Einsteigen - Verstehen - Beherrschen

DM 3,80 öS 30 sfr 3,80

computer kurs

Der Sanyo MBC-550

Umwelt erkennen

Gutes Bild: Monitore

Football-Manager

Heft 51

Ein wöchentliches Sammelwerk

Programmierkurs LISP

computer kurs

Heft 51

Inhalt

WIE SIE JEDE WOCHE IHR HEFT BEKOMMEN

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

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

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

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

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

Abonnement

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

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

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

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

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

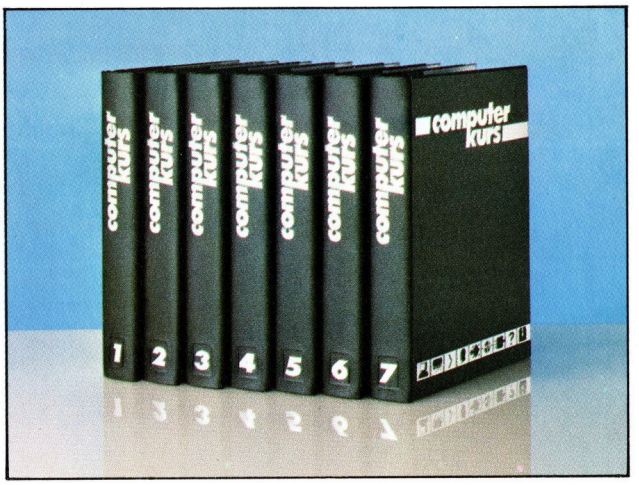
Schweiz: Der Sammelordner kostet sfr 15. Bitte wenden Sie sich an Ihren Kiosk; dort werden Sie jederzeit die gewünschten Exemplare erhalten.

INHALTSVERZEICHNIS

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

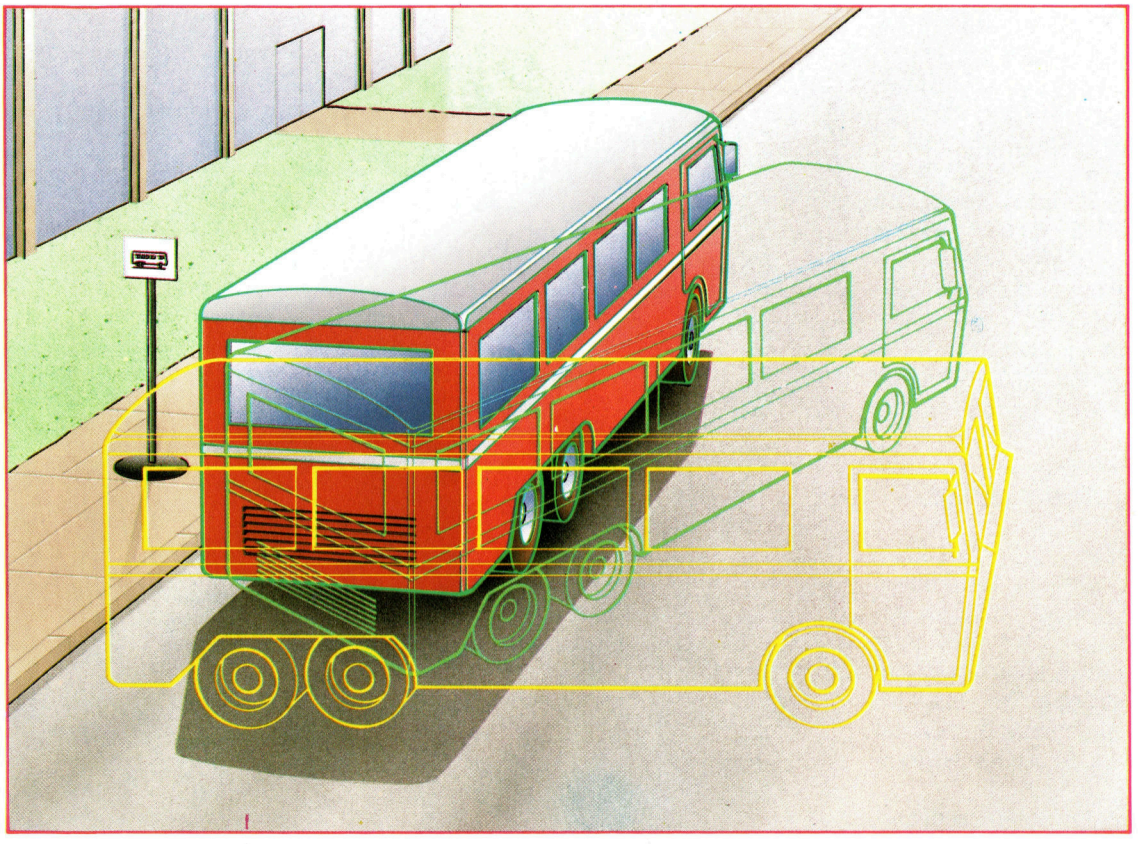
Redaktion: Winfried Schmidt (verantw. f. d. Inhalt), Elke Leibinger, Susanne Brandt, Uta Brandl (Layout), Sammelwerk Redaktions-Service GmbH, Paulstraße 3, 2000 Hamburg 1

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



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

Höhere Einsichten



Bei einigen Mustererkennungs-Systemen bedient man sich des „Top-Down"-Vergleichs, in dem ein Muster oder eine Szene auf einen bestimmten Gegenstand hin abgetastet wird. Eine Rahmendarstellung des Gegenstandes wird in verschiedenen Winkeln auf den Bildschirm projiziert, bis eine Projektion gefunden wird, die auf den Umriß des gesuchten, tatsächlichen Gegenstandes paßt.

Da Sehen und Verstehen zwei eng zusammenhängende Faktoren sind, die zum Begreifen unserer Umwelt beitragen, ist es oft schwer, die beiden eindeutig voneinander zu trennen.

Schönheit entsteht im Auge des Betrachters", heißt ein Sprichwort. Genauer aber muß es heißen, daß Schönheit im Gehirn des Wahrnehmenden entsteht oder – noch genauer – in einer komplexen Kette nervlicher Prozesse, die irgendwo auf der Netzhaut beginnt und in der Großhirnrinde am Hinterkopf endet.

Die visuelle Wahrnehmung ist so untrennbar mit dem Verstehen unserer Umwelt verbunden, daß wir häufig sagen „ich seh' schon" oder „ich sehe ein, daß", wenn wir meinen „ich verstehe". Irgendwie muß auch der Computer die über eine Kamera oder ein anderes lichtempfindliches Gerät erhaltenen Informationen umsetzen, um zu erfahren, was damit über den Zustand seiner Umgebung ausgesagt werden kann. Die Beschaffung der Information ist leicht – das Problem ist ihre Interpretation.

Der Prozeß der Umwandlung von Informationen in Aussagen ist vom Zeitablauf her in drei Schritte zu untergliedern:

1. Bild-Entwicklung: Die Umwandlung eines verwischten oder unscharfen Bildes in ein scharfes (relativ leicht lösbar).
2. Mustererkennung: An- oder Abwesenheit eines bestimmten Gegenstands feststellen (schwieriger).
3. Bildverständnis: Erarbeiten und verstehen dessen, was in der realen Welt geschieht (sehr schwierig).

Wenngleich der dritte Schritt – echtes Verstehen des Computers – noch nicht erlangt wurde, hat man auf den beiden vorhergehenden Ebenen bereits gute Ergebnisse erreicht.

Mustererkennung ist eine herausfordernde Aufgabe. Bei dieser Methode werden Bilder klassifiziert, indem diese mit einer begrenzten Reihe von Alternativen verglichen werden, so etwa bei optischen Zeichenerkennungs-Systemen mit den Buchstaben des Alphabets. Typisch ist dabei, daß gewisse Merkmale des digitalisierten Bildes isoliert untersucht werden. Das System kann, entsprechend dem Vorhandensein oder Nichtvorhandensein dieser Merkmale, die Muster erkennen und klassifizieren.

Die Erkennung mancher Merkmale, wie beispielsweise diagonaler Striche, scheint uns

klar und einfach. Oft aber erfolgt die Erkennung durch optische Systeme aus rein willkürlichen Berechnungen, die aufgrund der Bilddaten durchgeführt werden und somit unzureichende Ergebnisse hervorbringen.

