eine Yucca-Palme, die er auf verschiedene Arten „identifizieren" kann.

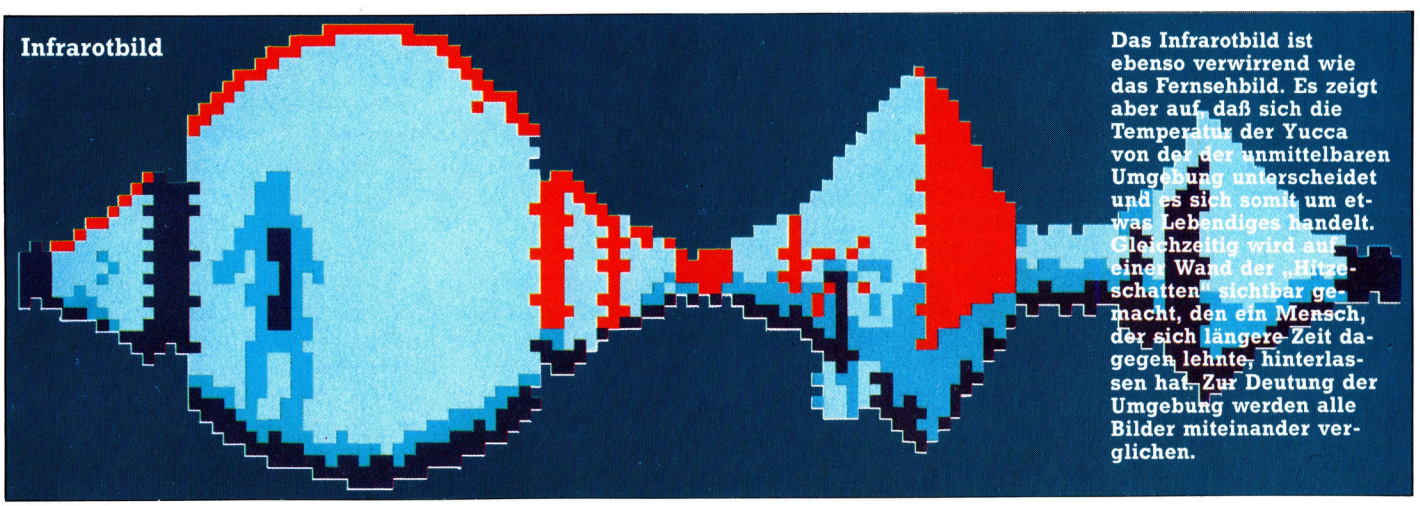




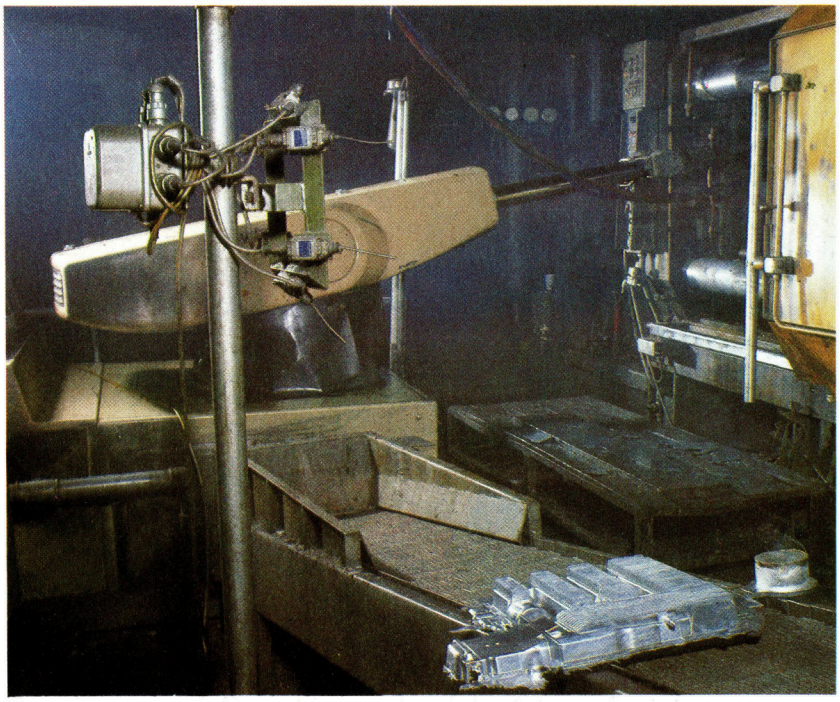
Der Entfernungs-Laser befähigt den Roboter, einen genauen Plan seiner Umgebung zu zeichnen, und vermittelt ihm die Kontur der Yucca-Palme. Bewegt sich der Roboter ein winziges Stück vorwärts, erfolgt eine Parallaxen-Verschiebung in Bezug auf die Yucca-Palme; denn sie hat Abstand von der Wand.



Die Fernsehkamera vermittelt ein niedrig auflösendes monochromes Panorama des Umfeldes, das durch die Umsetzungsmöglichkeiten des Prozessors verstärkt werden kann und damit die Feststellung von Ecken und Bewegung erlaubt. Zum Erkennen der Yucca-Palme trägt es indes wenig bei. In Verbindung mit anderen Tastern ist die Kamera jedoch ein nützliches „Sinnesorgan".



Das Infrarotbild ist ebenso verwirrend wie das Fernsehbild. Es zeigt aber auf, daß sich die Temperatur der Yucca von der der unmittelbaren Umgebung unterscheidet und es sich somit um etwas Lebendiges handelt. Gleichzeitig wird auf einer Wand der „Hitzeschatten" sichtbar gemacht, den ein Mensch, der sich längere Zeit dagegen lehnte, hinterlassen hat. Zur Deutung der Umgebung werden alle Bilder miteinander verglichen.



Dieser Industrie-Roboterarm reinigt Gußteile, die gerade aus der Gießform kommen. Für einen Menschen bzw. die menschliche Hand sind diese zu heiß. Der Roboter ist hitzeunempfindlich und kann die Arbeit deshalb ohne Wartezeit verrichten.

stand von einem beliebigen Ort aufnimmt und ihn anderswo wieder absetzt. Eine Umsetzungsmöglichkeit wäre, im Bewegungsradius des Arms fixierte Stopstationen anzubringen, die ihm erlauben, sich in beliebiger Richtung, jedoch nur in einem bestimmten Maximalumfang zu bewegen. Das bedeutet: Der Arm bewegt sich, bis er einen Haltepunkt erreicht hat, der über dem zu ergreifenden Objekt sein sollte (wenn alles andere korrekt positioniert worden ist). Nachdem er den Gegenstand aufgegriffen hat, dreht sich der Arm in entgegengesetzter Richtung, bis er an den anderen Haltepunkt gelangt. Dort wird der betreffende Gegenstand abgelegt.

Erkennen von Objekten

Ein noch besseres Beispiel ist die „Sehfähigkeit" von Robotern. Der Mensch vermag nur sichtbares Licht zu erkennen – ein großer Teil des elektromagnetischen Spektrums bleibt dem menschlichen Auge verborgen –. Es gibt jedoch keinen Grund, warum ein Roboter solchen Begrenzungen ebenfalls ausgesetzt sein sollte. Statt fotoelektrischer Zellen könnten also durchaus Infrarot-Detektoren Anwendung finden. Mit ihrer Hilfe ist das Messen der von einem Gegenstand erzeugten Hitze möglich. Industrieroboter könnten mit solchen Detektoren ausgestattet sein, um Abstand von gefährlich heißen Dingen zu halten. Ein Roboter wäre außerdem in der Lage, die Wärme des menschlichen Körpers zu registrieren. In der Praxis bedeutet das: Man könnte einen Roboter so programmieren, daß er Sie begrüßt, wenn Sie die Tür öffnen. Eine weitere Möglichkeit böte sich mit dem Erkennen von Magnetfeldern sowie der Unterscheidung zwischen magnetischen und nicht magnetischen Materialien der Umgebung an.

Für Abstandssensoren gibt es kein direktes Äquivalent beim Menschen. Dabei handelt es sich um Sensoren, die feststellen, wann sie in der Nähe eines Gegenstandes sind. Ein Mensch stellt Abstand und Nähe mittels einer Kombination von Augen- und Berührungssinn fest. Für robotische Anwendung reicht ein einfacher Abstandssensor völlig. Die Funktionsweise ist typenabhängig. So gibt es einen Sensorentyp, bei dem Luft durch eine Düse gepreßt wird. Gegenstände, die sich im Düsenstrahl befinden, leiten den Strahl in die Düse zurück. Dadurch entsteht ein Gegendruck, der durch einen entsprechenden Druckumformer ein Signal auslöst, das dem Roboter ein Hindernis meldet. Ein anders konstruierter Abstandssensor basiert auf dem Prinzip, daß sich ein elektrischer Strom innerhalb eines Stromkreises bei Annäherung an ein anderes Objekt verändert. Eine Art elektrisches „Leck" zwischen Sensor und Gegenstand (der einen eigenen Stromkreis hat) informiert den Roboter über das Vorhandensein eines Hindernisses.

Ferner gibt es Ultraschall-Sensoren. Sie strahlen ein Ultraschallsignal ab und empfangen das Echo eines vorhandenen Gegenstandes. Die Zeitverzögerung zwischen Signal und Echo erlaubt eine exakte Messung der Entfernung zwischen Roboter und Objekt.

Noch empfindlicher sind Laser-Sensoren. Hierbei wird ein Laserstrahl auf einen Gegenstand gerichtet, der dann das Laserlicht auf den Sensor reflektiert. Durch Vergleich der beiden Strahlen ist es möglich, den Abstand zum Objekt mit erstaunlicher Genauigkeit zu bestimmen. Diese Technik läßt sich auch über große Entfernungen anwenden. Bei der ersten bemannten Mondexpedition wurde auf der Mondoberfläche ein Reflektor so positioniert, daß ein Laser-Sensor die genaue Entfernung zwischen Erde und Mond messen konnte. Dieses Verfahren ist auf 15 Zentimeter genau – über eine Entfernung von 384 400 Kilometern!

Drucksensoren sind eine weitere Möglichkeit, physische Informationen zu bekommen, also zu „tasten". Und sie erweisen sich als weitaus empfindlicher als mechanische Mikroschalter. Ihr Funktionsprinzip basiert auf dem Wechsel der elektrischen Eigenschaften eines piezoelektrischen Kristalls, wenn dieser einem Druck ausgesetzt ist.

All diese Robotersensoren nennt man Wandler, da sie eine Größe in bestimmter Form messen (Licht, Geräusch oder Druck) und diese in eine andere Form umwandeln, die in etwa der originalen Meßgröße entspricht. Bei einem computergesteuerten Roboter setzen diese Wandler den Meßwert in ein elektronisches Signal um, das binär oder analog ist. Im letzteren Fall muß das elektronische Signal in eine Form umgesetzt werden, die der Computer verstehen kann. Dies geschieht mit einem Analog/Digital-Wandler (A/D-Konverter).

Östlicher Standard

Mehr als ein Dutzend japanischer Elektronik-Hersteller hat sich auf den von der Firma Microsoft geschaffenen MSX-Standard für Heimcomputer geeinigt.

Der HX-10 von Toshiba verfügt über zwei Anschlüsse für Joysticks, eine parallele Centronics-Schnittstelle, einen Schacht für ROM-Cartridges und einen Tastenblock zur Cursorsteuerung. Das MSX-BASIC verarbeitet die Impulse der Joysticks auf die gleiche Weise wie die der Cursorsteuerung. Damit lassen sich auf Joysticks ausgerichtete Spiele auch mit den Cursor-Steuertasten betreiben.

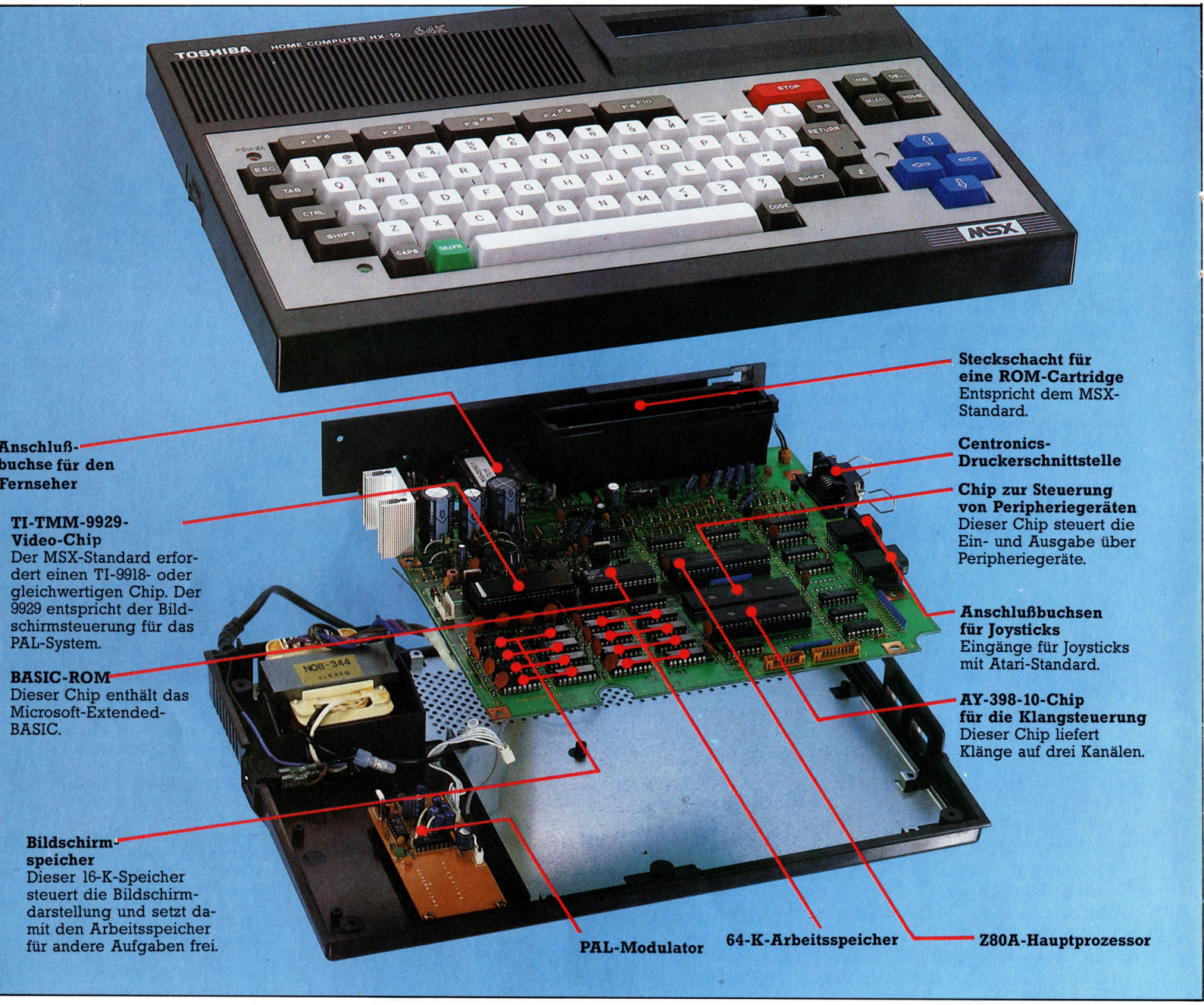
Der MSX-Standard legt fest, welche Zentraleinheit einzubauen ist (der langjährig bewährte Z80A), weiterhin das Minimum der ROM- (32 KByte) und RAM-Kapazität (8 KByte), den Standard der Ton- und Grafikchips, den Tastaturumfang (die Anordnung der Tasten kann variieren), die Mindestanzahl der Schnittstellen und ihren Aufbau, die Grafik- und Textdarstellung und natürlich die Programmiersprache BASIC, die im ROM vorhanden sein muß. Den Herstellern ist dabei freigestellt, ob sie die Speicherkapazität nach oben erweitern, eine bestimmte Tastatur auswählen oder zusätzliche Schnittstellen einbauen. Die Praxis hat gezeigt, daß die Maschinen von Sony und Toshiba – wie auch die vieler anderer Hersteller – Kapazitäten haben, die über den Minimalforderungen liegen.

Der Sony Hit-Bit wie auch der HX-10 von Toshiba sind mit qualitativ hochwertigen Tastaturen ausgerüstet. Beide Micros werden standardmäßig mit einem Arbeitsspeicher von 64 KByte und zusätzlichen 16 KByte für die Bildschirmdarstellung geliefert, – insgesamt 80 KByte RAM. Die Maschinen von Sony und Toshiba sind mit einer Standard-Centronics-Druckerschnittstelle sowie mit zwei Joystick-Ports ausgerüstet.

Ursprünglich wurde erwartet, daß die MSX-Maschinen zu extrem niedrigen Preisen auf den Markt kommen würden. Wechselkursschwankungen und gestiegene Herstellungskosten haben die Preise jedoch gehoben. Der Sony z. B. wird für ca. 1000 Mark verkauft, und auch der Toshiba HX-10, der voraussichtlich im Herbst 1985 auf den Markt kommen soll, wird auf diesem Preisniveau liegen.

Beim Anschalten einer MSX-Maschine erscheinen am unteren Bildschirmrand einige BASIC-Befehle wie RUN, LIST, CLOAD etc. Diese können über fünf Funktionstasten aufgerufen werden. Die automatische Tasten-Belegung kann mit dem Befehl KEY jedoch geändert werden. Die fünf Tasten können insgesamt zehn Funktionen aufrufen, wobei jede Taste doppelt belegt ist und die zweite Funktion durch die gleichzeitige Betätigung der Shift- und der gewünschten Funktionstaste definiert wird. Beim Drücken der Shift-Taste werden auch die auf dem Bildschirm dargestellten Schlüsselworte ausgetauscht. Jede Funktionsbezeichnung kann bis zu 15 Zeichen enthalten, von denen die ersten sieben auf dem Bildschirm erscheinen.

Das Zusammenspiel von Tastatur und Bildschirmeditor vereinfacht Eingabe und Korrektur. Mit den vier Steuerungstasten wird der Cursor problemlos an jede beliebige Stelle des Bildschirms gesetzt, die dann durch Überschreiben korrigiert werden kann. Auch das Einfügen und Löschen von Zeichen läßt sich mit jeweils einem einzigen Tastendruck ausführen. Die Cursortasten der MSX-Geräte sind leicht zugänglich und vereinfachen so die Be-



dienung erheblich.

Wie die Hardware verfügt auch die MSX-Software über viele zusätzliche Fähigkeiten. Das MSX-BASIC enthält Befehle wie AUTO und RENUM und kann über spezielle Kommandos Klänge, Grafik und „Interrupts" (Unterbrecherschaltungen) steuern. Für die Grafik gibt es drei Grundbefehle: LINE zeichnet eine Linie zwischen zwei Punkten, kann aber auch für die Darstellung eines Kastens eingesetzt werden, wenn hinter den Koordinaten ein B (für „Box") eingegeben wird. Die Buchstaben BF (für „Box Fill") füllen den gezeichneten Kasten mit Farbe aus. Auch der Befehl PAINT kann einen umrissenen Bereich einfärben, wobei auch die kompliziertesten Formen keine Probleme darstellen. Der Befehl CIRCLE zeichnet Ellipsen, Kreisbögen und einfache Kreise.

Das MSX-BASIC enthält noch eine Reihe weiterer interessanter Befehle, wobei die „Unterbrecherroutinen" am meisten beeindrukken. Unterbrecherroutinen sind besonders für Grafiken mit hohen Geschwindigkeitsanforderungen wichtig. Es gibt eine Anzahl Situationen, bei denen das Programm eine Aufgabe ausführt, während es ständig einen anderen Vorgang überprüfen muß. Ein typisches Beispiel dafür ist der Programm-Oldie „Space Invaders", bei dem sich die Angreifer auf dem Bildschirm bewegen, während gleichzeitig überprüft werden muß, ob der Joystickknopf gedrückt wurde. Das Programm bewältigt diese Aufgabe, indem es mit entsprechend hoher Geschwindigkeit zwischen beiden Funktionen umschaltet.