Historisch gesehen gibt es zwei voneinander abweichende Wege zur „Computervision" im Sinne von Bilderkennung – die „Top-Down"- oder Muster-Methode und die „Bottom-Up"- oder auf Daten basierende Methode.

Die auf der Muster-Methode basierende Szenenanalyse wird zuweilen als „kontrollierte Halluzination" bezeichnet. Das System verfügt über ein internes Muster dessen, was es betrachtet (zum Beispiel einen Doppeldecker-Bus), und es projiziert das Modell in verschiedenen Richtungen auf die Bildfläche. Dann sucht es nach Übereinstimmungen zwischen dem „eingebildeten" Muster und dem, was tatsächlich zu sehen ist.

Verstehen durch Abstraktion

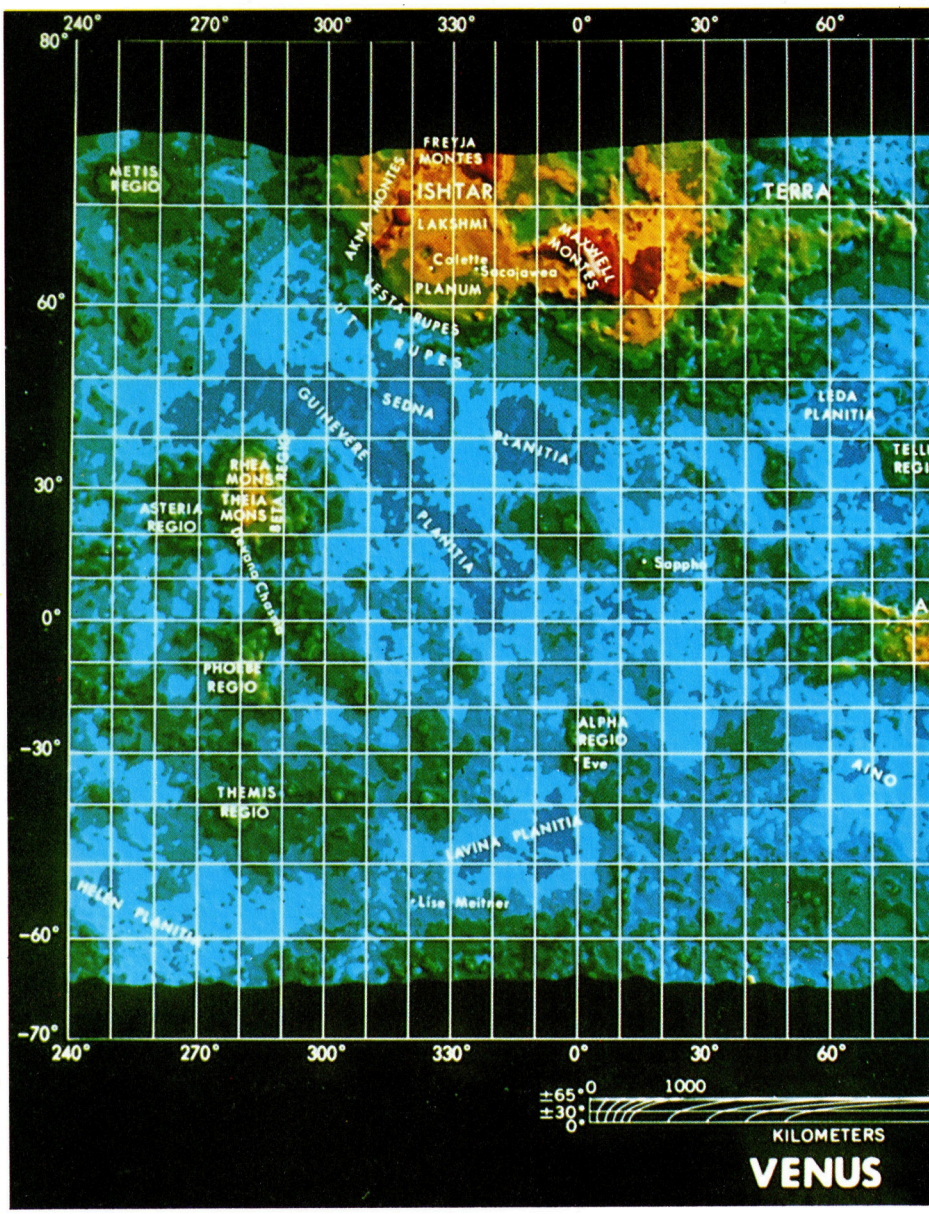
„Bottom-Up"-Systeme tasten die Bilddaten nach Linien, Kanten und anderen markanten Merkmalen ab. Damit wird eine einfache Beschreibung des Bildes konstruiert, das man „Rohskizze" nennt. Es handelt sich dabei um eine sehr abstrakte Darstellung. Der Sinn ist, daß man durch Verzicht auf Details die Skizze leichter mit gespeicherten Mustern vergleichen kann.

In der Praxis finden beide Erkennungsmethoden Anwendung. Erst seit kurzem gibt es in diesem Bereich auch lernfähige Systeme. Sie basieren auf vorprogrammierter Intelligenz.

Eine bemerkenswerte Ausnahme ist Igor Aleksanders WISARD (Wilkie, Aleksander & Stonham's Recognition Device). Dieses System kann selbst den vergleichsweise feinen Unterschied zwischen einem lächelnden und einem ernsten menschlichen Gesicht erkennen, und es kann sein Wissen sogar auf Gesichter übertragen, die es nie zuvor gesehen hat. WISARD arbeitet in Echtzeit (25 Bilder pro Sekunde).

WISARD arbeitet durch Zuweisung von jeweils acht Punkten zu einem ausgewählten RAM-Speicherbereich. Diese Achtergruppen (octuple) dienen als Merkmal-Detektoren, die sich jeweils in einem von 256 Zuständen befinden können (0 bis 255), – abhängig vom Status der jeweiligen acht Punkte. In der Lernphase wird eine Eins an der jeweiligen Adresse im RAM gespeichert, wenn ein bestimmtes Bild vorhanden ist. In der Erkennungsphase ist die Eins an der bestimmten Adresse Beweis, daß das „gedachte" Bild erneut vorhanden ist.

Durch Verwendung einer großen Zahl von Achtergruppen oder sogenannten „Unterscheidern" (Diskriminatoren) wird das System unempfindlich gegenüber falschen Daten (Störbildern). In Aleksanders System werden ein 512 × 512 messendes Bildraster und über 32 000 Achtergruppen verwendet.



0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0
0	0	1	0	1	1	0	0
0	1	1	0	0	0	1	0
0	0	1	1	1	1	0	0
0	0	1	0	0	1	0	0
0	1	1	0	0	1	0	0
0	1	1	0	0	0	1	0

0	0	0	0	0	0	0	0
0	0	0	1	1	1	0	0
0	0	1	0	0	1	0	0
0	1	0	0	0	1	0	0
0	1	1	1	1	0	0	0
0	1	1	0	0	1	0	0
0	1	0	0	0	1	0	0
0	1	0	0	0	0	1	0

Der Unterschied zwischen R und A

In der realen Welt gibt es nur selten exakt definierte Daten. Das obige Beispiel zeigt digitalisierte Formen der Buchstaben A und R. Doch keines der Muster ist genau definiert, sondern hat Übergänge und Abweichungen. „Störbilder" oder „Stördaten" wie diese verursachen Probleme bei der Mustererkennung. Wissen Sie, was welcher Buchstabe ist?