Das MSX-BASIC definiert zur Lösung dieser

Farbe bekennen

MSX-Standard

CPU Z80A, 3,58 MHz
RAM Minimum 8 KByte
ROM 32 KByte inklusive BASIC
Monitor 16 Farben, 256 x 192 Bildschirmpunkte, 40 x 24 Textdarstellung (oder 32 x 24), (TI 9918 oder ein entsprechender Chip als Videosteuerung)
Tonerzeugung 3 Kanäle, vom BASIC aus zugänglich, (AY38910-Chip für Tonsteuerung)
Schnittstellen Steckschacht für Cartridges, moduliertes Fernsehsignal, Centronics-Druckeranschluß, Cassettenrecorder-Anschluß
Tastatur QWERTY-Tastatur mit Funktionstasten, 4 Cursortasten und 10 programmierbaren Funktionstasten

MSX-Varianten

SONY HIT-BIT integrierte Software, RGB-Schnittstelle, 4K-RAM-Module zur Speichererweiterung
TOSHIBA HX-10 Erweiterungssteckleiste, 2 Buchsen für Joysticks
YAMAHA CX-5 Miniatur-Musiktastatur und MIDI-Software
PIONEER Schnittstelle zur Bildplattensteuerung
SANYO MPC100 Anschlußmöglichkeit für Lichtgriffel und entsprechende Software
JVC HC7GB Ausgang für einen RGB-Monitor
SPECTRAVIDEO SVI 728 Zehnerblock für Zahleneingaben

Obwohl nach dem MSX-Standard nur der Einbau eines 8-K-Arbeitsspeichers gefordert ist, verfügen alle aufgeführten Maschinen über 64 K RAM und einen zusätzlichen Bildschirmspeicher mit 16 K RAM.

Aufgabe einige Vorgänge als „Ereignisse". Der Computer erhält die Anweisung, festzustellen, ob dieses Ereignis eintritt. Ist dies der Fall, schaltet die Maschine automatisch auf ein Unterprogramm, das nun entsprechende Abläufe auslöst.

Der Grafikbildschirm der MSX-Rechner kann 16 Farben mit einer Auflösung von 256 x 192 Pixel darstellen. Bis zu 32 Sprites, die aus je acht mal acht Pixeln aufgebaut sind, lassen sich definieren (oder auch 16 Sprites zu je 16 mal 16 Punkten oder acht Sprites zu je 32 mal 32 Punkten). Zur Steuerung der Sprites enthält das MSX-BASIC eine Reihe von Spezialbefehlen wie z. B. SPRITE, mit dem ein Sprite definiert werden kann, und PUTSPRITE, der einen Sprite zudem auf eine beliebige Bildschirmposition setzt.

Inzwischen gibt es für die MSX-Computer einige Programme, wobei das Versprechen der Kompatibilität eingehalten wurde: Software, die für den HX-10 von Toshiba entwickelt wurde, läuft problemlos auf dem Sony Hit-Bit und umgekehrt. Nach jahrelanger Erfahrung mit nicht-kompatiblen Geräten ist es beeindruckend, die Cartridge eines Gerätetyps problemlos auf einem anderen einsetzen zu können. Die MSX-Hersteller rechnen damit, daß durch diese Eigenschaft schnell eine große Menge an Software für alle MSX-Maschinen zur Verfügung stehen wird.

Noch ist nicht abzusehen, ob der MSX-Standard den Marktdurchbruch auslösen wird, den sich die Japaner davon versprechen. Die starke Konkurrenz durch andere Firmen ist dabei ein wesentlicher Faktor. Abgesehen davon entsprechen die MSX-Maschinen den Ankündigungen der Hersteller: Sie verfügen über viele gut durchdachte Eigenschaften.

Standard-Interfaces

Toshiba HX-10 MSX

PREIS
voraussichtlich 1000 Mark

ABMESSUNGEN
365 x 245 x 60 mm

ZENTRALEINHEIT
Z80A, 3,58 MHz

SPEICHERKAPAZITÄT
64-K-RAM (28 K für BASIC verfügbar), 16-K-Bildschirmspeicher, 32-K-ROM inklusive BASIC

BILDSCHIRMDARSTELLUNG
24 Zeilen mit je 40 Zeichen, Grafik: 256 x 192 Bildpunkte, 16 Farben und 32 Sprites

SCHNITTSTELLEN
Centronics-Druckerschnittstelle, TV- und Monitorausgang, 2 Buchsen für Joysticks, Cassettenrecorder-Anschluß, Steckschacht für ROM-Cartridges und Erweiterungsbus

PROGRAMMIERSPRACHEN
Microsoft-Extended-BASIC

TASTATUR
68 Schreibmaschinentasten, Steuerblock für den Cursor, fünf programmierbare Funktionstasten

HANDBÜCHER
Die englischen Handbücher sind für den Erstbetrieb und das BASIC ausreichend.

STÄRKEN
Das MSX-BASIC besitzt viele interessante Fähigkeiten, darunter ausgeklügelte Befehle für die Steuerung von Ton und Grafik. Der MSX-Standard hat eindeutige Vorteile, da aufgrund der Kompatibilität in der Zukunft mit umfangreichen Softwareentwicklungen zu rechnen ist.

SCHWÄCHEN
Dem MSX-BASIC fehlt die Möglichkeit der strukturierten Programmierung. Die Menge der verfügbaren MSX-Maschinen und entsprechender Peripheriegeräte wächst leider nur langsam.

UNTERSCHIEDE
Obwohl der MSX-Standard schon einiges vorschreibt, statten die Hersteller die Geräte noch besser aus: Der Sony Hit-Bit hat z. B. drei zusätzliche Programme im ROM, einen RGB-Monitorausgang und eine andere Tastatur.

Lagerhaltung

In den ersten drei Folgen dieser Serie wurde dargestellt, wie ein Computer den Geschäftsablauf einer kleineren Handelsfirma vereinfachen kann. In diesem Artikel wird erläutert, wie eine Lagerhaltung funktioniert.

Der Besitzer oder Geschäftsführer eines gut geführten Geschäftes weiß genau, welche Waren seine Kunden verlangen und was im Augenblick gerade am Lager ist. Überfüllte Läger und fehlende Waren resultieren nur aus ungenauen Informationen, die sich durch den Einsatz eines Computers vermeiden lassen.

Bei einer effektiven Lagerhaltung muß ein Computer der Geschäftsführung eine ganze Reihe Fragen beantworten können. Dabei werden zunächst Informationen darüber benötigt, welche Waren am Lager sind, wie schnell (oder langsam) sich bestimmte Warengruppen umsetzen, wann nachgeordert werden muß und welchen Geldwert die Waren im Lager darstellen.

Mit einer computerisierten Lagerhaltung lassen sich auch die unterschiedlichen Kategorien der Warenbewegungen ausgezeichnet beobachten. Dabei gibt es: Auslieferungen aufgrund von Verkäufen, Eingänge aufgrund von Bestellungen, Ware, die für Kunden reserviert ist, und Ware, die bei Lieferanten bestellt, aber noch nicht geliefert ist.

Außer diesen vier Kategorien muß ein Lagerprogramm auch Eingabemöglichkeiten für Ware haben, die vom Käufer zurückgegeben bzw. an den Lieferanten zurückgesandt wurde. Weiterhin müssen sich Unterschiede zwischen dem theoretischen und dem tatsächlich vorhandenen Lagerbestand berichtigen lassen. Ein Lagerprogramm sollte außerdem den Lagerwert berechnen, Lagerbewegungen aufzeichnen und Preisinformationen speichern und ausgeben können.

Da die Lagerhaltung eng mit vielen Geschäftsbereichen zusammenhängt, lassen sich Lagerprogramme normalerweise mit anderen Programmen kombinieren, so daß die Daten auch für andere Verarbeitungsprozesse genutzt werden können. Ein typisches integriertes System kann mit der Buchhaltung, dem Bestellprogramm, einer Fakturierung und der Provisionsabrechnung verbunden werden.

Dieses Zusammenspiel bringt mehrere Vorteile. Nehmen wir als Beispiel ein Geschäft, dessen Bestellungen mit der Lagerverwaltung gekoppelt sind. Durch die Integration dieser beiden Systeme werden die Lagerbestände schon bei der Eingabe der Bestellung auf den neuesten Stand gebracht. Außerdem liefert dieses System neben der Warennummer die in der Lagerdatei enthaltene Warenbeschreibung und den aktuellen Verkaufspreis.

Grundlage jedes Lagerprogramms ist die Artikeldatei. Die Systeme bieten die Möglichkeit, einzelne Waren mit einer individuellen Lagernummer und einer Beschreibung zu kennzeichnen. Die Lagernummer wird dabei als Dateischlüssel verwendet.

Automatisierte Kassen entnehmen die Warenінformationen direkt aus dem Strichcode der Waren und speichern sie in der zentralen Lagerdatei des Computers. Mit dieser ständig aktuellen Information können große Warenhäuser sicherstellen, daß ihre Verkaufsregale immer mit ausreichenden Warenmengen gefüllt sind.

Gruppen-Kennzeichnung

Die meisten Programme erlauben die Eingabe von acht alphanumerischen Zeichen als Warenschlüssel und zwei zusätzlichen Zeichen als Gruppenkennzeichnung. Jede Ware kann damit einer von 50 Gruppen zugeordnet werden. Bei der Eingabe eines Warenschlüssels mit weniger als acht Stellen stellt das Programm diese Zahl automatisch nach rechts und füllt die restlichen Ziffernstellen mit Nullen. Die Eingabe 445 bedeutet daher das gleiche wie die Eingabe von 00445 oder 00000445.

Die Definierung des Eingabeformates ist sehr wichtig. Das Programm „ACT Pulsar's Stock Control System", das auf größeren Micros wie dem IBM PC und dem Sirius läuft, bietet die Möglichkeit, mit einem nach rechts oder nach links ausgerichteten Schlüsselsystem zu arbeiten. Die daraus entstehenden Lagerdateien sind völlig verschieden und können untereinander keine Daten austauschen.

Der Warengruppenschlüssel dieses Paketes kann bis zu 16 alphanumerische Zeichen enthalten. Das rechtsseitig ausgerichtete System benutzt eine nach Nummern aufgebaute Codierung, während das linksseitig ausgerich-

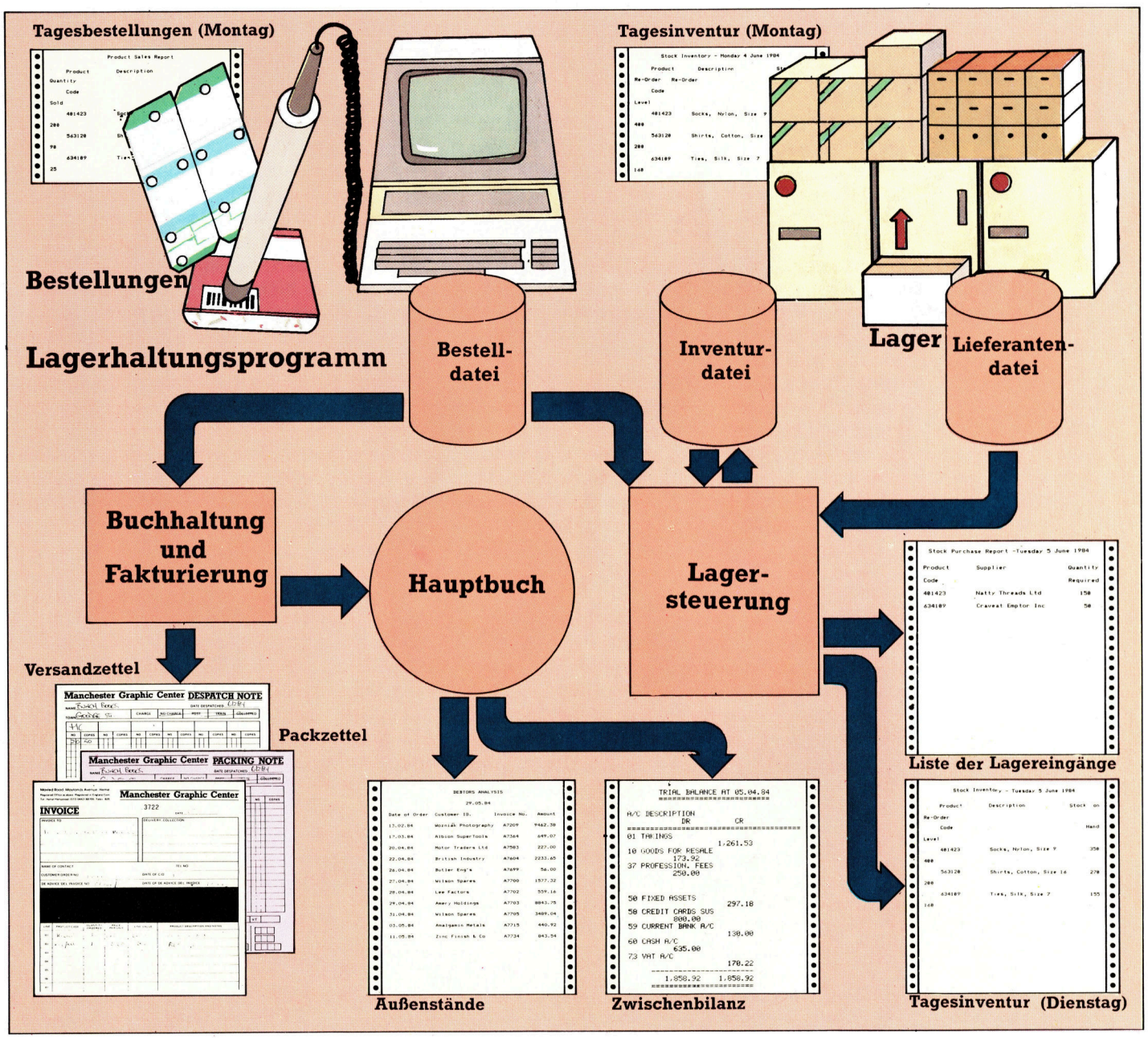
tete System für Anwender gedacht ist, die komplizierte Schlüsselsysteme mit alphanumerischen Zeichen z. B. PX445/44 verwenden. Damit lassen sich für unterschiedliche Warengruppen verschieden lange Warenschlüssel einsetzen, und der Schlüssel kann Angaben wie z. B. Farbe, Größe und Muster enthalten.

Das Programmpaket „Omicron Powerstock" läuft auf dem Sirius und ist eines der teureren Systeme, das für komplexe Anforderungen ausgelegt ist. Sein Schlüsselsystem ist komplex und baut auf den Kennzeichnungen einzelner Warengruppen auf. Eine Warengruppe kann dabei eine Anzahl Artikelsätze sein, die gemeinsame Merkmale wie Verarbeitung oder Haltbarkeitsdaten haben. Der Unterschied zu dem Pulsarsystem liegt in der Behandlung der Warengruppen, die unterschiedlichen Bearbeitungsvorgängen unterworfen werden können. Bei Pulsar lassen sich zwar jeder Warengruppe unterschiedliche Schlüsselnummern zuordnen, die Verarbeitung aber ist für alle Warennummern gleich.

Leichte Identifizierung

Die Vergabe der Warenschlüssel eines Lagerhaltungssystems muß jedoch flexibel sein, um dem Anwender eine Unterteilung seines Warenbestandes, aber auch eine leichte Identifizierung möglich zu machen. Lagerhaltungen, die auf kleineren Computern laufen, sind jedoch nur wenig flexibel und werden zudem durch den Umfang des Arbeitsspeichers und das Speichermedium stark eingeschränkt. Die Anzahl der Artikel wird von der Konfiguration des jeweiligen Computers bestimmt.

Für Geschäfte mit hohem Umsatz gibt es integrierte Verkaufs- und Lagersysteme, die mit computerisierten Kassen arbeiten, um ständig einen aktuellen Überblick über Lagerbestände und Buchhaltung zu haben. Die Wareninformation kann in Form von Strichcodes am Produkt angebracht werden. Ein optisches Lesegerät, das mit der Kasse verbunden ist, erfaßt die jeweiligen Markierungen.



Der letzte Schliff

Nachdem die Unregelmäßigkeiten, die beim Zusammenfügen der Programm-Module aufgetreten sind, beseitigt wurden, ist das elektronische Adreßbuch fertig.

Im letzten Teil dieses Kurses blieb eine Frage ungelöst: Warum wird beim Aufnehmen eines neuen Verzeichnisses (mit Routine *ADDVER*), anschließendem Suchen eines Verzeichnisses (mit Routine *FNDVER*) und abschließendem Beenden des Programms (mit Routine *ENPROG*) das neu hinzugefügte Verzeichnis nicht mit abgespeichert? Das Problem entstand durch die Verwendung der Variablen VMOD als Flag, um anzuzeigen, ob ein Verzeichnis modifiziert wurde oder nicht. Die *SRTVER*-Unterroutine würde die Datei in alphabetischer Reihenfolge sortieren und dann VMOD auf 0 setzen, in der Annahme, daß sich die Datei in der richtigen Reihenfolge befindet. Bei Ausführung von *ENPROG* wird überprüft, ob die Datei sortiert ist (VMOD=0). Ist dies der Fall, wird die Datei nicht neu abgespeichert.

Flags als Statushinweise

Das Hinzufügen eines Verzeichnisses (mit der Routine *ADDVER*) bewirkt, daß VMOD auf 1 gesetzt wird (da ein Verzeichnis modifiziert wurde bzw. ein neues Verzeichnis hinzugefügt wurde). Die Routine *SRTVER* dagegen setzt VMOD auf 0, da die Datei ja sortiert wurde. Dafür sind zwei Flags notwendig. Eines zur Indikation, daß ein Verzeichnis modifiziert wurde, sowie eines, das anzeigt, ob die Datei sortiert wurde oder nicht. Bei Bedarf können nun die entsprechenden Unterroutinen die Informationen aus dem „Sortier"- sowie aus dem „Modifiziert"-Flag abrufen.

Entsprechende Namen für die beiden Flags wären VMOD, um anzuzeigen, daß ein Verzeichnis modifiziert wurde, sowie SRTD, um anzuzeigen, daß die Datei sortiert wurde.

In einer bereits gezeigten Vorversion des Programms befand sich in Zeile 1230 die Anweisung LET SVED=0. Diese Variable wurde bisher nicht verwendet, doch hatten wir sie bereits unter der Vermutung integriert, daß VMOD als einzige Variable wahrscheinlich nicht ausreichen dürfte. Der Name wurde gewählt, da erst bestimmte Bedingungen zutreffen müssen, bevor ein neues Speichern notwendig ist (engl. SaVeEnD = Speichern Ende). Ein sinnvollerer Name für dieses Flag wäre jedoch SRTD (um anzuzeigen, daß die Datei in sortierter Form vorliegt). Die Originalzeile 1230 sieht nun wie folgt aus:

```
1230 LET SRTD = 1
```

Jetzt gibt es vier verschiedene Zustände, in denen sich die Datei befinden kann. Dies sind:

VMOD	SRTD	
0	0	Nicht modifiziert, nicht sortiert (illegal)
1	0	Modifiziert, nicht sortiert
0	1	Nicht modifiziert, sortiert
1	1	Modifiziert, sortiert

VMOD=0 und SRTD=0 ist illegal, da das Programm annimmt, daß die Datei grundsätzlich sortiert ist, bevor sie gespeichert wird. Wenn das Programm läuft, wird VMOD auf den Wert 0 gesetzt (Zeile 1220), um anzuzeigen, daß keine Modifikationen vorgenommen wurden, und SRTD wird auf 1 gesetzt (Zeile 1230), um anzuzeigen, daß die Datei sortiert ist.

Eine weitere bisher nicht verwendete Variable trägt den Namen CURR. Sie dient dazu, die Position eines durch die Such-Routine lokalisierten Verzeichnisses innerhalb der Datei zu speichern. CURR wird nicht gelöscht, wenn ihr ein Wert zugewiesen wurde. Sie wird verwendet, um Informationen über das gesuchte Verzeichnis anderen Unterroutinen zu übermitteln. Das Ende der *FNDVER*-(Such-)Routine wurde in den Zeilen 3320 und 3330 modifiziert, um den Wert von CURR zu setzen. Er wird auf 0 gesetzt, wenn das gesuchte Verzeichnis nicht gefunden werden konnte, oder aber auf den Wert von MID, wenn die Suche erfolgreich war.

Zeile 13340 verzweigt zur *NOTVER*-Unterroutine, wenn CURR den Wert 0 hat. Hier wird eine Meldung dargestellt, die besagt, daß das Verzeichnis nicht gefunden wurde. Außerdem wird der Suchschlüssel dargestellt (NAMFLD$(GROSS)). *NOTVER* verzweigt zurück zum Hauptmenü, sobald die Leertaste gedrückt wird. *NOTVER* könnte sehr leicht geändert werden, so daß der Anwender folgende weitere Möglichkeiten hat:

```
DRUECKE RETURN FUER NEUEN VERSUCH
   ODER
DRUECKE LEERTASTE, UM FORTZUFAHREN
```

Der einfachste Weg hierfür scheint das erneute Aufrufen der *FNDVER*-Unterroutine zu sein, wenn RETURN gedrückt wird. Obwohl

das Aufrufen einer Unterroutine aus der Unterroutine selbst in BASIC nicht verboten ist, wird die Rücksprungadresse (RETURN) „verwirrt". Dadurch kann es passieren, daß die Unterroutine auch dann wiederholt wird, wenn Sie es vielleicht nicht wünschen. Es gibt jedoch Möglichkeiten, um dieses Problem zu umgehen. Die Programmierung wird allerdings etwas kompliziert!