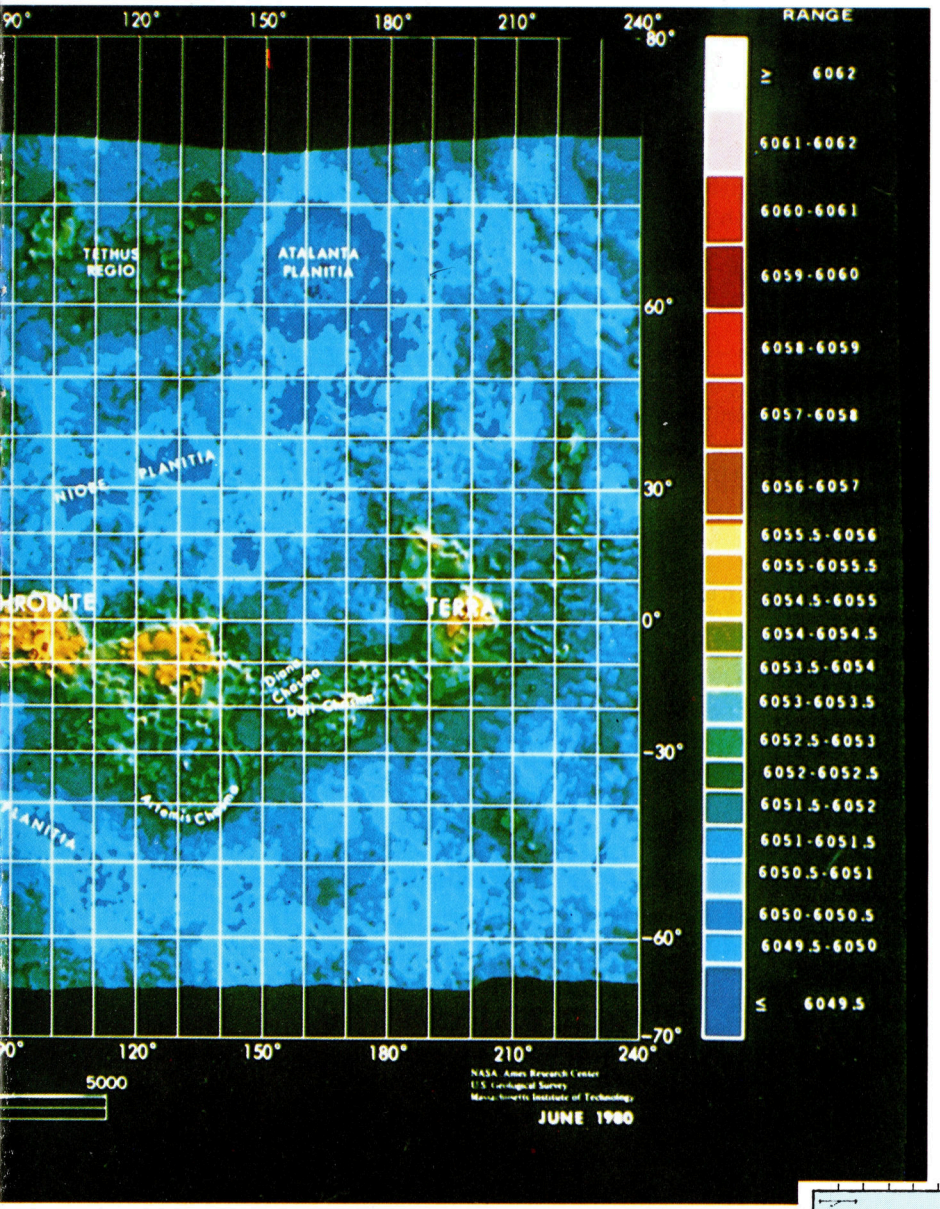


Bild-Verarbeitung

Das von einer Raumsonde aufgenommene Foto wurde von einem Computer bearbeitet, um die unterschiedlichen Oberflächenstrukturen zu verdeutlichen. Dazu verwendete man „Falsch-Farben". Interessant an diesem Bild des Planeten Venus ist, daß alle Farben „falsch" = künstlich erzeugt sind. Ohne Computerhilfe wäre ein solches Bild nicht sichtbar, da die Venus stets von dicken Wolken umgeben ist.

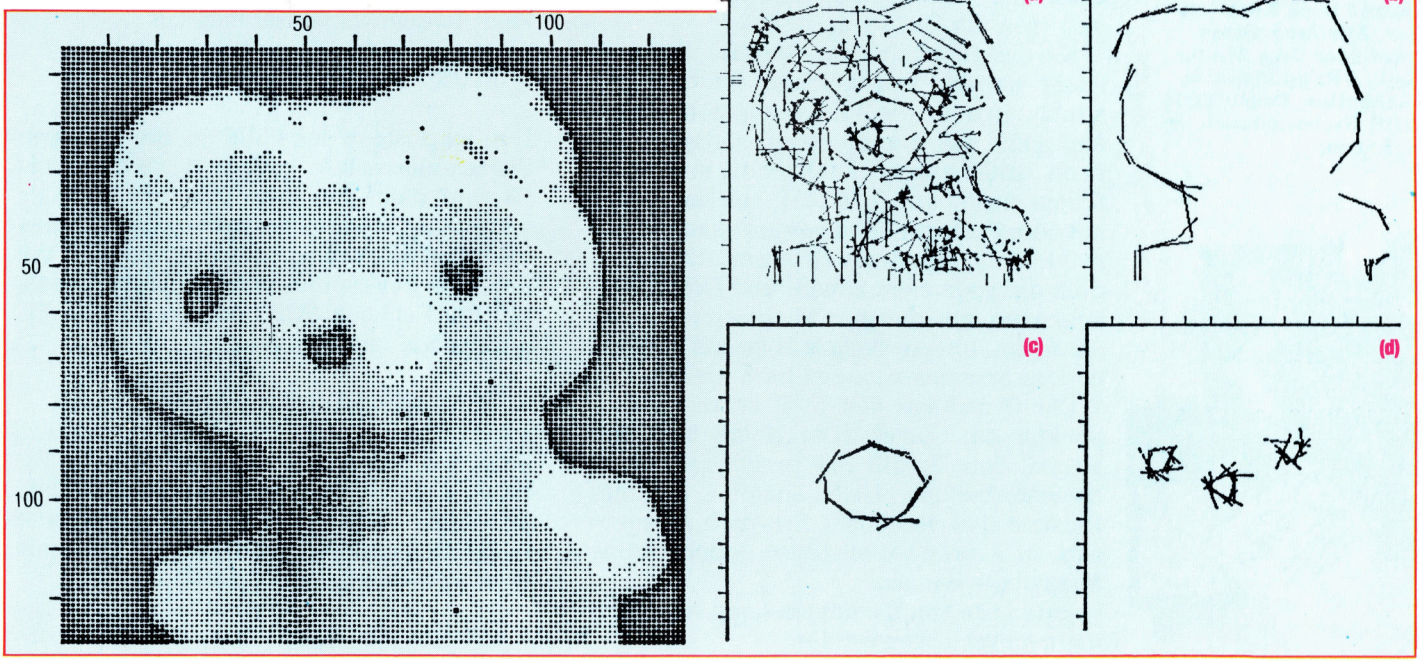
In der Astronomie haben sich solche Bilder als sehr nützlich erwiesen. Man nutzt dabei die Vorteile von Radio-, Infrarot- und Ultraviolett-Wellen ebenso wie das sichtbare Licht.

Auf der Erde beobachtet man die Meere mit Sonar-Wellen, um akustische Bilder zu erzeugen. Hierbei findet das Echolot-Prinzip Anwendung.

Andere Arten der „Bild-Verarbeitung" werden genutzt, um verzerrte oder unscharfe Bilder „zu reinigen".

Rohskizzen

Es gibt eine Methode der Bildverarbeitung, die nicht auf vorhandenen Erkenntnissen und gegebenem Wissen basiert. Die Rohskizze (a) isoliert einfache Konturen aus dem Abbild des Teddybären, indem es den Kontrast benachbarter Regionen vergleicht. Weitere Auszüge aus den Rohskizzen (b) bis (d) stellen wichtige Gruppen dar, die zur Erkennung des Originalbilds beitragen.



Listige Listen

Mit diesem Artikel beginnt eine Serie über LISP, in der wir die Anwendung dieser Sprache beschreiben und untersuchen, warum sie im Bereich der Künstlichen Intelligenz so weit verbreitet ist.

LISP wurde in den letzten Jahren hauptsächlich durch ihren Einsatz im Bereich der KI bekannt. Im Laufe der Zeit wurde dann deutlich, daß LISP sich als Allzwecksprache für einen breiten Anwendungsbereich eignet.

LISP taucht inzwischen in vielen sehr unterschiedlichen Bereichen auf. Unter LISP entstanden Betriebssysteme, Compiler und sogar Abenteuerspiele. Doch trotz der vielen LISP-Dialekte, die durch die weite Verbreitung der Sprache entstanden, halten sich die meisten LISP-Programme (aufgrund der einfachen und direkten Sprachstruktur) an einen Standard und lassen sich leicht von einer Maschine auf eine andere übertragen.

Beim Gebrauch von LISP wird man früher oder später bestimmte Funktionen und Befehle vermissen. Da ein „offizieller" Standard fehlt, haben die Programmierer der einzelnen Sprachversionen nur die Funktionen eingebaut, die ihnen wichtig erschienen. Die Sprache kann jedoch leicht um zusätzliche Befehle erweitert werden.

LISP unterscheidet sich grundlegend von bekannteren Sprachen wie PASCAL, FORTRAN und BASIC. Durch ihren sehr einfachen und einheitlichen Aufbau eignet sie sich ideal für Datenverarbeitung.

Die Listenstruktur der Sprache läßt sich leicht an die wesentlichen Informationsstrukturen des jeweiligen Computers anpassen. LISP selbst kann dabei sortieren, suchen, arithmetische Funktionen und sogar Spiele ausführen. Mit LISP (insbesondere mit dem in dieser Serie vorgestellten Acornsoft-LISP für den Acorn B) lassen sich außerdem die speziellen Sound- und Grafikmöglichkeiten der meisten Microcomputer ansprechen. Unsere Beispiele lassen sich auf andere Sprachversionen übertragen.

Die Grundlage von LISP ist die Datenstruktur der „Liste". (Der Name LISP entstammt dem Begriff „list processing" – Listenverarbeitung.) Listen ähneln in mancher Hinsicht den vertrauten Arrays. Im Gegensatz zu Arrays haben Listen jedoch keine festgelegte Länge.

Jede Liste von Elementen kann von Klammern umschlossen werden:

```
(a b c d e . . .)
```

wobei a, b, c etc. die Bestandteile der Listen sind. Die einzelnen Teile werden dabei „Atome" genannt. Sie können Zahlen, nichtnumerische Daten (Zeichen oder Variablen) oder auch andere Listen darstellen. Beachten Sie, daß Listenelemente nicht durch Kommas, sondern durch Leerzeichen getrennt werden.

Die Bearbeitung der Listen geschieht mit Funktionen, die ähnlich wie die BASIC-Funktion DEF FN Argumente verarbeiten und Ergebnisse liefern. Sie werden als

```
(func a b c d . . .)
```

definiert, wobei „func" den Namen der Funktion darstellt und a, b, c etc. die Argumente. Die Funktion bildet dabei das erste Element der Liste.

Eine Liste der ersten sechs Primzahlen wird folgendermaßen geschrieben:

```
(1 2 3 5 7 11)
```

Eine Addition (hier in BASIC):

```
1+2+3+5+7+11
```

arbeitet in LISP mit der PLUS-Funktion:

```
(PLUS 1 2 3 5 7 11)
```

und liefert das Ergebnis 29. Da die Funktion PLUS beliebig viele Argumente enthalten kann, ist auch folgendes möglich:

```
(PLUS 1 2 3 (PLUS 5 7 11))
```

Hier wird zuerst das PLUS mit der höchsten Schachtelungstiefe bewertet. Das Ergebnis 23 wird dann von dem äußeren PLUS als viertes Argument eingesetzt – als Ergebnis erscheint 29. Wichtig ist die Leichtigkeit, mit der sich Funktionen verschachteln lassen. Mit der Funktion SETQ ordnen wir das Ergebnis der Variablen A zu:

```
(SETQ A (PLUS 1 2 3 5 7 11))
```

SETQ hat dabei zwei Argumente – die Variable A und die Funktion, die die Ganzzahl 29 ergibt. SETQ ist natürlich selbst eine Funktion und liefert wiederum ein Ergebnis (hier den Wert 29). Die Zuordnung:

```
LET B = 1+2+3+5+7+11
```

Trotz des wachsenden Interesses an LISP gibt es nur wenige preiswerte Sprachversionen für Microcomputer. Das Acornsoft-LISP läuft auf dem Acorn B und dem Acorn Electron.

Für das Betriebssystem CP/M sind mehrere LISP-Versionen erhältlich, darunter Toolworks LISP/80, iLISP und muLISP-83 – die CP/M-Version von Microsoft.

Die mit CP/M arbeitenden Interpreter sollten auf allen CP/M-Micros laufen. Toolworks bringt 3600 Listenzellen und 11000 Namenszeichen für Atome in 48 KByte unter. Alle Programme sind über Gray Matter Ltd., 4 Prigg Meadow, Ashburton, Devon TQ13 7DF, Großbritannien, zu erhalten.

ist den meisten BASIC-Versionen nicht möglich, läßt sich in LISP aber als:

```
(SETQ B (SETQ A (PLUS 1 2 3 5 7 11)))
```

schreiben. Durch das Einsetzen der Funktion TIMES kann der doppelte Wert von A der Variablen B zugeordnet werden:

```
(SETQ B (TIMES 2 (SETQ A (PLUS 1 2 3 5
7 11))))
```

Dieser Ausdruck bewertet zuerst das PLUS und errechnet den Wert 29. Dieser wird von der inneren SETQ-Funktion der Variablen A zugeordnet, die das Ergebnis (29) als zweites Argument an die TIMES-Funktion weiterreicht. Das neue Ergebnis (58) wird nun an die äußere SETQ-Funktion übergeben, die diesen Wert in die Variable B setzt. Das Ergebnis des Ausdrucks ist also der Wert 58, der in weiteren Funktionen verwendet werden kann.

Selbständige Klammerauflösung

Aufgrund der vielen Klammern wird der Ausdruck sehr unübersichtlich. In einem sorgfältig aufgebauten Programm erledigt sich die Auflösung der Klammern jedoch fast von selbst. Einige LISP-Systeme geben hier Hilfestellung, indem sie über die Anzahl der offenen Klammern informieren. Das Acornsoft-LISP zeigt beispielsweise am Zeilenanfang durch Pfeile an, wie viele Klammern noch geschlossen werden müssen, bevor der Ausdruck vollständig ist.

In unserem letzten Beispiel hatte TIMES zwei Argumente: die Ganzzahl 2 und einen Listenausdruck, der zuvor auf das Ergebnis 29 gesetzt wurde. TIMES kann aber wie PLUS beliebig viele Argumente besitzen. Jedes der folgenden Beispiele ist legal und ergibt 1024:

```
(TIMES 1 2 4 8 16)
(TIMES 1 2 4 8
(TIMES 4 4))
(TIMES 1 2 (TIMES 2 2) (TIMES 2 4)
(PLUS 8 8))
```

In der Praxis begrenzen die meisten LISP-Systeme jedoch die Anzahl der Funktions-Argumente. Das Acornsoft-LISP unterstützt beispielsweise nur 28 Argumente, während andere Versionen die Anzahl noch mehr einschränken.

Die Beispiele zeigen deutlich, daß LISP-Befehle nur aus Argumentenlisten bestehen, in denen das erste Element die auszuführende Funktion angibt, während die darauf folgenden Elemente Argumente dieser Funktion sind. Diese Argumente können wiederum Listen sein, deren erste Elemente als Funktionen Ergebnisse liefern. Was geschieht jedoch, wenn wir keine Funktion benötigen, sondern nur eine Liste mit Datenelementen, die wir als Titel einsetzen wollen? Wir können

```
(COMPUTER KURS LISP SERIE)
```

schlecht als Liste mit vier Datenelementen anlegen, da LISP versuchen würde, COMPUTER als Funktionsname zu bewerten und die Argumente (KURS, LISP und SERIE) als drei Variablennamen ansieht. Wir können LISP jedoch durch ein voranstehendes (') mitteilen, daß der Ausdruck nicht bewertet werden soll:

```
'(COMPUTER KURS LISP SERIE)
```

läßt sich dann wiederum als Liste mit vier nicht-numerischen Elementen der Variablen MAG zuordnen:

```
(SETQ MAG '(COMPUTER KURS LISP
SERIE))
```

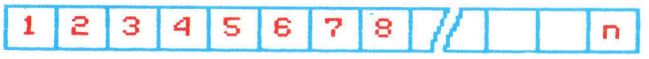
Beachten Sie, daß es in LISP keine unterschiedlichen Variablentypen gibt. Die folgenden Ausdrücke sind in LISP daher alle völlig legal:

(SETQ A 3)	A = die Ganzzahl 3
(SETQ A 'COMPUTER)	A = der String 'COMPUTER'
(SETQ A (PLUS 2 4 8))	A = das Ergebnis von 2+4+8
(SETQ A '(1 2 4 8))	A = die Liste (1 2 4 8)
(SETQ A '(COMPUTER KURS))	A = die LISTE (COMPUTER KURS)
(SETQ A B)	A = der Wert der Variablen B

Es gibt nur wenige LISP-Versionen, die mit Fließkommaarithmetik arbeiten. In den meisten Fällen reichen die Ganzzahlen aus, wobei sich Fließkommaberechnungen ohne größere Schwierigkeiten durch normale Ganzzahlen simulieren lassen.