Ein einfacherer Weg wäre die Verwendung eines Flags (beispielsweise KVER für „kein Verzeichnis"). Dieses Flag könnte dann in *NOTVER* zurückgesetzt werden und so der Unterroutine ermöglichen, das RETURN in normaler Form auszuführen und einen Rücksprung zu *AUSFUH* im Hauptprogramm zu „erzwingen". Dies kann beispielsweise so aussehen: 95 IF KVER=0 THEN 80.

Eine kleine Ergänzung in Zeile 10490 in *MODNAM* sollte beachtet werden. Die numerische Variable S sollte ebenfalls zurückgesetzt werden (LET S=0). Wird dies nicht gemacht, so kann es unter gewissen Umständen vorkommen, daß *MODNAM* nicht wunschgemäß funktioniert.

Eine andere in diese endgültige Fassung des Programms integrierte Routine ist *MODVER*. Diese Routine lokalisiert zuerst das zu modifizierende Verzeichnis, indem sie *FNDVER* aufruft (Zeile 14120). Diese Zeile ruft statt 13000 Zeile 13030 auf, um das Löschen des Bildschirms durch *FNDVER* zu unterdrücken. Wenn das Verzeichnis nicht lokalisiert werden kann, kehrt das Programm in gewohnter Weise zum Hauptmenü zurück. Wenn das Verzeichnis lokalisiert werden kann, wird dieses auf dem Bildschirm dargestellt, und der Anwender erhält folgende Meldung:

```
MODIFIZIERE NAME?
DRUECKE RETURN FUER NEUEN NAMEN
ODER LEERTASTE FUER NAECHSTES FELD
```

Die Routine, die herausfindet, welche der beiden Möglichkeiten gewählt wurde, befindet sich in den Zeilen 14190 bis 14280.

Löschen eines Verzeichnisses

Die Zeilen 14190 bis 14220 beinhalten eine einfache Schleife, die dann abgebrochen wird, wenn entweder RETURN oder die Leertaste gedrückt wird. Enthält A$ weder ein RETURN (CHR$ 13) noch eine Leerstelle (CHR$ 32), wird I zurückgesetzt, und die Schleife wird erneut durchlaufen. Wenn die RETURN-Taste gedrückt wurde, dienen die nächsten Zeilen zum Einfügen des neuen Namens in NAMFLD$(CURR). Außerdem werden sie zum Setzen von VMOD, zum Zurücksetzen von SRTD, zum Aufrufen von *MODNAM* sowie zum Einfügen des durch *MODNAM* standardisierten Namens in MODFLD$(CURR) (lokalisiert in MODFLD$ (GROSS)) verwendet.

Die andere Routine, die noch eingebaut wurde, ist *LOEVER* zum Löschen eines Verzeichnisses. Sie ist ausgesprochen einfach. Als erstes wird der Bildschirm gelöscht (Zeile 15020) und eine Meldung dargestellt, die den Vorgang erklärt. Danach wird *FNDVER* aufgerufen, um das zu löschende Verzeichnis zu lokalisieren. Dann wird ein Menü angeboten, entweder RETURN zu drücken, um das Verzeichnis zu löschen, oder die Leertaste, um zum Hauptmenü zurückzukehren. Außerdem erscheint eine warnende Meldung (Zeile 15160). Eine bessere Lösung ist sicherlich, mit einer Meldung wie „SIND SIE SICHER?" zu antworten, wenn RETURN gedrückt wurde. Das Verzeichnis sollte dann nur in dem Fall gelöscht werden, wenn die Taste J (für Ja) gedrückt wird (z. B. IF INKEY$="J" THEN . . .).

Datensätze werden verschoben

LOEVER setzt das SRTD-Flag nicht zurück. Wenn die Datei bereits in alphabetischer Reihenfolge nach Namen sortiert ist, wird diese durch das Löschen eines kompletten Verzeichnisses nicht durcheinandergebracht. Trotzdem bedeutet dieser Vorgang, daß die Datei modifiziert wurde und somit in Zeile 15340 VMOD gesetzt sowie der Wert von GROSS in Zeile 13550 um eins reduziert wird, da die Datei nun ein gültiges Verzeichnis weniger beinhaltet. Alle nach dem gelöschten Verzeichnis liegenden Datensätze werden in den Zeilen 15260 bis 15320 um einen Platz nach „unten" verschoben.

Sie haben sicherlich bemerkt, daß *FNDVER* einen bedingten Aufruf zu einer Unterroutine mit dem Namen *LSTCUR* beinhaltet, um das durch *FNDVER* gefundene Verzeichnis auszudrucken. Wenn Sie keinen Drucker besitzen, ersetzen Sie Zeile 13540 durch eine REM-Anweisung für zukünftige Ergänzungen und übergehen Sie die Zeilen 13600 bis 13690.

Hiermit ist das Adreßbuch-Programm fertig. Alle im Hauptmenü aufgeführten Optionen sind bereits fertiggestellt: Finden, Hinzufügen, Ändern und Löschen eines Verzeichnisses sowie das Beenden des Programmlaufs. Das Entwickeln eines computergestützten Adreßbuches diente zur Demonstration, wie ein Programmierer ein derartiges Problem angehen sollte. Diejenigen, die das Programm als ernsthafte Anwendung einsetzen möchten, müßten noch folgendes Problem lösen: Was passiert, wenn GROSS einmal den Wert 51 erreicht? Dies ist der Fall, wenn 50 Verzeichnisse in der Datei gespeichert sind.

Im nächsten Teil des BASIC-Programmier-Kurses werden wir uns mit Themen wie beispielsweise dem Programmierstil beschäftigen. Außerdem werden wir weiterführende Möglichkeiten der BASIC-Programmiersprache, Grundlagen der Cursorsteuerung und die Abfrage der Speicheradressen behandeln.

BASIC-Dialekte

LPRINT

Dieser Befehl ist auf einigen Computern wie Commodore 64, VC 20, Acorn B sowie Dragon 32 nicht verfügbar.
Bei einem Acorn B mit einem Parallel-Drucker fügen Sie die folgenden Zeilen ein:

```
13605 VDU 2
13680 VDU 3
```

Dies aktiviert und deaktiviert den Drucker. Ersetzen Sie außerdem in den Zeilen 13610 bis 13670 PRINT gegen LPRINT. Für weitere Informationen lesen Sie bitte in Ihrem Bedienungshandbuch nach.
Bei den Commodore-Computern fügen Sie folgende Zeilen ein:

```
13605 OPEN 4,4:CMD 4
13680 PRINT
      4:CLOSE 4
```

Dies aktiviert und deaktiviert den Drucker. Ersetzen Sie außerdem in den Zeilen 13610 bis 13670 PRINT gegen LPRINT.
Beim Dragon 32 fügen Sie folgende Zeilen ein:

```
13605 OPEN"0",-2
13680 CLOSE -2
```

Dies aktiviert und deaktiviert den Drucker. Ersetzen Sie außerdem in den Zeilen 13610 bis 13670 PRINT -2, (das Komma hier ist Bestandteil des Befehles) gegen LPRINT.

SPECTRUM

Die komplette Spectrum-Fassung des Adreßbuch-Programms finden Sie in der nächsten Folge des BASIC-Programmier-Kurses.

Adreßbuch-Programm

```
  10 REM "HAUPTPROGRAMM"
  20 REM * INITIL *
  30 GOSUB 1000
  40 REM *BGRUES *
  50 GOSUB 3000
  60 REM * AUWAHL *
  70 GOSUB 3500
  80 REM * AUSFUH *
  90 GOSUB 4000
 100 IF WAHL <> 9 THEN 60
 110 END
1000 REM * INITIL * UNTERROUTINE
1010 GOSUB 1100: REM *CREARR* (DEFINIERE BEREICHE) UNTERROUTINE
1020 GOSUB 1400: REM *LSINDT* (LESE DATEI EIN) UNTERROUTINE
1030 GOSUB 1600: REM *SETFLG* (SETZE FLAGS) UNTERROUTINE
1040 REM
1050 REM
1060 REM
1070 REM
1080 REM
1090 RETURN
1100 REM *CREARR* (DEFINIERE BEREICHE) UNTERROUTINE
1110 DIM NAMFLD$(50)
1120 DIM MODFLD$(50)
1130 DIM STRFLD$(50)
1140 DIM STDFLD$(50)
1150 DIM STAFLD$(50)
1160 DIM TELFLD$(50)
1170 DIM INDFLD$(50)
1180 REM
1190 REM
1200 REM
1210 LET GROSS=0
1220 LET VMOD=0
1230 LET SRTD=1
1240 LET CURR=0
1250 REM
1260 REM
1270 REM
1280 REM
1290 REM
1300 RETURN
1400 REM *LSINDT* UNTERROUTINE
1410 OPEN "I", #1, "ADBK.DAT"
1420 INPUT #1, TEST$
1430 IF TEST$ = "@ERST" THEN GOTO 1540: REM SCHLIESSEN UND RETURN
1440 LET NAMFLD$(1) = TEST$
1450 INPUT #1, MODFLD$(1), STRFLD$(1), STDFLD$(1), STAFLD$(1), TELFLD$(1)
1460 INPUT #1, INDFLD$(1)
1470 LET GROSS=2
1480 FOR L=2 TO 50
1490 INPUT #1, NAMFLD$(L), MODFLD$(L), STRFLD$(L), STDFLD$(L), STAFLD$(L)
1500 INPUT #1, TELFLD$(L), INDFLD$(L)
1510 LET GROSS=GROSS+1
1520 IF EOF(1)=-1 THEN LET L=50
1530 NEXT L
1540 CLOSE #1
1550 RETURN
1600 REM *SETFLG* UNTERROUTINE
1610 REM SETZT FLAGS NACH *LSINDT*
1620 REM
1630 REM
1640 IF TEST$ = "@ERST" THEN LET GROSS=1
1650 REM
1660 REM
1670 REM
1680 REM
1690 RETURN
3000 REM *BGRUES* UNTERROUTINE
3010 PRINT CHR$(12): REM LOESCHT BILDSCHIRM
3020 PRINT
3030 PRINT
3040 PRINT
3050 PRINT
3060 PRINT TAB(13); "*WILLKOMMEN ZUM*"
3070 PRINT TAB(11);"*ADRESSBUCHPROGRAMM*"
3080 PRINT TAB (8);"*DES COMPUTERKURSES*"
3090 PRINT
3100 PRINT TAB (1);"(DRUECKE DIE LEERTASTE, UM FORTZUFAHREN)"
3110 FOR L=1 TO 1
3120 IF INKEY$ <> " " THEN L=0
3130 NEXT L
3140 PRINT CHR$(12)
3150 RETURN
3500 REM * AUWAHL * UNTERROUTINE
3510 REM
3520 IF TEST$="@ERST" THEN GOSUB 3860:REM *ERSTLA* UNTERROUTINE
3530 IF TEST$="@ ERST" THEN RETURN
3540 REM 'CHMENU'
3550 PRINT CHR$(12)
3560 PRINT "WOLLEN SIE"
3570 PRINT
3580 PRINT
3590 PRINT
3600 PRINT "1. VERZEICHNIS DURCH NAMEN SUCHEN"
3610 PRINT "2. NAMEN DURCH TEIL EINES NAMENS SUCHEN"
3620 PRINT "3. VERZEICHNISSE NACH STADTANGABEN SUCHEN"
3630 PRINT "4. VERZEICHNISLISTE DURCH INITIALEN"
3640 PRINT "5. LISTE ALLER VERZEICHNISSE"
3650 PRINT "6. HINZUFUEGEN EINER ADRESSE"
3660 PRINT "7. AENDERN EINER ADRESSE"
3670 PRINT "8. LOESCHEN EINER ADRESSE"
3680 PRINT "9. PROGRAMM BEENDEN UND DATEN SPEICHERN"
3690 PRINT
3700 PRINT
3710 REM 'INWAHL'
3720 REM
3730 LET L=0
3740 LET I=0
3750 FOR L=1 TO 1
3760 PRINT "WAEHLEN SIE (1-9)"
3770 FOR I=1 TO 1
3780 LET A$=INKEY$
3790 IF A$= "" THEN I=0
3800 NEXT I
3810 LET WAHL=VAL(A$)
3820 IF WAHL< 1 THEN L=0
3830 IF WAHL > 9 THEN L=0
3840 NEXT L
3850 RETURN
3860 REM *ERSTLA* UNTERROUTINE (STELLT MELDUNG DAR)
3870 LET WAHL=6
3880 PRINT CHR$(12): REM LOESCHT BILDSCHIRM
3890 PRINT
3900 PRINT TAB(8);"ES SIND KEINE VERZEICHNISSE"
3910 PRINT TAB(8);"IN DER DATEI. SIE MUESSEN"
3920 PRINT TAB(8);"MIT DER EINGABE EINES VERZEICHNISSES BEGINNEN"
3930 PRINT
3940 PRINT TAB(5);"(LEERTASTE UM FORTZUFAHREN)"
3950 FOR B=1 TO 1
3960 IF INKEY$ <> " "THEN B=0
3970 NEXT B
3980 PRINT CHR$(12): REM LOESCHT BILDSCHIRM
3990 RETURN
4000 REM *AUSFUH* UNTERROUTINE
4010 REM
4020 REM
4030 REM
4040 IF WAHL = 1 THEN GOSUB 13000: REM *FNDVER*
4050 REM 2 IST *FNDNMS
4060 REM 3 IST *FNDSTD*
4070 REM 4 IST *FNDINT*
4080 REM 5 IST *LSTVER*
4090 IF WAHL = 6 THEN GOSUB 10000: REM *ADDVER*
4100 IF WAHL = 7 THEN GOSUB 14000: REM *MODVER*
4110 IF WAHL = 8 THEN GOSUB 15000: REM *LOEVER*
4120 IF WAHL = 9 THEN GOSUB 11000: REM *ENPROG*
4130 REM
4140 RETURN
10000 REM *ADDVER* UNTERROUTINE
10010 PRINT CHR$(12): REM LOESCHE BILDSCHIRM
10020 INPUT "GIB NAMEN EIN";NAMFLD$ (GROSS)
10030 INPUT "GIB STRASSE EIN";STRFLD$(GROSS)
10040 INPUT "GIB. STADT EIN";STDFLD$ (GROSS)
10050 INPUT "GIB STAAT EIN";STAFLD$ (GROSS)
10060 INPUT "GIB TELEFONNUMMER EIN"; TELFLD$ (GROSS)
10070 LET VMOD=1:LET SRTD=0: REM MODIFIZIERT & NICHT SORTIERT
10080 LET INDFLD$ (GROSS) = STR$ (GROSS)
10090 LET TEST$=""
10100 GOSUB 10200: REM *MODNAM*
10110 LET WAHL=0
10120 LET GROSS=GROSS+1
10130 REM
10140 REM
10150 RETURN
10200 REM *MODNAME* ROUTINE
10210 REM WANDELT INHALT VON NAMFLD$ IN GROSSBUCHSTABEN UM,
10220 REM ENTFERNT UNNOETIGE ZEICHEN UND SPEICHERT IN DER REIHENFOLGE:
10230 REM FAMILIENNAME + LEERSTELLE + VORNAMEN IN MODFLD$
10240 REM
10250 LET N$=NAMFLD$(GROSS)
10260 FOR L=1 TO LEN(N$)
10270 LET TEMP$=MID$(N$, L, 1)
10280 LET T=ASC (TEMP$)
10290 IF T >=97 THEN T=T-32
10300 LET TEMP$=CHR$(T)
10310 LET P$=P$+TEMP$
10320 NEXT L
10330 LET N$=P$
10340 REM LOKALISIERT LETZTE LEERSTELLE
10350 FOR L=1 TO LEN(N$)
10360 IF MID$(N$,L,1)=" " THEN S=L
10370 NEXT L
10380 REM ENTFERNT UNNOETIGE ZEICHEN UND
10390 REM SPEICHERT VORNAMEN IN VNAM$
10400 FOR L=1 TO S-1
10410 IF ASC(MID$(N$,L,1))> 64 THEN VNAM$=VNAM$+MID$(N$,L,1)
10420 NEXT L
10430 REM ENTFERNT UNNOETIGE ZEICHEN UND
10440 REM SPEICHERT FAMILIENNAMEN IN FAMNAM$
10450 FOR L=S+1 TO LEN(N$)
10460 IF ASC(MID$(N$,L,1))> 64 THEN FAMNAM$=FAMNAM$+MID$(N$,L,1)
10470 NEXT L
10480 LET MODFLD$ (GROSS)=FAMNAM$+" "+VNAM$
10490 LET P$="": LET N$="": LET FAMNAM$="": LET VNAM$="": LET S=0
10500 RETURN
11000 REM *ENPROG* UNTERROUTINE
11010 REM SORTIERT UND SPEICHERT (SICHERT)
11020 REM DIE DATEI WENN IRGENDEIN VERZEICHNIS
11030 REM MODIFIZIERT WURDE (VMOD=1)
11040 REM ODER NICHT SORTIERT WURDE (SRTD=0)
11050 REM VMOD=0 UND SRTD=0 IST ILLEGAL
11060 REM
11070 IF VMOD=0 AND SRTD=1 THEN RETURN
11080 IF VMOD=1 AND SRTD=0 THEN GOSUB 11200: REM *SRTVER*
11090 GOSUB 12000: REM *SPEVER*
11100 RETURN
11200 REM *SRTVER* UNTERROUTINE
11210 REM SORTIERT ALLE VERZEICHNISSE NACH MODFLD$ IN
11220 REM ALPHABETISCHER REIHENFOLGE UND AKTUALISIERT INDFLD$
11230 REM
11240 REM
11250 LET S=0
11260 FOR L=1 TO GROSS-2
11270 IF MODFLD$(L) > MODFLD$(L+1) THEN GOSUB 11350
11280 NEXT L
11290 IF S=1 THEN 11250
11300 REM
11310 REM
11320 LET SRTD=1: REM SETZT "DATEI SORTIERT"=FLAG
11330 REM
11340 RETURN
11350 REM *VTAVER* UNTERROUTINE
11360 LET TNAMFD$=NAMFLD$(L)
11370 LET TMODFD$ = MODFLD$(L)
11380 LET TSTRFD$ = STRFLD$(L)
11390 LET TSTDFD$ = STDFLD$(L)
11400 LET TSTAFD$ = STAFLD$(L)
11410 LET TTELFD$ = TELFLD$(L)
```

```
11420 REM
11430 LET NAMFLD$(L) = NAMFLD$(L+1)
11440 LET MODFLD$(L) = MODFLD$(L+1)
11450 LET STRFLD$(L) = STRFLD$(L+1)
11460 LET STDFLD$(L) = STDFLD$(L+1)
11470 LET STAFLD$(L) = STAFLD$(L+1)
11480 LET TELFLD$(L) = TELFLD$(L+1)
11490 LET INDFLD$(L) = STR$(L)
11500 REM
11510 LET NAMFLD$(L+1) = TNAMFD$
11520 LET MODFLD$(L+1) = TMODFD$
11530 LET STRFLD$(L+1) = TSTRFD$
11540 LET STDFLD$(L+1) = TSTDFD$
11550 LET STAFLD$(L+1) = TSTAFD$
11560 LET TELFLD$(L+1) = TELFD$
11570 LET INDFLD$(L+1) = STR$(L+1)
11580 LET S=1
11590 REM
11600 RETURN
12000 REM *SPEVER* UNTERROUTINE
12010 REM
12020 REM
12030 OPEN "0", #1, "ADBK.DAT"
12040 REM
12050 FOR L=1 TO GROSS-1
12060 PRINT #1, NAMFLD$(L);",";MODFLD$(L);","; STRFLD$(L);",";STDFLD$(L)
12070 PRINT #1,STAFLD$(L);",","; TELFLD$(L);","; INDFLD$(L)
12080 NEXT L
12090 REM
12100 REM
12110 REM
12120 REM
12130 CLOSE #1
12140 REM
12150 RETURN
13000 REM *FNDVER* (FINDE VERZEICHNIS) UNTERROUTINE
13010 PRINT CHR$(12): REM LOESCHE BILDSCHIRM
13020 REM
13030 IF SRTD=0 THEN GOSUB 11200: REM *SRTVER*
13040 PRINT
13050 PRINT
13060 PRINT TAB(9);"SUCHEN EINES VERZEICHNISSES"
13070 PRINT TAB(16); "NACH NAMEN"
13080 PRINT
13090 PRINT TAB (9); "GEBEN SIE DEN KOMPLETTEN NAMEN EIN"
13100 PRINT TAB(7);"(VORNAME, FAMILIENNAME)"
13110 PRINT
13120 PRINT
13130 REM
13140 INPUT "DER NAME LAUTET" ;NAMFLD$ (GROSS)
13150 GOSUB 10200: REM *MODNAM* UNTERROUTINE
13160 LET SUSCHL$=MODFLD$(GROSS)
13170 REM
13180 REM
13190 REM
13200 REM
13210 REM
13220 LET BTM=1
13230 LET TOP=GROSS-1
13240 FOR L=1 TO 1
13250 LET MID=INT((BTM+TOP)/2)
13260 IF MODFLD$(MID) <> SUSCHL$ THEN L=0
13270 IF MODFLD$(MID) < SUSCHL$ THEN BTM=MID+1
13280 IF MODFLD$(MID) >SUSCHL$ THEN TOP=MID-1
13290 IF BTM > TOP THEN L=1
13300 NEXT L
13310 REM
13320 IF BTM > TOP THEN LET CURR=0
13330 IF BTM < = TOP THEN LET CURR=MID
13340 IF CURR=0 THEN GOSUB 13700: REM *NOTVER*
13350 IF CURR=0 THEN RETURN
13360 REM
13370 REM
13380 PRINT CHR$(12)
13390 PRINT
13400 PRINT TAB(13); "*VERZEICHNIS GEFUNDEN*"
13410 PRINT
13420 PRINT "NAME; ", NAMFLD$(CURR)
13430 PRINT "STRASSE: ",STRFLD$(CURR)
13440 PRINT "STADT:",STDFLD$(CURR)
13450 PRINT "STAAT: ",STAFLD$(CURR)
13460 PRINT "TELEFON: ", TELFLD$(CURR)
13470 PRINT
13480 PRINT TAB(7); "DRUECKE BUCHSTABEN-TASTE ZUM AUSDRUCKEN"
13490 PRINT TAB (7);"ODER LEERTASTE,UM FORTZUFAHREN"
13500 FOR I=1 TO 1
13510 LET A$=INKEY$
13520 IF A$="" THEN I=0
13530 NEXT I
13540 IF A$ <> " " THEN GOSUB 13600: REM *LSTCUR*
13550 RETURN
13600 REM *LSTCUR* (LISTE GEGENWAERTIGES VERZEICHNIS) UNTERROUTINE
13610 LPRINT
13620 LPRINT „NAME: ", NAMFLD$(CURR)
13630 LPRINT "STRASSE: ", STRFLD$(CURR)
13640 LPRINT "STADT: ",STDFLD$(CURR)
13650 LPRINT "STAAT: ",STAFLD$(CURR)
13660 LPRINT "TELEFON: ",TELFLD$(CURR)
13670 LPRINT
13680 LPRINT
13690 RETURN
13700 REM *NOTVER* (VERZEICHNIS NICHT GEFUNDEN) UNTERROUTINE
13710 PRINT CHR$(12): REM LOESCHT BILDSCHIRM
13720 PRINT TAB(11);"*VERZEICHNIS NICHT GEFUNDEN*"
13730 PRINT TAB(4);"*IN DER FORM: ";NAMFLD$(GROSS);" *"
13740 PRINT
13750 PRINT TAB(5);"(LEERTASTE,UM FORTZUFAHREN)"
13760 FOR I=1 TO 1
13770 IF INKEY$ <>" " THEN I=0
13780 NEXT I
13790 RETURN
14000 REM *MODVER* (MODIFIZIERE VERZEICHNIS) UNTERROUTINE
14010 REM
14020 PRINT CHR$(12): REM LOESCHT BILDSCHIRM
14030 PRINT
14040 PRINT
14050 PRINT
14060 PRINT
14070 PRINT TAB (3);"*UM EIN VERZEICHNIS ZU MODIFIZIEREN*"
14080 PRINT TAB(3);"*LOKALISIEREN SIE ES ZUERST*"
14090 REM
14100 REM
14110 REM
14120 GOSUB 13030: REM *FNDVER*
      UNTERROUTINE OHNE LOESCHEN DES BILDSCHIRMES
14130 IF CURR=0 THEN RETURN: REM VERZEICHNIS NICHT GEFUNDEN
14140 PRINT
14150 PRINT TAB (3);"MODIFIZIERE NAMEN?"
14160 PRINT
14170 PRINT TAB(3);"DRUECKE RETURN FUER NEUEN NAMEN"
14180 PRINT TAB (3); "ODER LEERTASTE FUER NAECHSTES FELD"
14190 FOR I=1 TO 1
14200 LET A$=INKEY$
14210 IF A$ <> CHR$(13) AND A$ <> " " THEN I=0
14220 NEXT I
14230 IF A$=CHR$(13) THEN INPUT "NEUER NAME";NAMFLD$(CURR)
14240 IF A$=CHR$(13) THEN VMOD=1
14250 IF A$=CHR$(13) THEN SRTD=0
14260 IF A$=CHR$(13) THEN NAMFLD$(GROSS)=NAMFLD$(CURR)
14270 IF A$=CHR$(13) THEN GOSUB 10200:REM *MODNAM* UNTERROUTINE
14280 IF A$=CHR$(13) THEN LET MODFLD$(CURR)=MODFLD$(GROSS)
14290 PRINT
14300 PRINT TAB(3);"MODIFIZIERE STRASSE?"
14310 PRINT
14320 PRINT TAB(3);"DRUECKE RETURN FUER NEUE STRASSE"
14330 PRINT TAB(3);"ODER LEERTASTE FUER NAECHSTES FELD"
14340 FOR I=1 TO 1
14350 LET A$=INKEY$
14360 IF A$<>CHR$(13) AND A$ <> " " THEN I=0
14370 NEXT I
14380 IF A$=CHR$(13) THEN VMOD=1
14390 IF A$=CHR$(13) THEN INPUT "NEUE STRASSE";STRFLD$(CURR)
14400 PRINT
14410 PRINT TAB(3);"MODIFIZIERE STADT?"
14420 PRINT
14430 PRINT TAB(3);"DRUECKE RETURN FUER NEUE STADT"
14440 PRINT TAB (3);"ODER LEERTASTE FUER NAECHSTES FELD"
14450 FOR I=1 TO 1
14460 LET A$=INKEY$
14470 IF A$ <> CHR$(13) AND A$ <> " " THEN I=0
14480 NEXT I
14490 IF A$=CHR$(13) THEN VMOD=1
14500 IF A$=CHR$(13) THEN INPUT "NEUE STADT";STDFLD$(CURR)
14510 PRINT
14520 PRINT TAB(3);"MODIFIZIERE STAAT?"
14530 PRINT
14540 PRINT TAB(3);"DRUECKE RETURN FUER NEUEN STAAT"
14550 PRINT TAB(3);"ODER LEERTASTE FUER NAECHSTES FELD"
14560 FOR I=1 TO 1
14570 LET A$=INKEY$
14580 IF A$ <> CHR$(13) AND A$ <>  " THEN I=0
14590 NEXT I
14600 IF A$=CHR$(13) THEN VMOD=1
14610 IF A$=CHR$(13) THEN INPUT "NEUER STAAT";STAFLD$(CURR)
14620 PRINT
14630 PRINT TAB(3); "MODIFIZIERE TELEFONNUMMER?"
14640 PRINT
14650 PRINT "DRUECKE RETURN FUER NEUE TELEFONNUMMER"
14660 PRINT "ODER LEERTASTE, UM FORTZUFAHREN"
14670 FOR I=1 TO 1
14680 LET A$=INKEY$
14690 IF A$ <> CHR$(13) AND A$ <> " " THEN I=0
14700 NEXT I
14710 IF A$=CHR$(13) THEN VMOD=1
14720 IF A$=CHR$(13) THEN INPUT "NEUE NUMMER";TELFLD$(CURR)
14730 REM
14740 REM
14750 RETURN
15000 REM *LOEVER* (LOESCHE VERZEICHNIS) UNTERROUTINE
15010 REM
15020 PRINT CHR$(12): REM LOESCHT BILDSCHIRM
15030 PRINT
15040 PRINT
15050 PRINT
15060 PRINT
15070 PRINT TAB(3);"*UM EIN VERZEICHNIS ZU LOESCHEN*"
15080 PRINT TAB(3);"*LOKALISIEREN SIE DIESES ZUERST*"
15090 REM
15100 REM
15110 REM
15120 GOSUB 13030: REM *FNDVER*
      UNTERROUTINE OHNE LOESCHEN DES BILDSCHIRMES
15130 IF CURR=0 THEN RETURN: REM VERZEICHNIS NICHT GEFUNDEN
15140 PRINT
15150 PRINT TAB(3); "WOLLEN SIE DIESES VERZEICHNIS LOESCHEN?"
15160 PRINT TAB (3);"WARNUNG — DIES IST NICHT WIDERRUFBAR!"
15170 PRINT
15180 PRINT TAB(5);"DRUECKE RETURN ZUM LOESCHEN"
15190 PRINT TAB(5);"ODER LEERTASTE, UM FORTZUFAHREN"
15200 FOR I=1 TO 1
15210 LET A$=INKEY$
15220 IF A$ <> CHR$(13) AND A$ <> " " THEN I=0
15230 NEXT I
15240 IF A$=" " THEN RETURN
15250 FOR L=CURR TO GROSS-2
15260 LET NAMFLD$(L) = NAMFLD$(L+1)
15270 LET MODFLD$(L) = MODFLD$(L+1)
15280 LET STRFLD$(L) = STRFLD$(L+1)
15290 LET STDFLD$(L) = STDFLD$(L+1)
15300 LET STAFLD$(L) = STAFLD$(L+1)
15310 LET TELFLD$(L) = TELFLD$(L+1)
15320 LET INDFLD$(L) = STR$(L)
15330 NEXT L
15340 LET VMOD=1
15350 LET GROSS=GROSS-1
15360 REM
15370 REM
15380 REM
15390 RETURN
```