In der nächsten Folge werden wir sehen, wie LISP-Funktionen die Datenargumente bearbeiten und wie flexibel die Sprache bei Recursionen ist.

ARRAY (mit DIM auf die Länge n gesetzt)



LIST (keine feste Länge)

Listen haben Arrays gegenüber zwei große Vorteile. Zunächst muß vor ihrem Einsatz kein Speicherplatz für sie reserviert werden. Da sie dynamische Strukturen sind (das heißt keine feste Länge haben), können sie sich während der Programmausführung an die Menge der Daten anpassen. Listen lassen sich außerdem bei der Programmierung Künstlicher Intelligenz für recursive Prozeduren einsetzen.

Qual der Wahl

Die Menüsteuerung vieler Programme bietet dem Anwender in bestimmten Situationen ein Verzeichnis der möglichen weiteren Schritte an. Befehlsgesteuerte Systeme arbeiten anders, sie ermöglichen den Eingriff ins Programmgeschehen nur mit Hilfe von Kommandos. Dennoch haben auch sie Vorteile.

Ein Menü kann sowohl aus einer einfachen Liste numerierter Kurzbegriffe wie auch aus Symbolen aufgebaut sein – das Prinzip bleibt gleich. Menüs werden gebraucht, wenn die Programmlogik Verzweigungen zuläßt. Der Anwender kann aus einem auf dem Bildschirm dargestellten Angebot von Möglichkeiten wählen. Menügesteuerte Programme haben meist die Struktur eines verzweigten Baumes, den der Anwender an der Wurzel betritt und unter Anleitung des Menüs bis zu den feinsten Verästelungen durchläuft.

Eine gute Menüsteuerung erwartet vom Anwender keine genauen Kenntnisse der Programmstruktur. – Dem Anwender werden ständig Wegweiser angeboten, die ihm ein sicheres „Geleit" durchs Programm geben. Der geübte Computerbenutzer empfindet aber gerade diese Hilfe oft als lästig. Zu einem schnell benötigten Programmteil muß er sich häufig erst durch mehrere Menüs seinen Weg bahnen. Manchmal haben auch Laien mit der Baumstruktur ihre Probleme: Eine falsche Entscheidung kann oft nur nach der Rückkehr zu einem bereits bekannten und erledigt geglaubten Menü geändert werden, was nicht immer ganz einfach ist. Menüs sind allerdings gelegentlich auch in anderer Form organisiert – nicht als Baum, sondern als Netzwerk mit vielfältigen Querverbindungen und Schleifen.

Labyrinth der Möglichkeiten

Der Entwurf einer Menüsteuerung kann schwierig sein, auch wenn die eigentliche Programmierung unkompliziert ist. Hauptproblem ist, daß bereits vor der Programmerstellung die Eigenschaften und Möglichkeiten des Programms beschrieben werden müssen. Wer später zusätzliche Funktionen einführen möchte, muß nicht selten mehrere Menüs ändern oder gar die ganze Programmstruktur neu organisieren. Die gesamte Logik der Menüsteuerung sollte in einer einzigen Routine liegen, die beim Erreichen einer Verzweigung das entsprechende Unterprogramm aufruft. Die Menü-Routine dient ausschließlich als erweiterte Programmsteuerung. Welche Richtung eingeschlagen wird, entscheidet der Anwender allein. Das erleichtert nicht nur den Programmentwurf, sondern trennt auch die Programmsteuerung von den funktionellen Programmteilen. So können beide Bereiche unabhängig voneinander verbessert bzw. Fehler einfacher beseitigt werden.

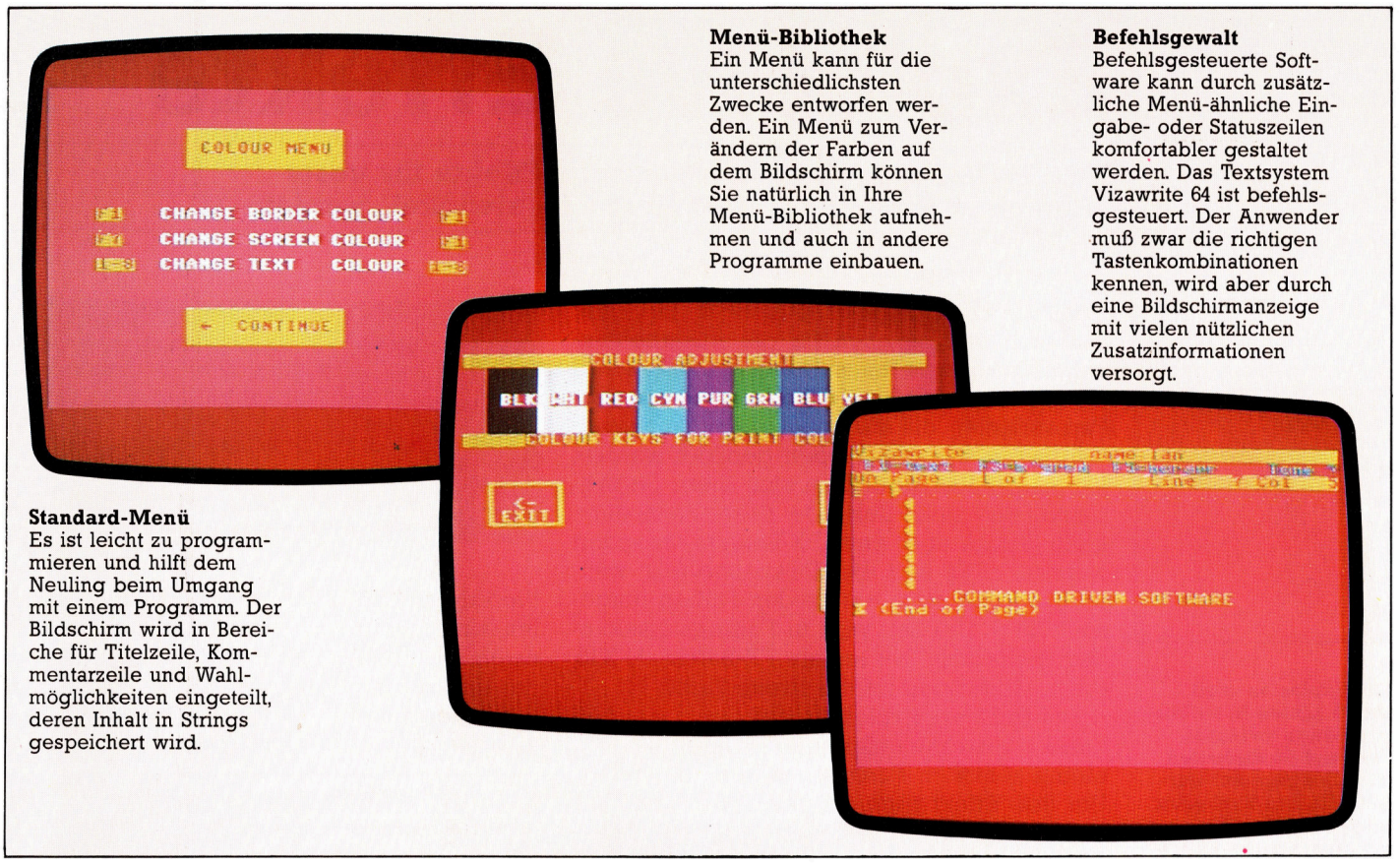
Menüstrukturen

Der Programmablauf folgt einem festen Muster. Bei jedem Menü auf dem Weg durch das Programm gibt die Menülogik einen Satz Anweisungen zu einem Programmteil weiter, der diesen in die „Leerstellen" eines „Menü-Rahmens" einsetzt. Die Leerstellen werden mit der Titelzeile und nötigen Zusatzinformationen gefüllt und enthalten neben den Wahlmöglichkeiten auch Anweisungen für die Bedienung des Menüs. Ein professionelles Menü-Layout enthält bis zu acht Optionen in einer Spalte. Links neben jeder Option sollte der dazugehörige Code für die Antwort angezeigt werden.

Die Menü-Routine ruft eine INPUT-Routine auf, die zwischen erlaubten und unzulässigen Eingaben unterscheiden sollte. Die Eingabe (meist ein einziger Tastendruck) muß interpretiert werden und führt dann entweder zum nächsten Menü oder zum gewünschten Programmteil. Nach der Ausführung des gewählten Programmteils wird entweder das letzte Menü nochmals gezeigt oder zu einem anderen Programmteil (etwa dem Eingangsmenü) gesprungen.

Menüs brauchen für Titelzeile, Anweisungen und Wahlmöglichkeiten relativ viel Text. Ein großer Teil davon kann durch den Menü-Rahmen wiederholt werden. Auch ein „Help"-Befehl, mit dem man sich die Bedienung erklären lassen kann, und ein Befehl für den direkten Zugang zum Hauptmenü treten immer wieder auf. Wer Platz sparen und die Programmlogik klar nachvollziehbar halten möchte, sollte den entsprechenden Text in einem String-Array speichern, von wo er durch Angabe der Indexnummer aufgerufen werden kann.

Befehlsgesteuerte Programme verfügen über einen Satz von Kommandos, die der Anwender zu jeder Zeit aufrufen kann. Jeder Befehl verzweigt sofort zu einem Unterprogramm, das die gewünschte Funktion ausführt. Jede Eingabe muß bei dieser Technik darauf geprüft werden, ob es sich um Daten oder einen Befehl handelt. Zur klaren Trennung werden



Menü-Bibliothek
Ein Menü kann für die unterschiedlichsten Zwecke entworfen werden. Ein Menü zum Verändern der Farben auf dem Bildschirm können Sie natürlich in Ihre Menü-Bibliothek aufnehmen und auch in andere Programme einbauen.

Befehlsgewalt
Befehlsgesteuerte Software kann durch zusätzliche Menü-ähnliche Eingabe- oder Statuszeilen komfortabler gestaltet werden. Das Textsystem Vizawrite 64 ist befehlsgesteuert. Der Anwender muß zwar die richtigen Tastenkombinationen kennen, wird aber durch eine Bildschirmanzeige mit vielen nützlichen Zusatzinformationen versorgt.

Standard-Menü
Es ist leicht zu programmieren und hilft dem Neuling beim Umgang mit einem Programm. Der Bildschirm wird in Bereiche für Titelzeile, Kommentarzeile und Wahlmöglichkeiten eingeteilt, deren Inhalt in Strings gespeichert wird.

Meist gibt es unterschiedliche Wege zum gleichen Ziel – die oben vorgestellten Bildschirmdarstellungen stammen aus drei verschiedenen Programmen und dienen alle demselben Zweck: dem Verändern der dargestellten Farben.

Befehlseingaben meistens mit der CONTROL-Taste eingeleitet.

Der Strukturbaum eines befehlsgesteuerten Programms ist sehr breit gefächert: Eine einzige Steuer-Routine leitet den Anwender zum gewünschten Unterprogramm. Dieses Steuerprogramm (Command Interpreter) hat vier Aufgaben zu erfüllen: Es muß auf eine Eingabe warten; danach soll die Eingabe analysiert, das heißt in ihre funktionalen Einheiten zerlegt werden; im dritten Schritt wird der Befehl in die Vorbereitung des gewählten Unterprogramms umgesetzt (An welcher Adresse steht das Unterprogramm? Müssen Parameter übertragen werden?). Erst der vierte Schritt ist dann der Aufruf des Unterprogramms.

Befehle können ein ganz unterschiedliches „Sprachniveau" haben. Bei UNIX sieht eine typische Befehlszeile etwa so aus:

Befehl + Liste der Parameter (optional)
Zum Beispiel:

```
L
```

oder L-1

Der Befehl L allein listet ein File-Verzeichnis. Mit L-1 (optionaler Parameter −1) erhält man das Verzeichnis in Langform.

Bei der Interpretation einer Zeile müssen die einzelnen Teile des Befehls voneinander getrennt werden. In UNIX ist das erste Wort meist der Befehl, Parametern wird ein Minuszeichen vorangestellt. Die Parameter einer Kommandosprache betreffen nicht den Interpreter selbst, sondern das von ihm aufgerufene Unterprogramm. Bei befehlsgesteuerten Systemen ist ein Standardformat für die Parametereingabe vorteilhaft, weil einzelne Parameter dann in der ursprünglichen Form (beispielsweise als String) übergeben werden können.

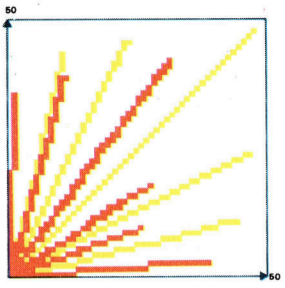
Befehle: schnell und direkt

Die Entwicklung eines befehlsgesteuerten Systems ist nicht nur einfacher als das Schreiben eines Menüprogramms, das befehlsgesteuerte System arbeitet auch schneller und ist zudem flexibler. Befehlsgesteuert sind auch die meisten Betriebssysteme – für den Neuling ein Nachteil, weil er ohne besondere Hinweise zurechtkommen muß und sogar Hilfsroutinen (falls überhaupt vorhanden) nur mit einiger Systemkenntnis zu starten sind. Durch die Vielfalt der möglichen Befehle und Parameter kann aber auch ein mit dem System vertrauter Anwender kaum ohne Handbuch arbeiten. Dennoch: Handelt es sich um Programme, die oft eingesetzt werden, schleifen sich die Befehle ein, und man braucht das Handbuch noch viel seltener.

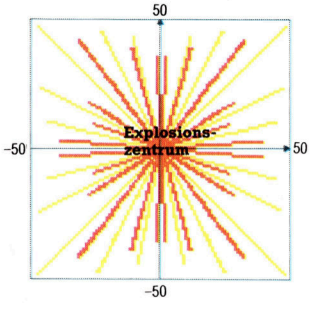
Die meisten Anfänger mögen Befehlssysteme nicht. Bei Profis dagegen trifft die Menüsteuerung selten auf Gegenliebe. Allenfalls gemischte Programme können diesen Konflikt ausschalten – so hat etwa WordStar ein Befehlssystem, kann vom Anwender aber wie ein menügesteuertes Programm eingesetzt werden. Von den Menüs werden die Befehlscodes als Eingabekürzel verwendet, die der Neuling schnell lernt.

Explosive Effekte

Jetzt sollen optische und akustische Effekte für unser Minenfeld-Projekt auf dem Acorn B entwickelt werden.



Die Explosion wird durch Linien in zufälliger Länge zwischen −50 und 50 Einheiten vom Zentrum aus dargestellt. Die Explosion wird in einem Quadranten ausgearbeitet und dann in die restlichen drei Quadranten des Koordinatenkreuzes übertragen.



Die Acorn-Computer bieten 16 mögliche Farbkombinationen für die Ausarbeitung eines Explosionseffekts. Dabei handelt es sich um acht verschiedene Farben sowie um acht Farbvarianten, die aus jeweils zwei nacheinander aufleuchtenden Farben gebildet werden. Normalerweise verändern sich diese Farbvarianten alle halbe Sekunde. Diese Zeitspanne kann jedoch über zwei FX-Befehle verändert werden. *FX9 legt die Zeitspanne fest, während der die erste der beiden Farben dargestellt wird. Die Zeit wird in Fünfzigstelsekunden gemessen. Der andere FX-Befehl, *FX10, gilt entsprechend für die zweite Farbe.

Die sichtbare Explosion wird durch mehrere kurze, verschiedenfarbige Linien, die von einem zentralen Punkt ausgehen, dargestellt. Um den Effekt noch zu steigern, können die Farben der Minen und der Wertungsanzeige ebenfalls verändert werden. Beide Darstellungen werden zunächst in der logischen Farbe 2 (Grün) angezeigt. Diese Farbe kann mit folgendem Befehl geändert werden:

```
VDU19,2,RND(15),0,0,0
```

RND(15) wählt eine ganze Zahl zwischen eins und fünfzehn aus.