Kontakte prüfen

Letzter Schritt bei elektronischen Basteleien ist eine gründliche Prüfung der Lötstellen vor dem Einschalten des Gerätes. Ein einziger Kurzschluß kann empfindliche Bauteile schnell zerstören. Leitungsprüfer kosten weniger als 10 Mark, und sogar ein richtiges Vielfach-Meßgerät ist mit seinem Preis von 50 Mark immer billiger als manches versehentlich „hingerichtete" IC!

Wichtig ist zuerst einmal die Überprüfung der Lötstellen. Äußerlich kann auch eine bei zu niedriger Temperatur gelötete Verbindung akzeptabel aussehen – ohne daß sie tatsächlich Strom leitet. Mit vorsichtigem Ziehen läßt sich die Fehlerquelle häufig schon entlarven. – Je früher man sie findet, um so besser!

Wenn Sie alle Lötstellen auf diese Art geprüft haben, ist das Meßgerät an der Reihe. Das einfachste Gerät dieser Kategorie ist der Kontaktprüfer, wie man ihn im Werkzeug- oder Autozubehörgeschäft bekommt. Damit können Sie testen, ob zwei Kontaktpunkte miteinander leitend verbunden sind.

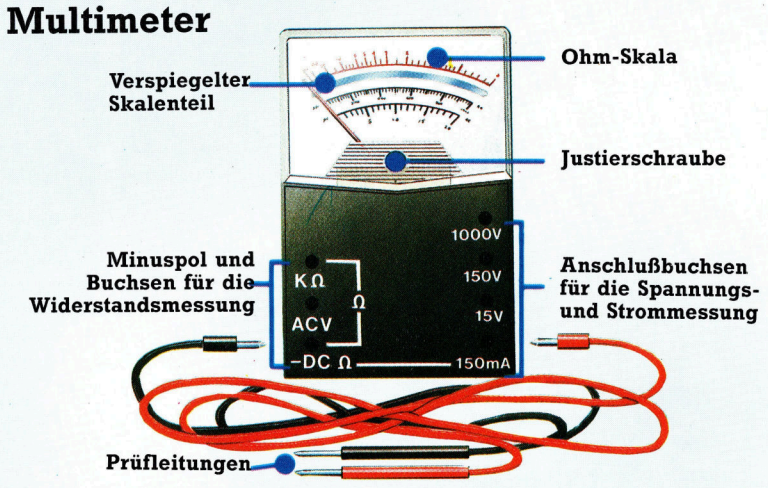
Für unsere Zwecke ist die Krokodilklemme, die sich am Kabelende des Spannungsprüfers befindet, nicht ganz das Richtige: Sie sollten besser eine Lötspitze hineinklemmen und das Ganze mit Isolierband umwickeln.

Ein Vielfachmeßgerät oder „Multimeter" ist schon ein etwas komfortableres Gerät: Damit läßt sich nicht nur auf Spannung testen, sondern man kann auch Widerstände messen. Einheit des elektrischen Widerstandes ist das „Ohm", so genannt nach dem Physiker Georg Ohm (1789–1854), der den Zusammenhang zwischen Strom, Widerstand und Spannung als erster beschrieb.

Eine solide Lötverbindung hat einen sehr geringen Widerstand: Das Anzeigelämpchen Ihres Leitungspüfers sollte also hell leuchten, der Zeiger des Multimeters muß ganz nach rechts ausschlagen. Ist das nicht der Fall, kann man auf eine „kalte" Lötstelle schließen, die noch einmal nachgearbeitet werden muß.

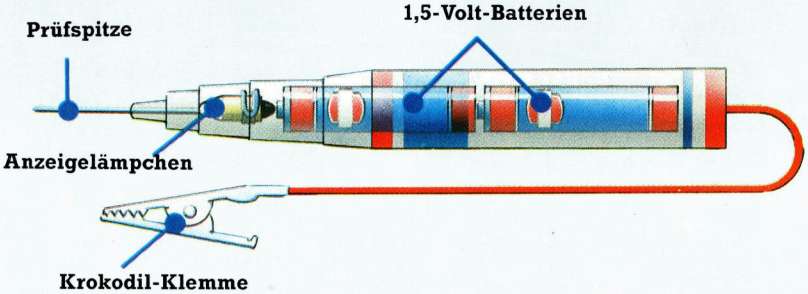
Mit dem Multimeter können Sie auch Spannung und Stromstärke messen: Liegt z. B. über einem 1-Ohm-Widerstand eine Spannung von 1 Volt, so fließt ein Strom von genau 1 Ampere.

Multimeter

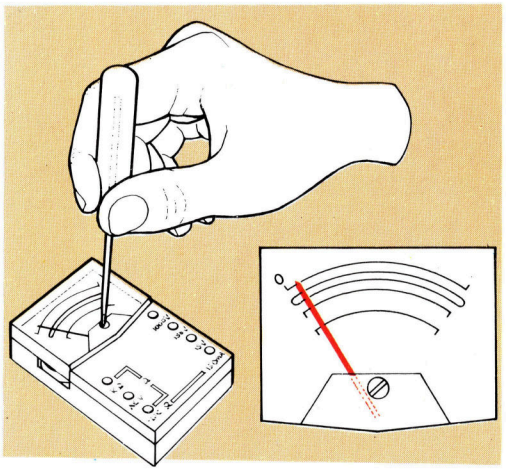


Die Skala eines Multimeters, mit dem man auf Durchgang und Widerstand prüfen kann, ist in „Ohm" geeicht. Zusätzliche Skalenteilungen gibt es meist für Strommessung (in Ampere) und für Spannungsmessungen, die auf der Volt-Skala abgelesen werden. Für ein Multimeter können Sie 50 Mark, aber auch 5000 Mark ausgeben – je nach erforderlicher Meßgenauigkeit. Die Anzeige erfolgt entweder digital, also mit Ziffern, oder analog, d. h. mit einem Zeiger über einer Skala. Die Analoginstrumente arbeiten mit einer Drehspule, an der der Zeiger befestigt ist. Sie sind meist preiswerter als die Digital-Multimeter.

Spannungsprüfer

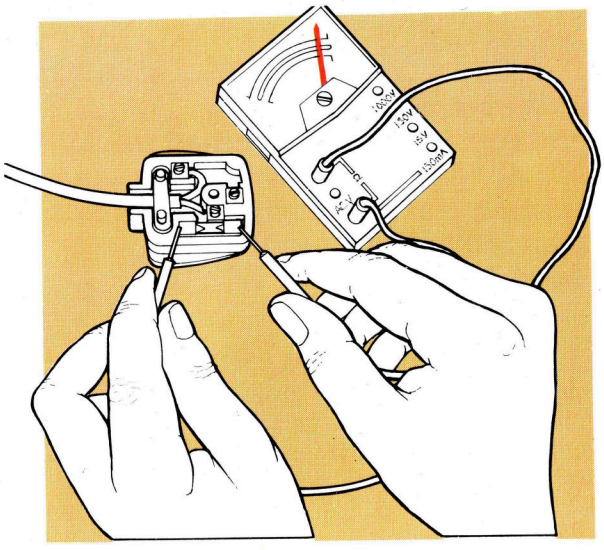


Nullpunkteinstellung



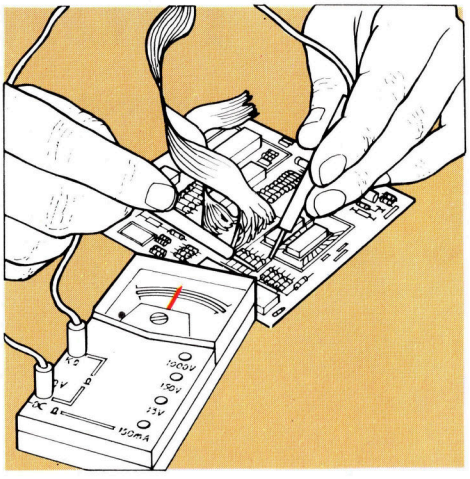
Analoge Meßgeräte, bei denen der Zeiger mechanisch bewegt wird, muß man vor Gebrauch justieren. Meist ist für diesen Zweck eine Schraube nahe beim Drehpunkt des Zeigers vorgesehen. Die seitliche Rändelschraube am Multimeter ermöglicht außerdem den Ausgleich der absinkenden Batteriespannung bei der Widerstandsmessung. Sie wird nachgestellt, wenn der Zeiger nicht mehr ganz nach rechts ausschlägt, wenn sich die Prüfspitzen berühren.

Spannung

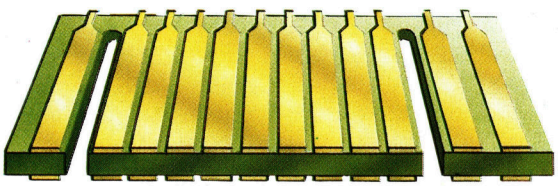


Eine der größten Fehlerquellen in elektrischen Geräten ist eine durchgebrannte Sicherung. Das passiert immer dann, wenn der Stromfluß den festgelegten Sicherungswert überschreitet. Natürlich kann man die Sicherung auf den bloßen Verdacht hin austauschen – besser ist, sie entweder mit dem Spannungsprüfer oder mit dem Multimeter zu testen. Wenn der Strom ausbleibt, die Sicherung aber in Ordnung ist, kann man mit Meßgeräten meist recht schnell die Fehlerursache ermitteln.

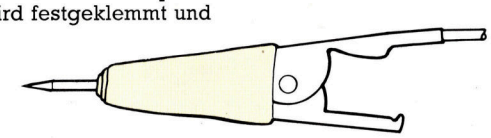
Widerstand



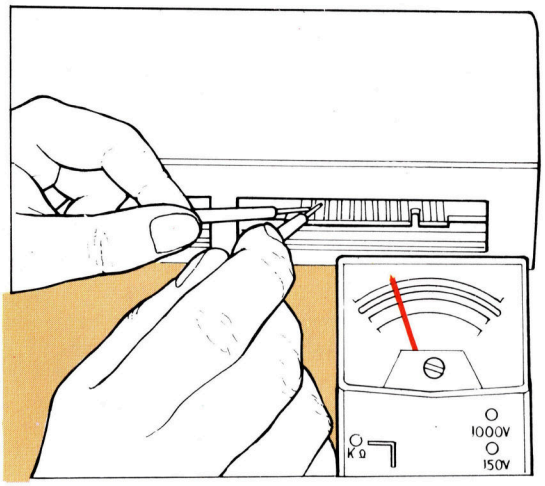
Steckplätze
Steckplätze der unten abgebildeten Art dienen zum Anschluß von Peripheriegeräten an den Computer. Manchmal ist nur ein einziger solcher Steckplatz vorhanden, etwa beim Sinclair Spectrum, oft gibt es aber auch mehrere dieser Schnittstellen. Diese Steckplätze sind in der Leiterplatine integriert.



Krokodil-Klemmen
Die relativ groben Krokodil-Klemmen des Spannungsprüfers lassen sich leicht verfeinern: Eine dünne Lötkolbenspitze wird festgeklemmt und mit einem Isolierband-Streifen an der Klemme befestigt.

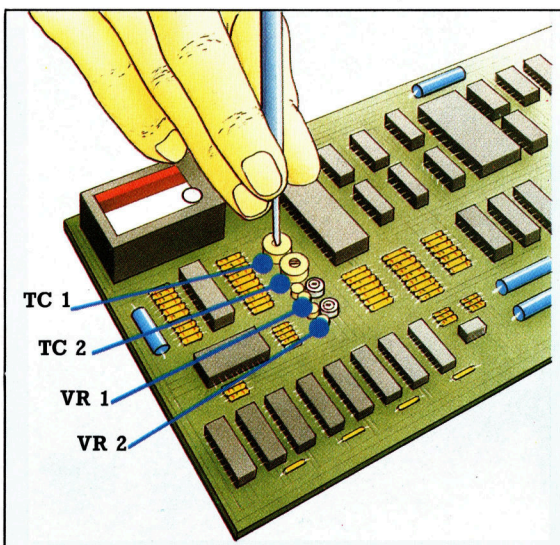


Spannungen messen
Ein gutes Beispiel für die Anwendung des Multimeters ist die Überprüfung der Betriebsspannung an der Platinen-Schnittstelle Ihres Rechners. Im Handbuch ist die Position der Leiterbahnen für 0 Volt, +5 Volt und +9 Volt angegeben. Die schwarze (negative) Prüfspitze wird auf die Bahn für 0 Volt, die rote Spitze auf die +5- oder +9-Volt-Bahn gehalten. Die Skala zeigt, ob die angegebenen Spannungswerte des Rechners korrekt sind.



Achtung!

Während der Garantiezeit darf Ihr Heimcomputer nur vom Hersteller bzw. seiner Vertragswerkstatt geöffnet werden. Nach Eingriffen in das Gerät wird meist jede weitere Garantieleistung verweigert.



Optimierung der Bildwiedergabe

Wie bereits erwähnt, läßt sich bei den ersten Spectrum-Geräten die Qualität der Bildschirmdarstellung verbessern. Die Potentiometer VR 1 und VR 2 regeln die Farbbalance zwischen Rot/Grün und Blau/Gelb. Mit den Drehkondensatoren TC 1 und TC 2 wird die Zeichendarstellung und Farbsättigung beeinflußt. Das Gehäuse wird durch Lösen der fünf sichtbaren Kreuzschlitz-Schrauben an der Unterseite geöffnet. Denken Sie dabei aber an Ihre Garantie – sie verfällt beim Öffnen des Gehäuses! An Spectrum-Rechnern der Version 3 kann man allerdings keine Einstellarbeiten vornehmen.

Floh-Sprünge

Booga-Boo – so ungewöhnlich wie sein Name ist auch das Spielprogramm selbst, das auf dem Spectrum und dem Commodore 64 läuft: Als Floh muß der Spieler den Weg aus einem Höhlensystem suchen. Das für einen Bestseller erstaunlich einfache Programm erhält seinen besonderen Reiz durch eine ausgefeilte Grafik und nicht zuletzt durch die Geschwindigkeit seiner Abläufe.

Zu Beginn des Programms wird der Spieler darüber informiert, daß er sich Cebella 7, dem Planet des ewigen Lebens nähert. Nach der sicheren Landung bleibt wenig Zeit zur Erkundung der Planetenoberfläche mit ihrer interessanten Pflanzenwelt: Der Absturz durch einen tiefen Schacht endet in einer großen Höhle. Schafft es der Spieler, sich daraus wieder zu befreien?

Vorsprünge im wild-bunten Felsgestein dienen als Kletterhilfe in einer Umgebung, die an Bilder von Hieronymus Bosch erinnert: Es wimmelt von bunten Pilzen und Blumen, Spinnen sitzen an der Höhlendecke. Die Vorsprünge sollen bei der Flucht vor dem Floh-fressenden Drachen benutzt werden. In der Commodore-Version sind außerdem noch die Venusfliegenfallen zu meiden.

In beiden Spiel-Versionen eröffnet der Bildschirm nur ein Fenster auf die gesamte Umgebung, die sich daher beim Klettern ständig verändert. Die ganze Höhle mißt ca. drei mal fünf Bildschirmgrößen, es gibt also einiges zu entdecken. Die unterschiedlichen Routen werden durch listig plazierte Vorsprünge zusätzlich erschwert.

Der originelle Schauplatz des Spiels kann leider über einige Mängel nicht hinwegtäuschen: Der Spieler hat nur ein einziges „Leben", und sobald er dieses verliert, erscheint erneut die Auftaktsequenz. Der Drache gibt sich zudem mit viel Erfolg alle Mühe, dem Spieler das Leben sauer zu machen bzw. zu verkürzen . . .

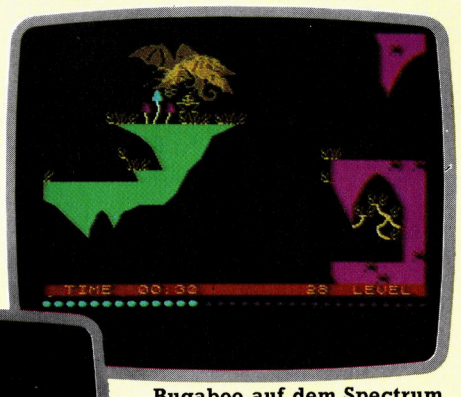
Die Felsvorsprünge werden beim Fortgang des Spieles zwar immer schwerer erreichbar, neue Gefahren tauchen aber nicht auf. Nun verbraucht zwar die Speicherung mehrerer Bildschirm-Inhalte viel Raum im Memory und läßt wenig Platz für umfangreiche Regeln, andere Programme zeigen aber, daß man doch mehr unterbringen könnte. Die Commodore-Version ist zwar in der Grafik besser als die Spectrum-Ausführung, nutzt jedoch den zusätzlichen Speicherplatz ebenfalls nicht vollständig.

Beim Spectrum läßt sich das Programm über die Tasten 1 und 0 – rechts/links – steuern: Je länger sie gedrückt werden, um so weiter der Sprung. Beim Commodore braucht man zum Spielen den Joystick – für schnelle Spielabläufe in jedem Fall die bessere Lösung.

Wenn man von den oben erwähnten Mängeln einmal absieht, hat Booga-Boo genügend Eigenschaften, die lobenswert sind.

Von Fels zu Fels . . .

Das hervorstechende Merkmal des Spiels ist seine detaillierte Grafikdarstellung. Um aus einer tiefen Höhle zu entkommen, muß der als Floh auftretende Spieler von Felsvorsprung zu Felsvorsprung hüpfen. Die Version für den C 64 ist in der Grafik besser.



Bugaboo auf dem Spectrum

Bugaboo auf dem Spectrum

Booga-Boo auf dem Commodore 64

Bugaboo (The Flea): Für den 48K Spectrum

Booga-Boo (The Flea): Für den Commodore 64

Hersteller: Quicksilva

Autoren: Indescomp & Microbyte

Joysticks: Nur für die Commodore-Version

Format: Cassetten

Die Musik-Macher

In dieser neuen Serie befassen wir uns detailliert mit MIDI – dem Musical Instrument Digital Interface. Weitere Themen werden die digitale Manipulation von Klängen durch Veränderung ihrer Abfolge, Modulationssynthese und natürliche Klangmuster sein. Damit lassen sich Ergebnisse erzielen, die vor kaum einem Jahrzehnt nicht einmal vorstellbar waren.

Neue Musik verlangt nach neuer Notation. Stockhausens Partitur mit ihren bildlichen Darstellungen von Klängen und grafischen Zeit-/Synchronisations-Anweisungen haben mit klassischen Partituren nichts mehr zu tun. Sie erinnern an Diagramme elektronischer Schaltungen.

Seit vielen Jahrhunderten ist die Wechselbeziehung zwischen Musik und der Wissenschaft von Zahlen und Proportionen bekannt. Der griechische Mathematiker Pythagoras ließ eine Vielzahl von Schmiedehämmern wiegen, um herauszufinden, warum sie „gleichstimmig" klangen, wenn man mit ihnen auf einen Amboß schlug. Er stellte dabei fest, daß ein Hammer, der halb so schwer wie ein anderer war, einen Ton erzeugte, der genau die doppelte Frequenz hatte bzw. eine Oktave höher lag. Damit wurde erstmals die Verwandtschaft der Töne in eine mathematische Regel gefaßt.

Derzeit sorgt MIDI, das Musical Instrument Digital Interface, für einige Neuerungen im Musikgeschehen. Diese Schnittstelle wurde geschaffen, um jedem digitalen System einschließlich der Microcomputer die Möglichkeit zu geben, die Funktionen des anderen zu kontrollieren. Da die Mehrzahl elektronischer Musikinstrumente heute „digital" ist, eröffnet sich dem Computerbesitzer so ein neues Reich aufregender Möglichkeiten.

Doch MIDI ist keine Zauberkiste. Ein Computeranwender wird damit nicht über Nacht zu einem Vangelis oder einem Stevie Wonder. Musikalische Fähigkeiten und Phantasie bleiben unverändert Voraussetzung für optimale Ergebnisse, sei die Musik nun mit einer Reihe verbundener Synthesizer erzeugt oder auf einer akustischen Gitarre.

Die Geschichte des MIDI

Um den Musikinstrumententypus, für den MIDI entwickelt wurde, und elektronische Musik generell zu verstehen, müssen wir ein halbes Jahrhundert zurückgehen. Schon vor Beginn des Zweiten Weltkrieges hatten Musiker begonnen, mit einfachen „Sinus-Ton-Generatoren" zu experimentieren. Diese elektrischen Geräte brachten eine Metallsaite zum Schwingen und erzielten so einen Dauerton, dessen Höhe variierte. Dieser Sound wurde häufig für Science-Fiction-Filmmusiken der fünfziger Jahre verwendet, um eine unheimliche oder futuristische Atmosphäre zu vermitteln. Die in den dreißiger Jahren aufkommenden Hammond-Orgeln waren elektronisch – und basierten auf dem zuvor beschriebenen Sound.

Doch erst die elektronische Entwicklung während des Zweiten Weltkrieges, und dabei speziell die Entwicklung des Tonbandgerätes in Deutschland, gab Musikern die Möglichkeit, Klänge auf ganz andere Art zu schaffen und zu manipulieren. Dies war durch Zusammenfügen analoger Tonbandaufnahmen machbar, durch das Verbinden von Bandabschnitten, auf denen Geräusch, „Musik" oder ähnliches bereits aufgezeichnet worden war. Die Elemente fügte man sorgfältig zusammen und schuf so Kollagen von Klangereignissen. Mit dieser „neuen

Pioniergeist

Brian Eno, bekannt geworden Anfang der siebziger Jahre durch seine Arbeit mit Roxy Music, war einer der ersten Synthesizer-Pioniere. Seit 1973, als er die Gruppe verließ, gehört er zu den Vorreitern der „Avantgarde"- und elektronischen Musik. Er hat auch mit Musikern wie David Bowie und Robert Fripp zusammengearbeitet und entwickelte mit seinem Bruder die Begleitmusik für den Mondlandungs-Archivfilm der NASA.

Musik" wurden alle Regeln herkömmlicher Musik durchbrochen. Manche Zuhörer waren fasziniert davon, andere fanden derartige Klanggebilde „fürchterlich".

Etwa zur selben Zeit gelang es, Geräte zu bauen, mit denen einfach erzeugte Oszillator-Töne variiert und verzerrt werden konnten, und deren Ergebnis sich filtern und modulieren ließ. In den fünfziger Jahren arbeiteten Komponisten wie der Deutsche Stockhausen in kleinen Studios, die Rundfunksendern angegliedert waren, und erzeugten „reine" elektronische Musik. Pierre Schaeffer war einer der Pioniere jener Musikrichtung, die er „musique concrète" (konkrete Musik) nannte. Dabei handelte es sich um Musikcollagen, die aus Geräuschen der Umwelt zusammengesetzt waren. Tontechniker des ORTF, des französischen Rundfunks in Paris, arbeiteten eng mit ihm zusammen.

Bei den Bell Telephone Laboratories in Amerika entstand das, was man den ersten Synthesizer nennen könnte. Das „Ding" beanspruchte mehrere Räume. Es war zu dem Zweck gebaut worden, die Synthese der menschlichen Stimme zu analysieren. Die Gesellschaft wußte, daß ihre Mitarbeiter in der Telefon-Vermittlung in verschiedenen Teilen Amerikas Probleme aufgrund unterschiedlicher Akzente und Aussprachen hatten. Ergebnis waren überdurchschnittlich viele falsche Verbindungen. Man hoffte, was für jene Zeit natürlich im nachhinein sehr optimistisch anmutet, das Problem mit Hilfe einer synthetisierten, also „künstlichen", allgemein verständlichen Stimme lösen zu können.

Das erste nennenswerte Unternehmen in Sachen Computermusik fand 1957 statt. Damals gab Lejaren Hiller eine Reihe entsprechender Instruktionen in den „Illiac"-Computer der Universität von Illinois ein. Diese Befehle hatte man in vier Gruppen technischer Daten aufgelöst, die dann in die herkömmliche musikalische Schreibweise übertragen wurden. Das Ergebnis war eine Satz-Komposition für ein Streichquartett, betitelt „Illiac Suite". Die Musik selbst, obwohl für eine Aufführung mit Cello, Viola und zwei Violinen arrangiert, klingt wirr und irgendwie konzeptlos.

Einige Jahre später schuf Hiller ein anderes Werk, diesmal unter Verwendung eines IBM 7090 Computers. Er entwickelte ein Programmierschema mit der Bezeichnung MUSICOMP (MUsic SImulator-Interpreter for COMpositional Procedures – etwa: Musik-Simulator-Interpreter für Kompositionszwecke). Es ermöglichte größere Flexibilität und mehr Abwechslung bei der Arbeit an Kompositionen. Hiller nannte sein Werk „Computer-Kantate". Es wurde für eine Stimme geschrieben, die zu auf Band aufgezeichneten elektronischen Klängen sprach. Damit hatte Hiller seinen musikalischen Zunftgenossen demonstriert, daß ein Computer durchaus effektiv für kreatives Arbeiten eingesetzt werden konnte.

Seine Arbeit war nur ein kleiner Teil der umfangreichen Forschungen und Experimente, die in den folgenden Jahren an amerikanischen Universitäten durchgeführt wurden. Ein anderer Pionier, John Chowning, verwendete später einen Computer, um die Veränderung eines Klanges zu erforschen, wenn sich die Klangquelle bewegt. Seine Forschungsergebnisse flossen unmittelbar in jene Art von Yamaha-Synthesizern ein, die Mitte der achtziger Jahre gebaut wurden.

Mit Ausnahme der Anwendung in Science-Fiction-Filmen, hatte elektronische Musik jahrelang einen ähnlichen Stellenwert wie klassische Musik. Veränderungen in Art und Technik wurden dem Publikum dann aber langsam durch Avantgarde-Komponisten bewußt. Ein typisches Konzert mit „neuer Musik" in den sechziger Jahren sah mehrere Musiker vor, von denen einige klassische Instrumente spielten, andere lediglich die Funktion hatten, den Klang dieser Instrumente durch Frequenzteilung und Filter zu modifizieren.

Klang und Wirkung

Ergänzend zu ausführlichen Beschreibungen wie etwa die exakte Position von Mikrofonen und Filtervarianten versuchten die Komponisten, bildhaft zu verdeutlichen, wie der zu erzeugende neue Klang in der Aufführung zu wirken hatte. In manchen Fällen spielten Musiker von Musikblättern (von Notenblättern kann man wohl nicht reden), die mehr wie das Schmierpapier eines Grafikers aussahen. Dieses Problem – wie man Aufführungsanweisungen gibt, welche Sprache man verwendet und wie sich das Ergebnis präzise sichtbar machen läßt – gibt es auch noch heute.

Ab 1960, als die Entwicklung der Popmusik fortschritt und sich die Jugendkultur entwikkelte, verbrachten Musiker immer mehr Zeit in

den Aufnahmestudios und begannen dort, mit elektronischer Musik zu experimentieren. Das klassische Beispiel sind die Beatles, die in George Martin nicht nur einen hervorragenden Toningenieur fanden, sondern auch einen Musiker, der die Entwicklungen in der klassischen Musik aufmerksam verfolgt hatte. Er ermutigte die Beatles, das Studio als totales Musikinstrument zu benutzen, und ziemlich bald schon arbeiteten sie mit Bandschnitt-Technik unter Einbeziehung synthetisch erzeugter Klänge.

Klangmodifikationen

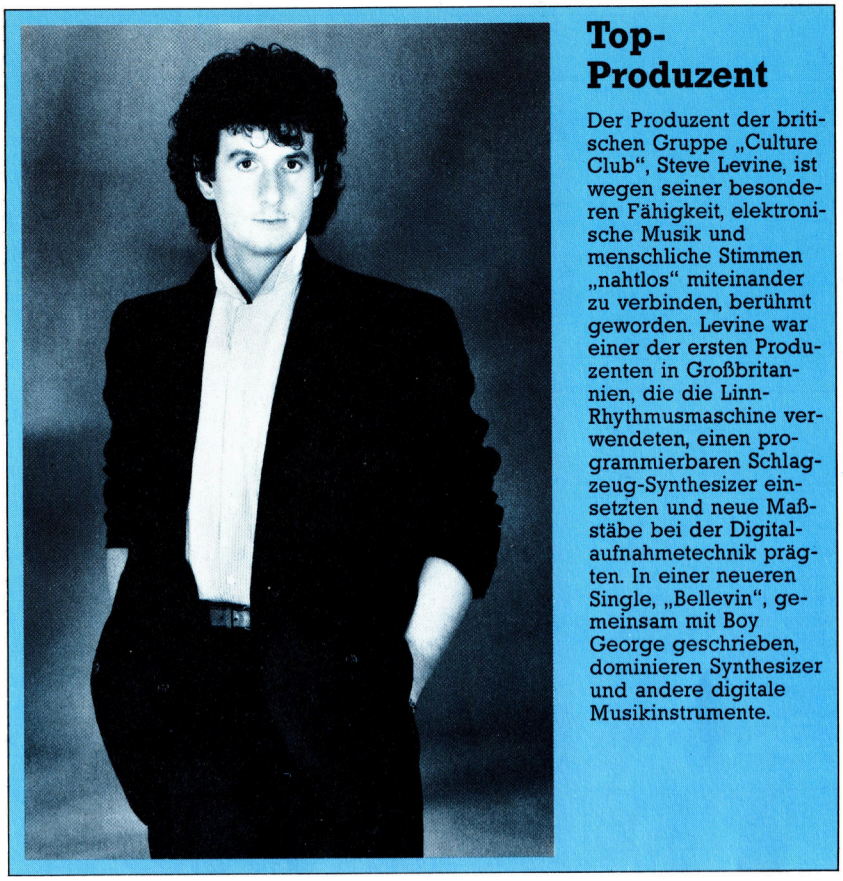
Es gab Musiker, die sich durch die Verwendung bestimmter Klangerzeuger einen Namen machten. So baut der Stil des Gitarristen Jimmy Page auf dem der farbigen amerikanischen Musiker der vierziger Jahre auf. Doch durch Verwendung einer Reihe von Verzerrern entstand ein Sound, der eben eindeutig für „Led Zeppelin" steht. Dies, verbunden mit Jimi Hendrix' Technik, der viel mit Rückkoppelung und jenen Filtern arbeitete, die heute als „Wah-wah-Pedal" bekannt sind, war Gundlage für die Heavy-Metal-Musik.

Im Laufe der siebziger Jahre verwendete man bei den verschiedenen Klangerzeugern und Klangmodifikations-Einheiten Transistoren. So wurden die Geräte kleiner, tragbar und waren nicht mehr an ein Studio gebunden. Gitarristen konnten live spielen und dabei verschiedene Arten von Effekt-Pedalen einsetzen. Und nur wenig später gab es handliche Synthesizer, die Organisten und Pianisten die Bühnenverwendung möglich machten.

Normalerweise sind diese Synthesizer mit stimmbaren Oszillatoren ausgestattet, verschiedenen Generatoren (zur Erzeugung der unterschiedlichen Hüllkurven), variablen Filtern, Ring-Modulatoren (die Signale in neue Frequenzen zerlegen können) und Rauschgeneratoren. Was Jimi Hendrix für die Gitarristen entwickelt hatte, schuf Brian Eno für die Synthesizer-Spieler.