Die für die Explosionslinien verwendete Farbe kann über GCOL 0,RND(3) zufällig gewählt werden. Bisher verwendeten wir die hochauflösenden Befehle MOVE und DRAW. DRAW(X,Y) zeichnet immer eine Linie zu einem „absoluten" Koordinatenpunkt (X,Y). Es gibt aber noch eine andere Gruppe von Zeichenbefehlen, mit denen „relative" Punkte spezifiziert werden können. PLOT K,X,Y zieht Linien zwischen relativen oder absoluten Punkten, abhängig vom Wert K. Die Tabelle oben rechts zeigt einige der möglichen Variationen mit PLOT.

K=0 Bewegung relativ zum letzten Punkt
K=1 Ziehe Linie zur relativen Position in Vordergrundfarbe
K=2 Ziehe Linie zur relativen Position in logischer, inverser Farbe
K=3 Ziehe Linie zur relativen Position in Hintergrundfarbe
K=4 Bewegung zum absoluten Punkt
K=5 Ziehe Linie zur absoluten Position in Vordergrundfarbe
K=6 Ziehe Linie zur absoluten Position in logischer, inverser Farbe
K=7 Ziehe Linie zur absoluten Position in Hintergrundfarbe

PROCexplode

PLOT4 und PLOT5 entsprechen genau den Befehlen MOVE und DRAW. Für unseren Explosionseffekt brauchen wir PLOT1, um Linien relativ zum Explosionszentrum zu ziehen.

Im letzten Kapitel wurde die Testroutine PROCexplode gezeigt, in der die Werte von xgraph und ygraph an x_explode und y_explode übertragen wurden. In der Testroutine hatte dies keine Auswirkung. Jetzt wird damit das Explosionszentrum definiert. Wird diese Prozedur in Zeile 3390 von PROCmove aufgerufen, sind xgraph und ygraph die Grafikkoordinaten des Mittelpunkts der Zeichenzelle, in der sich die Mine befindet. MOVE x_explode, y_explode bewegt den Grafik-Cursor ins Zentrum der gewählten Explosionsstelle.

Nun der Sound

Sobald wir mit MOVE dort angelangt sind, ziehen wir eine Linie in eine beliebige Richtung, die maximal 50 Einheiten lang sein sollte.

```
PLOT 1,RND(50),RND(50)
```

Dieser Befehl erfüllt seinen Zweck nicht ganz, da keine negativen Koordinaten angegeben werden (eine negative X-Koordinate zieht eine Linie nach links, eine negative Y-Koordinate entsprechend nach unten). Die wiederholte Verwendung dieses Befehls füllt nur ein Viertel der Fläche um das Zentrum herum aus.

Um negative Werte einsetzen zu können, muß RND(50) gegen RND(100)−50 ersetzt werden. Durch mehrmaliges MOVE zum Explosionszentrum und Ziehen relativer Linien mit maximal 50 Einheiten Länge in zufällige Richtungen und in verschiedenen Farben erzielt man überraschende farbkräftige Darstellungen der Explosion.

Wie werden nun Klangeffekte erstellt, die die Explosion akustisch untermalen? Mit den BASIC-Befehlen SOUND und ENVELOPE können eindrucksvolle Töne generiert werden. SOUND erstellt Geräusche für Spezialeffekte. ENVELOPE wird zusammen mit SOUND verwendet, um den erzeugten Klang zu variieren.

SOUND hat vier zugeordnete Zahlen (bzw. Parameter), über die die Charakteristiken der Töne gesteuert werden:

```
SOUND C,A,P,D
```

- C ist die Kanalnummer und liegt zwischen 0

und 3. Kanal 1–3 produziert Noten. Kanal 0, den wir hier verwenden, ist für spezielle Klangeffekte reserviert.

- A bestimmt die Lautstärke. Der Maximalwert beträgt A-15. Bei A = 0 ist kein Ton hörbar. Positive Werte für A werden in Kombination mit ENVELOPE für die Schattierung der Klänge verwendet.
- P ist die Tonhöhe der Note.
- D gibt die Dauer oder Länge der gespielten Note an. Werte für D von 0 bis 254 lassen eine Note für eine in Zwanzigstelsekunden gemessene Zeit klingen. Um also eine Note eine Sekunde klingen zu lassen, muß D auf 20 gesetzt werden. Wird D auf −1 gesetzt, erklingt der Ton so lange, bis er von Ihnen abgebrochen wird.

Der Klang der Explosion muß natürlich sehr laut sein. Daher wird A auf −15 gesetzt. Die Einstellung für P ist etwas komplizierter. In Kombination mit Kanal 0 nimmt P Werte zwischen null und sieben gemäß an. Die genaue Belegung können Sie dem untenstehenden Kasten entnehmen.

Die vier Impuls-Wellenformen ergeben Brummtöne, während die Geräusch-Wellenformen eher ein Rauschen produzieren. Der in unserem Programm verwendete SOUND-Befehl lautet:

```
SOUND 0,−15,4,40
```

Er bewirkt einen hohen Zischlaut in voller Lautstärke über zwei Sekunden. Nachfolgend sehen Sie das komplette Listing dieser Prozedur:

```
3550 DEF PROCexplode(x_explode,y_explode)
3560 REM ** SOUND EFFECT **
3570 SOUND 0,−15,6,50
3580 REM ** SET FLASH RATE **
3590 *FX9,20
3600 *FX10,50
3610 FOR I=1 TO 100
```

P=0 Impuls in Hochfrequenz
P=1 Impuls in mittlerer Frequenz
P=2 Impuls in niedriger Frequenz
P=3 Impuls, Frequenzeinstellung über Kanal 1
P=4 Geräusch in Hochfrequenz
P=5 Geräusch in mittlerer Frequenz
P=6 Geräusch in niedriger Frequenz
P=7 Geräusch, Frequenzeinstellung über Kanal 1

```
3620 MOVE x_explode,y_explode
3630 VDU19,2,RND(15),0,0,0
3640 GCOL 0,RND(3)
3650 PLOT 1,RND(100)−50,RND(100)−50
3660 NEXT I
3670 PROCreset
3680 ENDPROC
```

Nach einer Explosion sind folgende Dinge nacheinander zu erledigen:

- Verringerung der Anzahl verbleibender Leben. Überprüfung, ob alle Leben verbraucht sind.
- „Aufräumungsarbeiten" nach der Explosion auf dem Bildschirm.
- Plazierung von Suchgerät und Assistent an ihrer Startposition.

„end flag"-Zähler

Die erste Aufgabe ist über einen Zähler zu lösen, der bei jedem Aufruf der Prozedur erhöht wird. Übersteigt der Zähler den Wert 4, wird eine Variable „end flag" auf 1 gesetzt, um das Spielende anzuzeigen.

Danach müssen die Explosionsspuren vom Bildschirm verschwinden. Wir könnten dazu zum Beispiel Leerstellen über die gezeichnete Explosionsfläche PRINTen. Wir aber wollen den gesamten Bildschirm bereinigen, die Minen neu setzen sowie Zeit und Wertung neu anzeigen. Fairerweise sollten nur die Minen neu gesetzt werden, die vor der Explosion vorhanden waren. Das kann anhand der Wertung leicht errechnet werden. Da jede Mine mit 150 Punkten bewertet wird und der Anfangsstand 50 war, können wir die Anzahl noch verbleibender Minen relativ leicht errechnen und direkt an die Prozedur „lay mines" in Zeile 3960 übertragen.