Etwa zur gleichen Zeit wurde die Ausrüstung der Studios immer ausgefeilter, da Musiker nach Möglichkeiten suchten, Elemente in ihre Musik einzubeziehen, die auf der Bühne nicht erzeugbar waren. Ein Mischpult beispielsweise, mit dem man heute 16 oder 24 Spuren gleichzeitig verarbeiten kann, war für Tourneen zu groß. In Amerika und Großbritannien entwickelte sich eine neue Generation von Produzenten. Sie begannen ihre Karriere meist als Tontechniker und waren mit dem Equipment vertrauter als die Musiker, die sie dafür bezahlten, daß sie „den richtigen Sound" herausholten.

In Jamaica begannen Toningenieure damit, das Mischpult selbst als Instrument zu benutzen. Fertige Songs, die auf Mehrspurband aufgezeichnet worden waren, wurden in ihre individuellen Rhythmusspuren zerlegt. Die Gesangs- oder Instrumentalsequenzen nutzte man dann als Rohmaterial, das beliebig eingefügt oder herausgenommen und außerdem modifiziert werden konnte.



Top-Produzent

Der Produzent der britischen Gruppe „Culture Club", Steve Levine, ist wegen seiner besonderen Fähigkeit, elektronische Musik und menschliche Stimmen „nahtlos" miteinander zu verbinden, berühmt geworden. Levine war einer der ersten Produzenten in Großbritannien, die die Linn-Rhythmusmaschine verwendeten, einen programmierbaren Schlagzeug-Synthesizer einsetzten und neue Maßstäbe bei der Digitalaufnahmetechnik prägten. In einer neueren Single, „Bellevin", gemeinsam mit Boy George geschrieben, dominieren Synthesizer und andere digitale Musikinstrumente.

Durch Einführung des digitalen Synthesizers wurde die Eingabe nichtelektronischer Klänge möglich. Diesen Prozeß nennt man „Sampling". Rhythmusmaschinen wie die von „Linn" waren begehrte Studioinstrumente und fanden bald auch Anwendung auf der Bühne. Heute ist „Sampling" wie die Manipulation des Sounds eine Selbstverständlichkeit. Gut ausgerüstete Studios und vergleichbares Bühnen-Equipment umfassen mehr digitale als analoge „Instrumente". Erfolgreiche Gruppen wie „Culture Club" kombinieren ihre eigenen musikalischen Fähigkeiten mit den digitalen Produktionstechniken etwa von Steve Levine, der über einen „Park" von Instrumenten und Studioeinheiten verfügt, deren Wert in die Hunderttausende geht.

Die Notwendigkeit eines Interfaces, das die Verbindung eines Instruments mit dem anderen herstellt oder die Möglichkeiten eines Synthesizers durch Hinzufügen eines Computers erweitert, brachte die Musikinstrumentenhersteller zusammen. Die erste MIDI-Spezifikation, ein Standard also, ist seit April 1983 gültig. Seitdem hat es kaum ein Hersteller riskiert, einen nicht MIDI-kompatiblen Synthesizer auf den Markt zu bringen. Über Hintergrund und Entwicklung von MIDI berichten wir an anderer Stelle noch ausführlich.

Texte und Zeichen

In dieser Folge wird untersucht, wie BASIC-Programme im Arbeitsspeicher untergebracht und vom BASIC-Interpreter ausgeführt werden. Diese Informationen geben wesentlichen Aufschluß darüber, wie der Maschinencode den Speicher handhabt.

Wie Sie bereits wissen, ist von dem Augenblick an, in dem Sie Ihr Gerät einschalten, ein hochentwickeltes Programm aktiv: das Betriebssystem. Es ist in einigen ROM-Chips im Inneren des Computers eingebrannt und steuert die Grundfunktionen der Maschine. Beispielsweise bringt das Betriebssystem das Bild auf den Monitor, handhabt Druckerausgabe und Diskettenspeicherung und überprüft die Tastatur auf Eingaben. Das Betriebssystem sieht alle einkommenden Daten als Impulse an, die es verarbeiten muß.

Eines der Betriebssystemmodule ist der BASIC-Interpreter, mit dem die Texte von BASIC-Programmen analysiert und ausgeführt werden. Bei der Eingabe erkennt das Betriebssystem an der Zeilennummer (am Anfang jeder neuen Zeile), daß ein BASIC-Programm vorliegt. Es bringt die Zeichen der Programmzeilen (mit wenigen Ausnahmen) dann als einzelne Bytes in einem speziell für BASIC-Programmtexte reservierten Speicherbereich unter. Der Interpreter wiederum versteht BASIC-Programme nur als Daten, die verarbeitet werden müssen. Bei der Eingabe von RUN wird ihm vom Betriebssystem die Steuerung übergeben, und er bearbeitet nun selbständig die BASIC-Daten.

Der Interpreter verändert das Programm nicht, sondern übersetzt es in die Maschinensprache und läßt es ablaufen. Da er allen Befehlen gehorcht, können Sie auch den Speicherbereich abfragen, der das Programm enthält. Wenn möglich führt der Interpreter die Programmbefehle aus. Ist dies nicht machbar, meldet er SYNTAX ERROR, OVERFLOW ERROR oder ähnliches.

Mit Ausnahme der Schlüsselworte werden BASIC-Programme vom Betriebssystem zeichenweise gespeichert. Wird die Buchstabenkombination (oder auch die Zeichen, Zahlen oder elektrischen Impulsfolgen) eines Schlüsselwortes erkannt, dann tauscht das Betriebssystem dieses Wort gegen eine Codezahl mit der Länge eines Bytes aus, die „Token" (Ersatzzeichen) genannt wird. Dieser Vorgang spart Speicherplatz (der Befehl RESTORE beispielsweise würde sonst sieben Zeichen belegen) und vereinfacht somit die Arbeit des Interpreters.

Obwohl alle Maschinentypen unterschiedliche Tokensysteme verwenden, sind den Token jedoch fast immer Zahlenwerte über 127 zugeordnet. Die ASCII-Codezahlen für druckbare Zeichen liegen zwischen 32 und 127. Trifft der Interpreter auf ein Token-Byte, ruft er einfach die entsprechende Unterroutine auf.

Wenn Sie LIST aufrufen, um Ihr Programm anzusehen, untersucht das Betriebssystem die einzelnen Bytes des Textbereiches. Trifft es auf einen Wert über 127, dann interpretiert es diesen Wert als Token. Im Speicher befindet sich ebenfalls eine Liste aller BASIC-Schlüsselworte in ASCII-Darstellung. Der Tokenwert zeigt auf eines dieser Worte, und das Betriebssystem setzt es anstelle des Tokens für die Bildschirmdarstellung des Programms ein.

ASCII-Codes, größer als 127, können jedoch auf einigen Computern im Original dargestellt werden, wenn sie in Anführungszeichen eingeschlossen sind. Ohne Anführungszeichen werden sie als Token interpretiert.

Die Codes 32 bis 127 enthalten die druckbaren ASCII-Zeichen, die (mit wenigen Ausnahmen) auf allen Computern identisch sind. In den Handbüchern der einzelnen Maschinen sind die zugehörigen ASCII-Codes genau beschrieben.

ASCII CODE	ASCII	ASCII CODE	ASCII
32	Space	80	P
33	!	81	Q
34	"	82	R
35	#	83	S
36	$	84	T
37	%	85	U
38	&	86	V
39	'	87	W
40	(	88	X
41	)	89	Y
42	*	90	Z
43	+	91	[
44	,	92	\
45	-	93	]
46	.	94	^
47	/	95	_
48	0	96	`
49	1	97	a
50	2	98	b
51	3	99	c
52	4	100	d
53	5	101	e
54	6	102	f
55	7	103	g
56	8	104	h
57	9	105	i
58	:	106	j
59	;	107	k
60	<	108	l
61	=	109	m
62	>	110	n
63	?	111	o
64	@	112	p
65	A	113	q
66	B	114	r
67	C	115	s
68	D	116	t
69	E	117	u
70	F	118	v
71	G	119	w
72	H	120	x
73	I	121	y
74	J	122	z
75	K	123	{
76	L	124	!
77	M	125	}
78	N	126	~
79	O	127	Delete

Die Zeilenlänge

Generell enthalten die ersten drei oder vier Bytes einer BASIC-Programmzeile die Zeilennummer und Informationen über die Zeilenlänge (siehe Bild auf der nächsten Seite). Das Format kann sich allerdings von Computer zu Computer unterscheiden. Die Zeilennummer wird dabei nicht als ASCII-Code gespeichert (beispielsweise würde die Zeile 61030 allein für ihre Nummer schon fünf Bytes benötigen), sondern als eine aus zwei Bytes bestehende Ganzzahl (Integer). Den Zahlen von 0 bis 255 (die sich in einem einzigen Byte unterbringen lassen) wird dabei ein Byte mit dem Wert Null vorangestellt, um das Zwei-Byte-Format zu erhalten. Größere Zahlen werden ähnlich wie bei der Seitenadressierung gespeichert: Das erste Byte wird mit 256 multipliziert und mit dem zweiten Byte addiert. Die Zahl 1000 wird daher als 3232 (3 * 256 + 232 = 1000) gespeichert. Diese beiden Bytes befinden sich in Programmzeilen auf immer der gleichen Stelle, die jedoch von System zu System variieren kann.

Die Information über die Zeilenlänge ist auf dem Acorn B in einem einzigen Byte und auf dem Spectrum in zwei Bytes gespeichert. Der Inhalt dieser Bytes gibt die Zahl der in der Zeile enthaltenen Bytes einschließlich der Bytes für Zeilennummer und Länge an. Ist die

Adresse des ersten Bytes einer BASIC-Programmzeile bekannt, brauchen Sie zu diesem Wert nur den Inhalt des Bytes der Zeilenlänge zu addieren, um die Anfangsadresse der nächsten Programmzeile zu erhalten. Da ein Byte nur Zahlen bis 255 speichern kann, lassen sich in einer Programmzeile des Acorn B auch nur 255 Zeichen unterbringen. Versuchen Sie einmal, mit dem Programm Mempeek festzustellen, ob diese Grenze sich auf die Anzahl der Zeichen bezieht, die Sie in einer Programmzeile eingeben können, oder auf die Länge der Zeile, die gespeichert wird.

Auf dem Commodore wurde das Byte für die Zeilenlänge durch eine aus zwei Bytes bestehende „Verbindungsadresse" (Link Address) ersetzt. Diese Verbindungsadresse enthält die Anfangsadresse der nächsten Programmzeile.

Dabei ist interessant, daß der Acorn B und der Spectrum sich die Anfangsadresse der nächsten Zeile aus der aktuellen Adresse und der Zeilenlänge errechnen (ein Verfahren, das zwar langsam ist, aber ein Byte einspart). Der Commodore dagegen speichert diese Adresse und benötigt dafür ein zusätzliches Byte, ist aber schneller. Dieses Beispiel zeigt, daß es keine absolute Formel für die Konstruktion eines Computers gibt, sondern nur die individuellen Methoden der einzelnen Designer. Das spiegelt auch die Überlegungen der Computer-Konstrukteure wider, die bei der Planung eines Gerätes zwischen zwei grundlegenden Spezifikationen wählen müssen: entweder eine schnelle, jedoch teure Maschine

ASCII-CODE	ASCII	Commodore	Spectrum	Acorn B
0	NUL – keine Auswirkung	–	–	Null
1	SOH – Startsignal einer Zeichenfolge	–	–	Nächstes Zeichen zum Drucker senden
2	STX – Start einer Zeichenfolge	–	–	Drucker anschalten
3	ETX – Ende einer Zeichenfolge	–	–	Drucker ausschalten
4	EOT – Ende der Übertragung	–	–	Text/Grafikcursor trennen
5	ENQ – Signalanfrage	"Weiß"-Taste	–	Text/Grafikcursor verbinden
6	ACK – Zeichen empfangen	–	PRINT	VDU-Treiber einschalten
7	BEL – Klingelzeichen	–	EDIT	kurzes Klingelzeichen
8	BS – Backspace	CBM-Taste abschalten	Cursor links	Backspace Cursor
9	HT – Horizontaler Tabulator	CBM-Taste anschalten	Cursor rechts	Vorwärtsschritt des Cursors
10	LF – Zeilenvorschub	–	Cursor eine Zeile nach unten	Cursor eine Zeile nach unten
11	VT – Vertikaler Tabulator	–	Cursor eine Zeile nach oben	Cursor eine Zeile nach oben
12	FF – Formularvorschub	–	Löschtaste	Textbereich löschen
13	CR – Wagenrücklauf	RETURN	ENTER	Return
14	SO – Dauerumschaltung	Kleinbuchstaben einschalten	Zahl	Seitenschaltung ein
15	SI – Rückschaltung	–	–	Seitenschaltung aus
16	DLE – Datenübertragungsumschaltung	–	INK	Grafikbereich löschen
17	DC1 – Gerätesteuerung 1	Cursor eine Zeile nach unten	PAPER	Textfarbe definieren
18	DC2 – Gerätesteuerung 2	Umgekehrte Zeichendarstellung an	FLASH	Grafikfarbe definieren
19	DC3 – Gerätesteuerung 3	Cursor auf HOME-Position	BRIGHT	logische Farbe definieren
20	DC4 – Gerätesteuerung 4	Löschtaste	INVERSE	auf Systemeinstellung der logischen Farbe zurückstellen
21	NAK – Zeichen nicht empfangen	–	OVER	VDU-Treiber abschalten
22	SYN – Synchronisierung	–	AT	Darstellungsart auswählen
23	ETB – Ende eines Datenblocks	–	TAB	dargestelltes Zeichen umprogrammieren
24	CAN – ungültig	–	–	Grafikfenster definieren
25	EM – Ende der Aufzeichnung	–	–	Plot m,x,y
26	SUB – Zeichen ersetzen	–	–	auf Systemeinstellung der Fenster zurückstellen
27	ESC – Codeumschaltung	–	–	Null
28	FS – Filetrennung	"Rot"-Taste	–	Textfenster definieren
29	GS – Gruppentrennung	Cursor ein Zeichen nach rechts	–	Grafikmode definieren
30	RS – Untergruppentrennung	"Grün"-Taste	–	bewege Textcursor
31	US – Teilgruppentrennung	"Blau"-Taste	–	Textcursor auf Koordinaten x,y stellen

Der „American Standard Code for Information Interchange" (Amerikanische Standardcodierung für den Informationsaustausch) ordnet jedem Standardzeichen einen Zahlencode zwischen 0 und 127 zu. Die Zeichen der Codes von 0 bis 31 lassen sich nicht drucken, werden aber als Steuersignale für den Betrieb von Peripheriegeräten wie Bildschirm und Drucker eingesetzt. Die Bedeutung dieser Codes ist daher von Maschine zu Maschine verschieden – wie aus der Tabelle ersichtlich. Viele Geräte – in unserer Tabelle der Commodore und der Spectrum – nutzen einen Teil der Codes nicht (mit -- gekennzeichnet).

oder ein langsames, aber preiswertes Gerät. Das Optimum ergibt sich natürlich aus der Verbindung beider positiver Eigenschaften. Vor eine ähnliche Entscheidung sind auch Sie gestellt, wenn Sie BASIC-Programme auf Maschinen mit begrenztem Speicher (zum Beispiel Grundversionen des VC 20 und ZX81) schreiben und das Verhältnis zwischen Ausführungsgeschwindigkeit und Speicherausnutzung gründlich abwägen müssen.

Schließlich hat jede Programmzeile entweder am Anfang oder am Ende eine Markierung. Auf dem Acorn B beginnt jede Zeile mit einem Byte, das die Zahl 13 enthält (ASCII-Bode für „Return" – Wagenrücklauf). Der Spectrum markiert damit das Zeilenende. Das Commodore-BASIC schließt seine Zeilen mit dem „Null"-Byte (ASCII für "") ab.

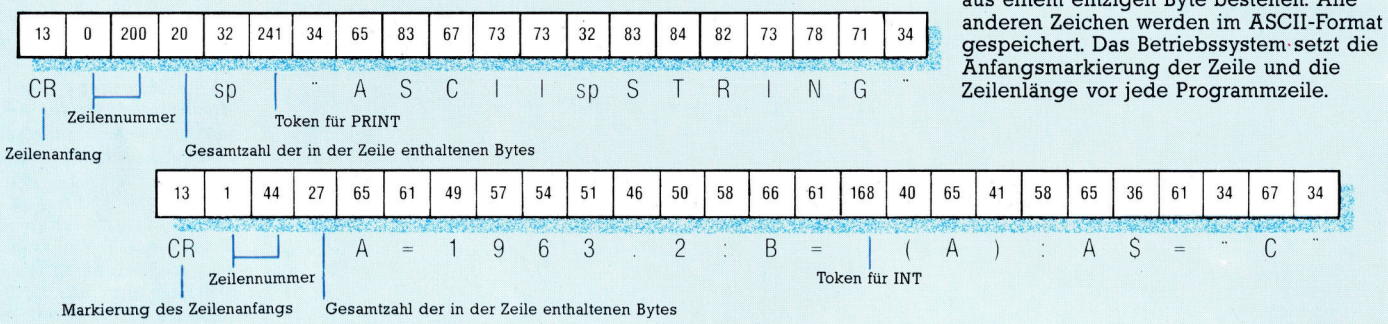
Die Speicherung von BASIC-Programmen

Jede Maschine speichert BASIC-Texte nach einer eigenen Methode. Im folgenden haben wir drei Beispiele aufgeführt.

Acorn B

```
200 PRINT"ASCII STRING"
300 A=1963.2:B=INT(A):AS="C"
```

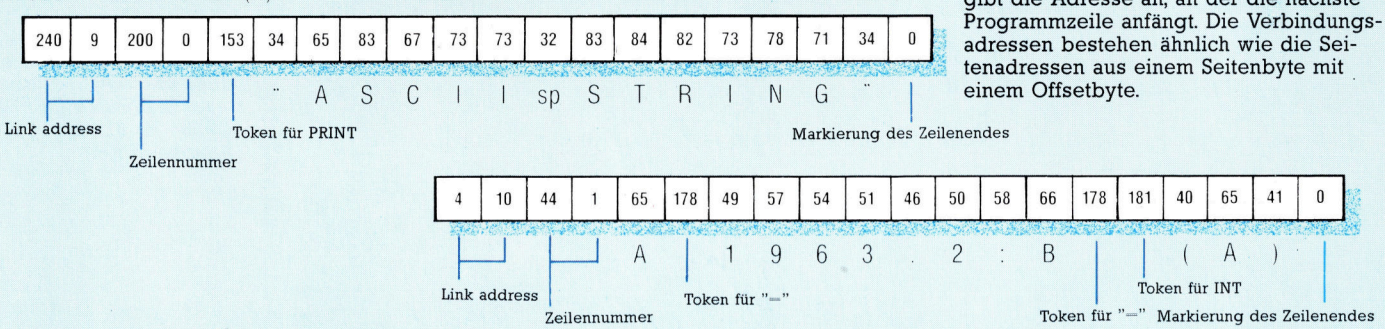
In diesem Beispiel werden die Schlüsselwörter des BASIC von Token ersetzt, die aus einem einzigen Byte bestehen. Alle anderen Zeichen werden im ASCII-Format gespeichert. Das Betriebssystem setzt die Anfangsmarkierung der Zeile und die Zeilenlänge vor jede Programmzeile.



Commodore 64

```
200 PRINT "ASCII STRING"
300 A=1963.2:B=INT (A)
```

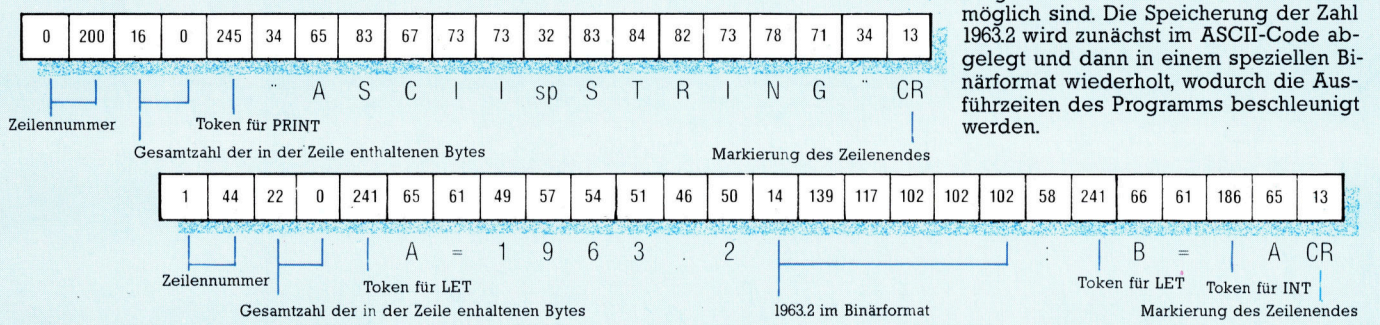
Die Verbindungsadresse (Link address) gibt die Adresse an, an der die nächste Programmzeile anfängt. Die Verbindungsadressen bestehen ähnlich wie die Seitenadressen aus einem Seitenbyte mit einem Offsetbyte.



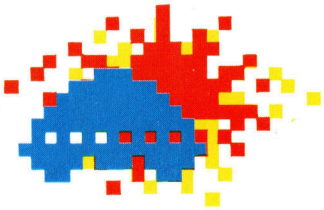
Spectrum

```
200 PRINT "ASCII STRING"
300 LET A=1963.2:LET B=INTA
```

Die Zeilenlänge wird hierbei in zwei Bytes statt nur in einem gespeichert, so daß Programmzeilen mit mehr als 255 Zeichen möglich sind. Die Speicherung der Zahl 1963.2 wird zunächst im ASCII-Code abgelegt und dann in einem speziellen Binärformat wiederholt, wodurch die Ausführzeiten des Programms beschleunigt werden.



Invasion aus dem Weltraum



Mit dem Spiel „Space Invaders" ist der Firma Atari ein Volltreffer gelungen: Millionen begeisterter Bildschirm-Astronauten haben bereits in den Spielhallen Bekanntschaft damit gemacht.

„Space Invaders" läßt sich guten Gewissens als „das Computerspiel an sich" bezeichnen. Der Name dieses Spiels ist so populär, daß er häufig als Überbegriff für alle anderen Arcadespiele gebraucht wird.

Begonnen hat der Siegeszug des Spiels 1978. Eine wahre Epidemie brach vor allem unter den Jugendlichen aus, gefolgt von den Unkenrufen einer besorgten Elternschaft, die schon eine ganze Generation in Spielhallen ihr Vermögen verspielen sah.

Was statt dessen passierte, war ein Wandel der Betrachtungsweise: Die Angst vor dem „Schreckgespenst Computer" wurde durch den spielerischen Umgang mit der Maschine gemindert. Aus dem unbekannten elektronischen Gegenüber wurde nach und nach der vertraute Spielkamerad.

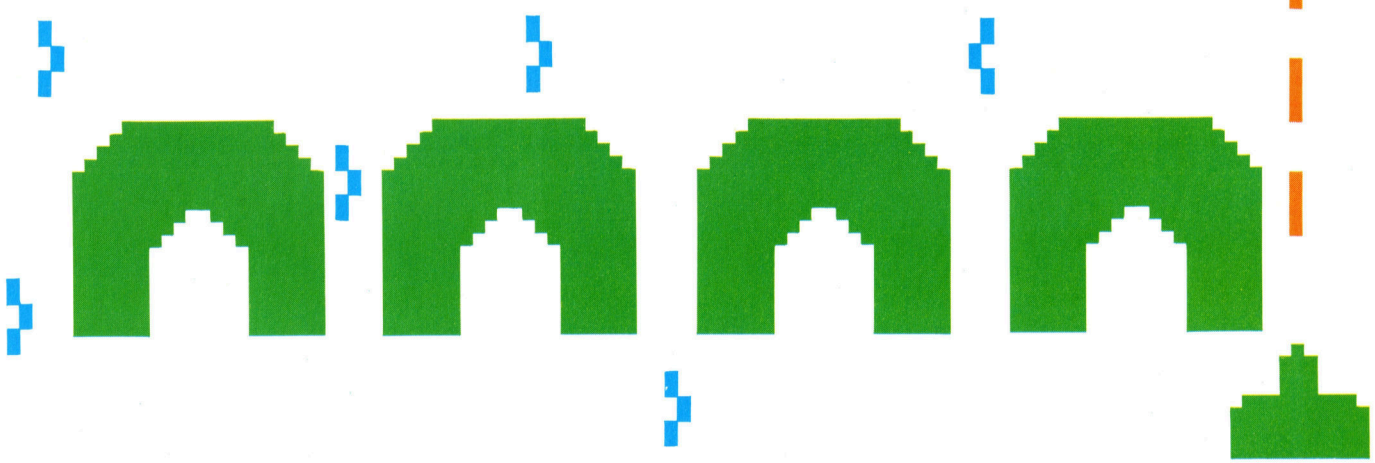
„Space Invaders" wurde Wegbereiter für eine Vielzahl von Computerspielen, in denen die Welt meist recht simpel aussieht: Mit der „FIRE"-Taste – und „Three Lifes Left" – kämpft ein heldenhafter Spieler oder eine ebensolche Spielerin gegen eine Übermacht bösartiger Kreaturen, die unablässig angreifen.

Nach heutigem Standard ist „Space Invaders" zwar bereits ein wenig zu einfach und damit veraltet, es bleibt aber trotzdem das populärste Spiel seiner Art. Der Spieler steuert eine bewegliche Laser-Basis, von der aus auf angreifende Feinde geschossen wird. So wird verhindert, daß diese die Erde erreichen. Dringen die Angreifer bis zum unteren Rand des Bildschirms vor bzw. treffen sie mit ihren Geschossen die Laser-Basis, verliert der Verteidiger eines seiner drei Leben.

Zwischen dem originären Arcadespiel und der Heimcomputer-Version bestehen üblicherweise einige Unterschiede: Die Angreifer entstehen nicht mehr einfach im luftleeren Raum, sondern kommen aus einer großen Rakete auf der linken Bildschirmseite. Die Invaders wurden farbenfroher gestaltet und die einzelnen Sprites sehr präzise ausgelegt. Dagegen fehlen in der Heimversion die Barrieren, die zum Schutz der Laser-Basis dienten. Auch die Strecke, die die Angreifer bis zur Erdoberfläche zurücklegen müssen, wurde verkürzt. Nur am nervtötenden „Herzschlag"-Geräusch änderte sich nichts. Vielleicht ist gerade der hieraus resultierende Adrenalinstoß das Erfolgsrezept von „Space Invaders"? Selbstverständlich gibt es auch noch den Mystery-Bonus – für das Abschießen eines der sporadisch auftauchenden UFOs.

Trotz harter Konkurrenz durch teilweise weiterentwickelte Computerspiele hat „Space Invaders" auch in seinem siebten Jahr nichts vom ursprünglichen Reiz eingebüßt. Es ist schon jetzt ein echter „Software-Klassiker".

Space Invaders: Für alle Atari-Computer
Hersteller: Atari
Autor: Atari
Joysticks: Notwendig
Format: Cartridge



Bits in der Schlinge

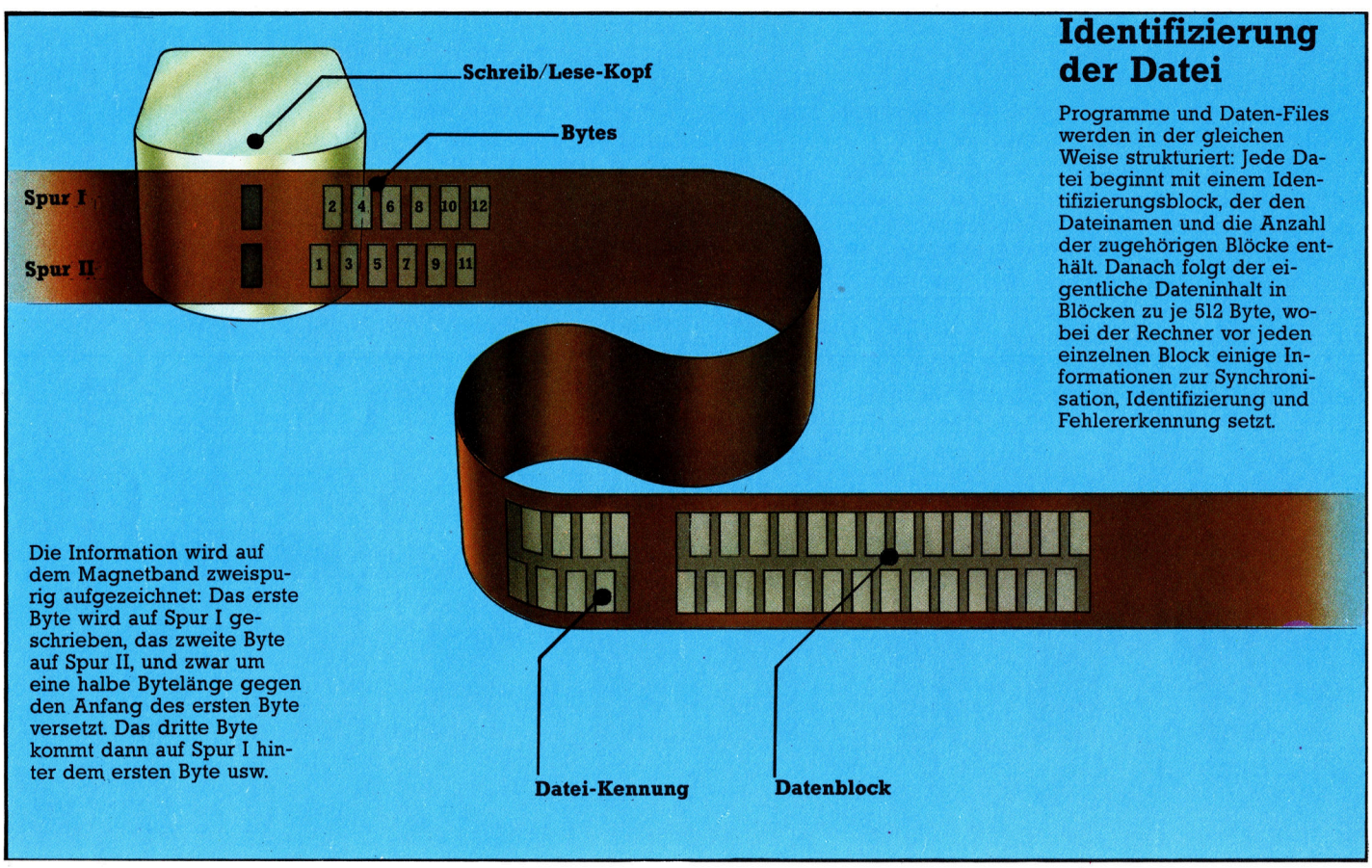
Der Sinclair Spectrum ist als preisgünstiger, leistungsfähiger Heimcomputer bekannt. Dazu fehlte aber ein billiger, schneller Massenspeicher, bis Sinclair das „Microdrive" und die zugehörige Schnittstelle auf den Markt brachte.

Das Microdrive-Laufwerk arbeitet mit einem ca. 5 m langen und 2 mm breiten Magnetband, das in einer kleinen Cartridge als Endlosschleife alle sieben Sekunden einmal umläuft. Der Zugriff ist ähnlich wie bei einer Diskette organisiert, daher auch die Bezeichnung „floppy tape" oder „stringy floppy" („Banddiskette"). Die Speicherung erfolgt digital, also nicht akustisch wie bei Cassettenrecordern. Das Magnetband wird zweispurig beschrieben, wobei die Bits sequentiell in einem Zickzackmuster abgelegt werden. Dadurch wird mit Hilfe eines zweifachen Schreiblesekopfs die doppelte Aufzeichnungsdichte und Geschwindigkeit erreicht wie bei einem Einspursystem. Jeder Datenblock umfaßt 512 Bytes und ist zur Identifikation mit einer Startsequenz von 27 zusätzlichen Bytes versehen.

Solch ein Block wird als Sektor bezeichnet. Auf das 5m-Band passen ca. 200 Sektoren, d. h., die Aufzeichnungsdichte beträgt rund 220 Byte/cm. Dateien von weniger als 512 Byte Länge belegen immer einen ganzen Sektor, bei Dateien, die mehrere Sektoren beanspruchen, bleibt der Rest des letzten Sektors frei. Die theoretische Speicherkapazität von 100 KByte ist daher in der Praxis nur zu 85–90 KByte nutzbar. Die durchschnittliche Zugriffszeit (Finden und Laden) auf ein Programm beträgt etwa 10–15 Sekunden.

Die ZX-Schnittstelle

Zum Anschluß an den Rechner benötigen Sie das ZX-Interface 1, das mit der Peripherie-Steckleiste des Spectrum verbunden wird. Bis zu acht Microdrives können miteinander verbunden werden. Das Interface 1 dient außerdem als RS 232-Schnittstelle, zum Aufbau von lokalen Rechnernetzen und als Verbindung für den ZX-Drucker. Das Interface-ROM enthält zudem eine Erweiterung des Sinclair-BASIC



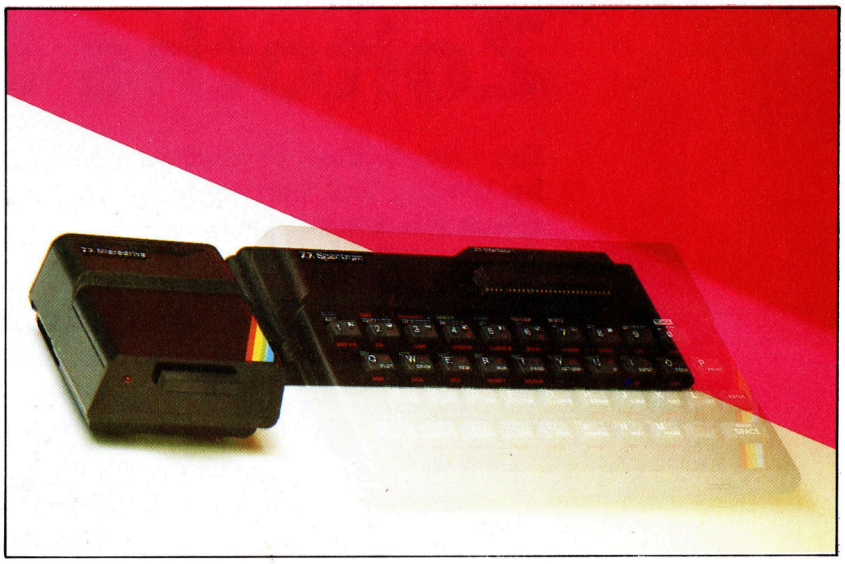
Identifizierung der Datei

Programme und Daten-Files werden in der gleichen Weise strukturiert: Jede Datei beginnt mit einem Identifizierungsblock, der den Dateinamen und die Anzahl der zugehörigen Blöcke enthält. Danach folgt der eigentliche Dateninhalt in Blöcken zu je 512 Byte, wobei der Rechner vor jeden einzelnen Block einige Informationen zur Synchronisation, Identifizierung und Fehlererkennung setzt.

Die Information wird auf dem Magnetband zweispurig aufgezeichnet: Das erste Byte wird auf Spur I geschrieben, das zweite Byte auf Spur II, und zwar um eine halbe Bytelänge gegen den Anfang des ersten Byte versetzt. Das dritte Byte kommt dann auf Spur I hinter dem ersten Byte usw.

zur Steuerung des Datenflusses über die verfügbaren Schnittstellen.

Das Microdrive und das Interface 1 erhöhen die Leistungsfähigkeit des Spectrum-Systems. Die Einsatzmöglichkeiten eines Floppy-Laufwerks werden weitgehend nachgeahmt. Die fragliche Langzeit-Zuverlässigkeit der Cassetten und die Wartezeiten infolge der sequentiellen Datei-Speicherung beeinträchtigen allerdings den insgesamt positiven Eindruck. Außerdem gibt es bisher noch wenig Software auf Microdrive-Cassetten, wenn sich das auch wahrscheinlich bald ändern wird. – Der neue Sinclair QL mit zwei eingebauten Microdrives dürfte die Entwicklung sehr beschleunigen. Die erforderliche Zuverlässigkeit vorausgesetzt, können die Microdrive-Laufwerke für den äußerst wertvoll sein, der eine preisgünstige und schnelle Speichermöglichkeit braucht.



Das Microdrive bietet eine preisgünstige, leistungsfähige Alternative zum Cassettenrecorder. Es wird über das Interface 1 an den Spectrum angeschlossen, das gleichzeitig eine RS232- und eine Cassettenrecorder-Schnittstelle enthält.

Microdrive – Befehle

Die wichtigen Tastenbefehle sind:

FORMAT CAT SAVE* VERIFY* LOAD*
MERGE* ERASE OPEN# PRINT# INPUT#
INKEY$# CLOSE# MOVE

Weiter steht allgemein „M“ für Microdrive, der Parameter N bezeichnet das angesprochene Laufwerk (1–8) und S den zugeordneten Kanal (4–15).

Format

Besorgt die Formatierung, versieht die Cassette mit einem Namen und löscht alle alten Informationen.

FORMAT "M"; N; "NAME" oder
FORMAT M$; N; C$

Dabei steht NAME als Cassettentitel (1–10 Buchst.). M$ (oder m$) ist dasselbe wie "M", und C$ (1–10 Buchst.) entspricht dem Namen.

Cat

Liefert über den Bildschirm eine Auflistung der Dateien auf der Cassette im Laufwerk N. Eingabeform:

CAT N oder CAT#S;N

Die Liste enthält den Cassettennamen, bis zu 50 Dateinamen und den freien Speicherplatz in KByte.

Save*

Erzeugt Dateien, die Programme, mit Namen versehene Zeichenketten oder Daten enthalten können; mögliche Formen:

1. SAVE* "M"; N; "DATEINAME"
2. SAVE* "M"; N; "DATEINAME" SCREEN$
3. SAVE* "M"; N; "DATEINAME" DATA A( )
4. SAVE* "M"; N; "DATEINAME" LINE X

Dadurch wird jeweils folgendes erzeugt:

1. eine beliebige Datei,
2. eine Bildschirm-Datei,
3. eine Datei mit dem Datenfeld A( ),
4. eine Datei, die nach dem Laden mit Run ab Zeile X aufrufbar ist.

Verify*, Merge* und Erase*

Werden wie bei SAVE unter Ziffer 1 aufgebaut; VERIFY* vergleicht die mit "DATEINAME" bezeichnete Datei mit dem aktuellen Inhalt des Arbeitsspeichers und löst bei Abweichungen eine Fehlermeldung aus. MERGE* überschreibt "DATEINAME" mit dem Inhalt des Arbeitsspeichers, und ERASE* löscht die Datei "DATEINAME".

Load*

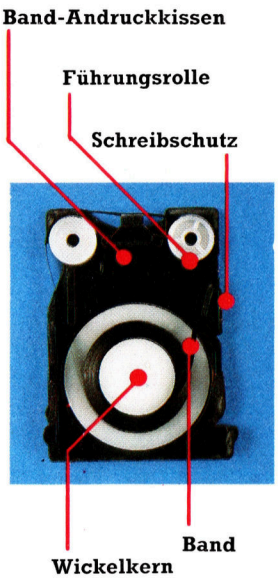
Kann wie SAVE* (1 und 2) aufgebaut werden. Bei Ausführung von LOAD wird der Inhalt der angesprochenen Datei ins RAM geladen.

Open#, Print#, Input#, Inkey#, Close# und Move#

Diese Befehle betreffen die Handhabung der Dateien. Sie ermöglichen trotz der sequentiellen Speicherung eine Nachbildung des Direktzugriffs, indem aus den Dateien nach dem Einlesen die benötigten Einzeldaten herausgezogen werden. Ebenso kann eine Datei im Arbeitsspeicher geändert und dann wieder gespeichert werden. Dazu werden durch OPEN und CLOSE Datenkanäle organisiert, die über vereinbarte Kanalnummern ansprechbar sind. Z. B. wird durch

OPEN#S; "M"; N; "DATEINAME"

der Datei "DATEINAME" auf einer Cassette im Laufwerk N der Kanal S zugeordnet. Einträge in diese Datei sind dann mit PRINT # S und Lesevorgänge mit INPUT # S oder INKEY$# S und Lesevorgänge mit INPUT# S möglich. Mit MOVE können Daten auf dem Band verschoben werden. Nicht mehr benötigte Kanäle sollten durch CLOSE# S geschlossen werden.



Die Cassette enthält eine 5 m lange Endlosschleife aus 2 mm breitem spezialbeschichtetem Band. Beim Einschieben ins Laufwerk preßt das Andruckkissen das Band gegen den Magnetkopf während ein Microschalter den Schreibschutz abtastet. Ist dieser kleine Zapfen herausgebrochen, kann das Band nur noch gelesen werden, so daß die Information gegen Überschreiben gesichert ist.

Zoltoths Schrein

Die wichtigsten Prozeduren für das Abenteuer-Spiel wurden bereits im letzten Teil fertiggestellt. Jetzt geht es darum, Bewegung und vor allem Abenteuer in diese Phantasiewelt hineinzuprogrammieren.

Zuerst müssen Möglichkeiten geschaffen werden, sich von einem Raum zum nächsten – in den Richtungen Norden, Süden, Osten und Westen – bewegen zu können.

```
TO N
  GEHEN "N :AUSGANG.LIST
END

TO S
  GEHEN "S :AUSGANG.LIST
END

TO O
  GEHEN "O :AUSGANG.LIST
END

TO W
  GEHEN "W :AUSGANG.LIST
END
```

Die GEHEN-Prozedur prüft zuerst, ob Sie die gewünschte Richtung einschlagen können, und übergibt den Bewegungswert an GEHEN1.

```
TO GEHEN :RICHT :LIST
  IF EMPTY? :LIST THEN PRINT [DIESEN
    WEG KANNST DU NICHT NEHMEN] STOP
  MAKE "AUSGANG FIRST :LIST
  IF :RICHT = FIRST :AUSGANG THEN
    GEHEN1 LAST :AUSGANG STOP
  GEHEN :RICHT BUTFIRST :LIST
END

TO GEHEN1 :NR
  MAKE "RAUM.NAME HIER.DETAILS
  MAKE "HIER :NR
  ZUWEISEN.VARIABLEN
  SEHEN
END
```

Die Prozedur GEHEN1 erwartet eine Raumnummer als Eingabe. Danach werden Elemente aus den einzelnen Listen abgerufen und dem Raumnamen zugeordnet (die Details können sich inzwischen auch geändert haben). Und schließlich wird HIER mit der neuen Raumnummer definiert.

```
TO HIER.DETAILS
  OUTPUT (LIST :RAUMBESCHREIBUNG
    :OBJEKTE :AUSGANG.LIST)
END
```

Der Befehl LIST stellt eine Liste seiner Eingaben dar. Der Unterschied zwischen LIST und SENTENCE läßt am besten anhand eines Beispiels demonstrieren:

```
LIST [A] [B] [C] ergibt [[A] [B] [C]]
SENTENCE [A] [B] [C] ergibt [A B C]
```

LIST eignet sich bei diesem Programm besser als SENTENCE, da die individuellen Elemente in Unterlisten gefaßt werden sollen.

In jedem „richtigen" Adventure gibt es zahlreiche Gefahren, wie giftige Schlangen oder Treibsand, die der Spieler bewältigen muß. Sobald er mit einem derartigen gefährlichen Element zusammentrifft, muß eine Aktionsfolge aufgerufen und gleichzeitig verhindert werden, daß der Spieler den Raum verläßt, bevor er die gefährliche Situation gemeistert hat. Dafür wird eine weitere Liste erstellt, die spezielle Gefahrenprozeduren beinhaltet. Man kann RAUM.2 zum Beispiel so definieren: [[[DU BIST IN EINER DUNKLEN HOEHLE] [VOR DIR STRAHLT EIN LICHT]] [KISTE] [[N 5] [O 6]] [SCHLANGE]], wobei SCHLANGE die Gefahr ist. Um diese Möglichkeiten zu integrieren, muß die bereits gezeigte SEHEN-Prozedur geändert werden.

```
TO SEHEN
  PRINTL :RAUMBESCHREIBUNG
  PRINT "
```

```
    PRINT [DU SIEHST:]
    IF EMPTY? :OBJEKTE THEN PRINT [NICHTS
       BESONDERES] ELSE PRINT :OBJEKTE
    PRINT "
    PRINT [DU KANNST GEHEN NACH:]
       PRINT.AUSGAENGE :AUSGANG.LIST
    PRINT "
    IF GEFAHR? THEN RUN :GEFAHREN
END
```

Der Befehl RUN erwartet eine Liste als Eingabe und arbeitet die in der Liste definierten Prozeduren ab.

```
TO GEFAHR?
   IF EMPTY? :GEFAHREN THEN OUTPUT
      "FALSE
   OUTPUT "TRUE
END
```

Die folgenden Prozeduren müssen umgeschrieben werden, um die „Gefahren" einzubauen:

```
TO ZUWEISEN.VARIABLEN
   MAKE "RAUM.NAME WORD "RAUM. :HIER
   MAKE "RAUM THING :RAUM.NAME
   MAKE "RAUMBESCHREIBUNG
      RAUMBESCHREIBUNG :RAUM
   MAKE "OBJEKTE OBJEKTE :RAUM
   MAKE "AUSGANG.LIST AUSGANG.LIST
      :RAUM
   MAKE "GEFAHREN GEFAHREN :RAUM
END

TO GEFAHREN :RAUM
   OUTPUT ITEM 4 :RAUM
END

TO HIER.DETAILS
   OUTPUT (LIST :RAUMBESCHREIBUNG
      :OBJEKTE :AUSGANG.LIST :GEFAHREN)
END

TO GEHEN :RICHT :LIST
   IF GEFAHR? THEN PRINT [DIESEN WEG
      KANNST DU NICHT NEHMEN] STOP
   IF EMPTY? :LIST THEN PRINT [DIESEN
      WEG KANNST DU NICHT NEHMEN]
      STOP
   MAKE "AUSGANG FIRST :LIST
   IF :RICHT = FIRST :AUSGANG THEN
      GEHEN1 LAST :AUSGANG STOP
   GEHEN :RICHT BUTFIRST :LIST
END
```

GEHEN verhindert jegliche Bewegung, bis die GEFAHREN-Liste leer ist – []. Die definierten Gefahren lassen sich nun auf jeden gewünschten Raum übertragen, indem nur die Raumbeschreibung geändert wird.

Mit den hier entwickelten Prozeduren können wir ein komplettes Abenteuerspiel zusammenstellen. Dieses trägt den Titel „Der Schrein von Zoltoth". In diesem Adventure ist der Spieler auf der Suche nach dem Zepter von Gilgesh, das die Hohepriester von Zoltoth gestohlen und in ihren Tempel gebracht haben. Bei Beginn des Abenteuers befindet man sich am Eingang zu der unterirdischen Höhle, die zum Schrein von Zoltoth führt. Bevor Sie Ihr Programm entwerfen, sollten Sie ein Drehbuch für eine erfolgreiche Reise durch das Spiel erstellen und das Programm auf dieser Basis strukturieren. Das Drehbuch zu unserem Spiel wird hier nicht veröffentlicht, damit die Spannung nicht verlorengeht.

Befehle werden definiert

Im nächsten Stadium ist das Programm in RAEUME zu gliedern. Das bedeutet, Räumlichkeiten, ihre Inhalte und ihre Lage zueinander sind zu planen. Dieser Entwurf der Phantasiewelt dient dann als Grundlage, um die Räume im Programm zu definieren und ihre möglichen Ausgänge hinzuzufügen. Die Abenteurer ihrerseits sollten während des Spielverlaufs ebenfalls eine Karte fertigen.

Schließlich muß der im Spiel verwendete Wortschatz bestimmt werden. Welche Eingabeworte des Abenteurers soll das Programm verstehen? Wir legen fest:
1. Sieben Einzelwort-Befehle: START, SEHEN, N, S, O, W und INVENTAR.
2. Zusammengesetzte Befehle bestehen aus einem Verb, auf das ein Substantiv folgt.
Die Verben sind: NEHMEN, ABLEGEN, PRUEFEN, TOETEN, REIBEN und OEFFNEN.
Die Substantive sind: SCHWERT, TRUHE, ZEPTER, RING und SCHLANGE.
Alle Befehle werden direkt eingegeben. Sind die Eingaben im Programm definiert, werden sie ausgeführt, andernfalls erfolgt eine LOGO-Fehlermeldung.

Es wäre allerdings sinnvoll, eine Fehlermeldung wie „ICH KENNE DIESES WORT NICHT" zu integrieren, statt sich mit der üblichen LOGO-Fehlermeldung zu begnügen. Dazu ist eine Schleife erforderlich, die die Eingaben überprüft und sie, wenn sie richtig sind, ausführt.

Hier ist eine Möglichkeit, das bereits definierte Vokabular zu testen (beachten Sie dabei die bereits aufgezeigten Unterschiede bei den LOGO-Versionen):

```
TO START
  MAKE "HIER 1
  MAKE "INVENTAR []
  SET.RAEUME
  ZUWEISEN.VARIABLEN
  SEHEN
  SPIEL
END
TO SPIEL
  PRINT1 "BEFEHL:
  MAKE "INPUT REQUEST
  IF KORREKT? :INPUT RUN :INPUT ELSE
    PRINT [ICH VERSTEHE NICHT]
  SPIEL
END
TO KORREKT? :BEF
  IF ((COUNT :BEF) = 1) THEN OUTPUT
    KORR1? :BEF
  IF ((COUNT :BEF) = 2) THEN OUTPUT
    KORR2? :BEF)
  OUTPUT "FALSE
END
TO KORR1? :BEF
  IF MEMBER? FIRST :BEF [INV W O S N
    SEHEN START] OUTPUT "TRUE
  OUTPUT "FALSE
END
TO KORR2? :BEF
  IF ALLOF KORRV? FIRST :BEF KORRS?
    LAST :BEF
  OUTPUT "TRUE
  OUTPUT "FALSE
END
TO KORRS? :SUBSTAN
  IF MEMBER? :SUBSTAN [SCHWERT TRUHE
    ZEPTER RING SCHLANGE] OUTPUT
    "TRUE
  OUTPUT "FALSE
END
TO KORRV? :VERB
  IF MEMBER? :VERB [NEHMEN ABLEGEN
    PRUEFEN TOETEN REIBEN OEFFNEN]
    OUTPUT "TRUE
  OUTPUT "FALSE
END
```

Zunächst sind alle Prozeduren aus der letzten Folge einzugeben. Um das Spiel zu beginnen oder es erneut spielen zu können, wird START eingegeben.

```
TO START
  MAKE "HIER 1
  MAKE "INVENTAR []
  SET.RAEUME
  ZUWEISEN.VARIABLEN
  SEHEN
END
```

SET.RAEUME generiert die Räume entsprechend der Karte.

```
TO SET:RAEUME
  MAKE "RAUM.1 [[[DU STEHST AM EIN-
    GANG] [EINER HOEHLE]] [] [[O 2]] []]
  MAKE "RAUM.2 [[[DU BIST IN EINER
    DUNKLEN FEUCHTEN HOEHLE]] [] [[S 3]
    [O 4] [W 1]] [] ]
  MAKE "RAUM.3 [[[DU BIST IN EINER
    DUNKLEN FEUCHTEN HOEHLE]] [] [[N 2]
    [O 5]] []]
  MAKE "RAUM.4 [[[DU BIST IN EINER
    GROSSEN UNTERIRDISCHEN KAMMER]]
    [] [[N 6] [S 5] [W 2]] [SCHLANGE.GREIF-
    TAN]]
  MAKE "RAUM.5 [[[DU BIST IN EINER
    DUNKLEN FEUCHTEN HOEHLE]]
    [SCHWERT] [[N 4] [W 3]] []]
  MAKE "RAUM.6 [[[DU BIST IN EINEM
    HEILIGEN SCHREIN—RAUM] [IN EINER
    NISCHE IN DER NORDWAND] [STEHT
    EIN ALTAR]] [] [[N 7] [S 4] [O 8]] [TOR]]
  MAKE "RAUM.7 [[[DU STEHST VOR] [DEM
    ALTAR DES ZOLTOTH] [UEBER DEM
    ALTAR STEHT GESCHRIEBEN:] ["KEIN
    UNEDLES METALL DARF MIR NAHE
    KOMMEN"]] [RING] [[S 6]] []]
  MAKE "RAUM.8 [[[DU BIST IN EINER
    DUNKLEN FEUCHTEN HOEHLE]] []
    [[S 10] [O 9] [W 6]] [SCHLANGE.GREIF-
    TAN]]
  MAKE "RAUM.9 [[[DU BIST IN EINER
    DUNKLEN FEUCHTEN HOEHLE]] [TRUHE]
    [[S 11] [W 8]] []]
  MAKE "RAUM.10 [[[DU BIST IN EINER
    DUNKLEN FEUCHTEN HOEHLE]] [] [[N 8]
    [O 11]] []]
  MAKE "RAUM.11 [[[DU BIST IN DER
    SAKRISTEI] [DER PRIESTER VON
    ZOLTOTH]] [ZEPTER] [[N 9] [W 10]] []]
END
```

Fachwörter von A bis Z

Bit = Bit
Offiziell ist dieses Wort eine Abkürzung des Begriffes Binary DigIT (Binärstelle) als Name für die kleinste Informationseinheit. Das „Bit" taucht im Computer-Jargon in vielen Zusammensetzungen auf. Hier eine kleine Auswahl:

Bit-Kopierer: Ein Hilfsprogramm, das einen Datenträger Bit für Bit auf einen anderen kopiert. Mit diesem Verfahren werden die meisten Software-Raubkopien erstellt, da so manche Schutzvorkehrung umgangen werden kann.

Bit-Fehler: Das „Umkippen" eines oder mehrerer Bits auf einen anderen logischen Status. Dabei kann es sich um nichtreproduzierbare Fehler handeln, wie beispielsweise infolge von Schaltspitzen im Stromnetz oder infolge kosmischer Strahlung während erhöhter Sonnenfleckenaktivität. Diese treten nach Neustart des Rechners nicht wieder auf. Oder es handelt sich um permanente Fehler, deren Ursache z. B. ein defekter Transistor in einem RAM-Baustein sein kann.

Bit-parallel: Ein Verfahren, bei dem alle Bits einer Informationseinheit auf entsprechend vielen parallelen Leitungen gleichzeitig übermittelt werden. Dies ist aufwendiger, aber schneller, als alle Bits nacheinander „bitseriell" auf einer einzigen Leitung zu übertragen.

Bit-Verarbeitung (auch Bit-Manipulation): Bezeichnung für die unterste Stufe der Informationsverarbeitung (nämlich bitweise).

Block = Block
Ein „Block" ist eine definierte Datenmenge, und zwar die kleinste Einheit, die bei einem Schreib- bzw. Lesevorgang gleichzeitig verarbeitet wird. Wenn bei einem Diskettenlaufwerk jeweils vier komplette Sektoren zu je 256 Bytes beschrieben werden, dann beträgt die Blocklänge 1 KByte. Jeder Block ist eine abgeschlossene Einheit und enthält oft noch Kontrollinformationen, die eine Fehlererkennung beim Einlesen ermöglichen. Daneben sind Anfang und Ende des Blocks mit besonderen Kennsymbolen ausgezeichnet.

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.

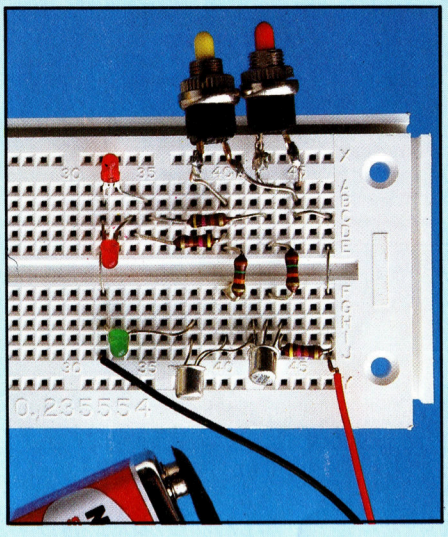
Boot = Booten
Ein Programm auf Diskette kann sich nicht selbständig in den Speicher laden. Dafür muß das Betriebssystem sorgen. Dieses befindet sich bei Heimcomputern üblicherweise im ROM und wird daher beim Einschalten des Rechners sofort aktiviert. Bei größeren Rechnern ist das Betriebssystem ein separates Programm im RAM.

Für das Einlesen ist ein im ROM abgelegtes Programm, der „bootstrap loader" oder Urlader erforderlich. Der Urlader lädt nach dem Einschalten das Betriebssystem Byte für Byte in einen bestimmten RAM-Bereich und startet es. Dieser Vorgang wird im Computer-Jargon als „Booten" bezeichnet.

Branch =
Sprung oder Verzweigung
Die Ausführung einer Sprunganweisung bewirkt, daß der Rechner das Programm nicht mit dem nächstfolgenden Befehl fortsetzt, sondern zu einem anderen Programmteil springt. Zu unterscheiden sind der „unbedingte" Sprung, der in jedem Fall erfolgt, und der „bedingte" Sprung, der an die Erfüllung bestimmter Bedingungen gebunden ist. In BASIC bewirkt der einfache GOTO-Befehl einen unbedingten Sprung auf die in der Anweisung genannte Zeilennummer. Dagegen würde die bedingte Sprunganweisung

IF SUM > 4 THEN GOTO 100

nur dann einen Sprung auf Zeile 100 auslösen, wenn SUM größer ist als 4.



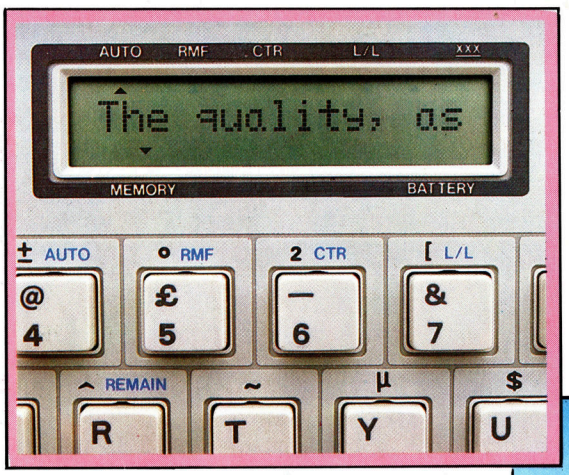
Breadboard = Experimentierplatte
Als Breadboard werden Hilfsmittel bezeichnet, auf denen Elektronik-Bauteile für Versuchsaufbauten verschaltet werden können. Meist handelt es sich um gelochte Platten, die einseitig mit Lötaugen oder parallelen Kupferbahnen ausgestattet sind. IC-Sockel, Transistoren, Widerstände usw. werden mit ihren Pins durch die Karte gesteckt und verlötet. Die Leiterbahnen werden, wo nötig, mit einem Schaber aufgetrennt oder mit Draht überbrückt, um die gewünschte Schaltung zu realisieren.

Alternativ gibt es auch lötfreie Experimentier-Steckbretter, die mehr für die Ausbildung als für den professionellen Gebrauch gedacht sind. Sie enthalten in einem zentimeterstarken Plastikgehäuse eine Vielzahl von Löchern mit intern verbundenen Federbuchsen. Die Bauelemente werden nur eingesteckt und können zur Veränderung der Schaltung leicht wieder entfernt werden.

Bildnachweise
589, 592: Steve Cross
590: Kevin Jones
593, 595: Chris Stevens
596: Sainsbury
597: Tony Duncan-Smith, Liz Dixon
604, 613: Ian McKinnell
605: Marcus Wilson-Smith
606: BBC Television
611: Liz Dixon
612: Alun Jones
614, 615: David Higham

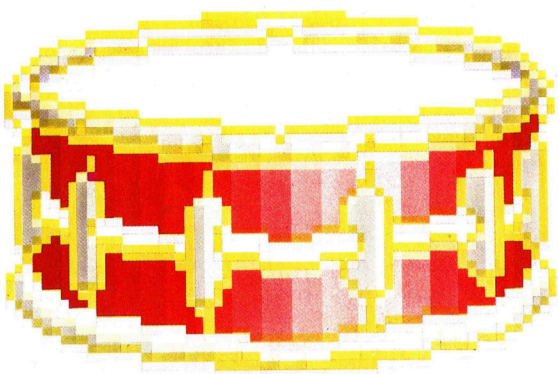
+++ Vorschau +++ Vorschau +++ Vorschau +++

computer kurs Heft 23



Vielseitig
ist die EP-44 von Brother. Das Gerät hat die Größe einer Reiseschreibmaschine, kann aber weit mehr.

Commodore 16
heißt das Nachfolge-Modell des bekannten VC 20. Verbessert wurde das BASIC, zudem gibt es integrierte Hilfsprogramme.



Computerklänge
halten mehr und mehr Einzug in die heutige Musik. Unsere Serie über Codierung und Erzeugung synthetischer Sounds erklärt die Grundlagen.

+++ Bewegungsabläufe bei Robotern +++

BASIC-Übungen +++ Maschinencode +++

Tips für die Praxis +++ Abenteuerspiel

+++ Software: Lagerhaltung +++ LOGO-

Kurs +++ Firmenporträt: Acorn +++