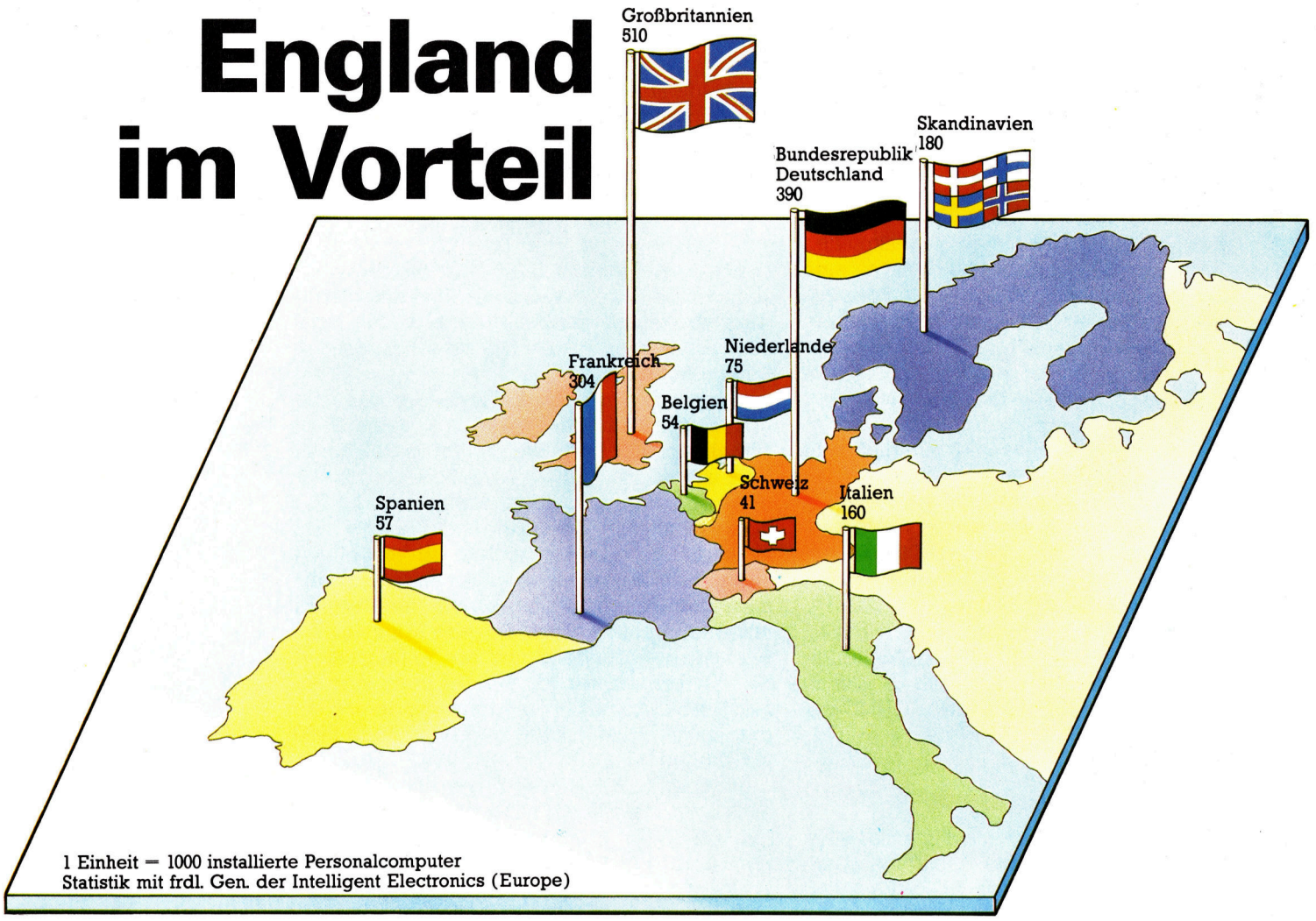
Suchgerät und Assistent werden durch Initialisieren ihrer Koordinaten und durch Aufrufen von „position chars" an ihre alten Positionen zurückgesetzt. Nachfolgend sehen Sie das komplette RESET-Listing. Fügen Sie diese Zeilen und die explode-Prozedur in das Programm ein.

```
3880 DEF PROCreset
3890 count=count+1
3900 IF count>4 THEN end_flag=1:ENDPROC
3910 CLS
3920 VDU19,2,2,0,0,0
3930 COLOUR 2
3940 PROCinitialise_variables
3950 mines_left=50−score/150
3960 PROClay_mines(mines_left)
3970 PROCdraw_border
3980 PRINTTAB(2,27);"Time"
3990 PRINTTAB(2,28)"Score"
4000 PRINTTAB(11,28)score$
4010 PRINTTAB(2,29)"Hi score"
4020 PRINTTAB(11,29)hi_score$
4030 remaining_men$=LEFT$(men$,4−count)
4040 COLOUR 1
4050 PRINTTAB(2,30);remaining_men$;" "
4060 COLOUR 2
4070 PROCposition_chars
4080 ENDPROC
```

Im bereits gezeigten „Calling-Programm" muß folgendes geändert werden:

```
60 UNTIL TIME>12099 OR end flag = 1
```

England im Vorteil



Bei der Markteinführung der Heimcomputer hatte England eine denkbar gute Startposition. Zwischen Amerika und England gibt es keine Sprachbarrieren – auch deshalb stehen in Großbritannien mehr Computer als in irgendeinem anderen europäischen Land. Aber das Blatt scheint sich zu wenden. Fachleute erwarten, daß die Bundesrepublik bereits 1988 das Vereinigte Königreich überrunden wird.

Zu Beginn des Computer-Zeitalters gab es nur eine Sprache, in der Programme und Dokumentationen erstellt wurden: Englisch. In Zukunft sollen aber auch die Länder nicht mehr zu kurz kommen, in denen andere Sprachen gesprochen werden. – Die Softwarehäuser passen ihre Programme den verschiedenen Landessprachen an.

Bis vor kurzer Zeit erhielt man Programme und Dokumentationen ausschließlich in englischer Sprache – eine Erblast aus den fünfziger und sechziger Jahren, als noch fast jeder Computer in den USA gebaut wurde.

Die Sprachbarriere wurde zu Beginn dieses Jahrzehnts beim Durchbruch der Microcomputer zu einem erheblichen Problem. Der europäische Markt zeigte sich – von England abgesehen – widerspenstig, ein Teufelskreis entstand: Die meisten Anwender bestanden – verständlicherweise – auf Programmen, die in ihrer Sprache dokumentiert waren. Die Softwarehäuser hingegen waren nicht bereit, die Übersetzungs- und Entwicklungskosten aufzubringen. Die Verkaufsziffern im vielsprachigen Europa würden die hohen Kosten für die Erstellung der vielen verschiedenen Versionen nicht decken können. Die Folge: England wurde, durch die Sprache begünstigt, zur führenden Computer-Nation Europas.

Es scheint allerdings so zu sein, daß dieser Vorsprung in Kürze schrumpfen wird: Nach langem Zögern haben die Softwarehersteller das große Potential des kontinentalen Europa erkannt und machen mit der oft verschobenen Übersetzung ihrer Programme doch noch Ernst. Mit ersten anderssprachigen Programmen für kommerzielle IBM-Rechner war Lotus Software dabei der Vorreiter.

Lotus 1-2-3 und Symphony gehören zu den größten Verkaufserfolgen bei der „integrierten Software". Diese Programmpakete bestehen meist aus einem Kalkulationsprogramm, einer Datenbank sowie Textverarbeitung und Grafikmöglichkeiten, wobei Daten zwischen Einzelprogrammen ausgetauscht werden können.

Auf den ersten Blick erscheint die Übersetzung der Programme beispielsweise aus dem Englischen ins Italienische einfach – eine simple Übertragung der Kommentarzeilen in die Landessprache. Aber eben das ist schwierig – besonders wenn Teile des Textes im Quellprogramm enthalten sind. Ist der übersetzte Text – bei Übertragungen aus dem Englischen ins Deutsche ist das in der Regel der Fall – länger als das Original, kommt ein erhöhter Bedarf an Speicherplatz hinzu. Also verschieben sich die Adressen aller folgenden Befehle. Schleifen und Unterprogramme arbeiten in einem solchen Fall nicht mehr korrekt.

Schwierig ist auch das Eingabeformat. Wenn ein Engländer mit einem File arbeiten will, lautet die Syntax COMMAND gefolgt von FILENAME. Andere Länder, andere Sitten: In Deutschland müßte die Eingabe des Filenamens dem Sprachgebrauch entsprechend eigentlich an erster Stelle erfolgen. Ähnliche Schwierigkeiten gibt es bei der Eingabe eines Datums, wofür in Amerika die Reihenfolge Monat/Tag/Jahr, in den meisten Ländern Europas aber Tag/Monat/Jahr üblich ist. Ein Software-Paket muß also sehr sorgfältig an die nationalen Eigenheiten angepaßt werden.

Die unterschiedlichen nationalen Zeichensätze in Europa machen den Softwarehäusern weiteres Kopfzerbrechen. Im Französischen gibt es Buchstaben wie é und à, und das deutsche und skandinavische Alphabet enthalten Umlaute. Damit nicht genug, in verschiedenen Alphabeten stehen diese Sonderzeichen auch noch in unterschiedlicher Reihenfolge. Ein Programm, das diese Besonderheiten nicht berücksichtigt, kann Verheerendes anrichten.

Bei der Übersetzung des Programms Symphony in die wichtigsten europäischen Sprachen kam man bei Lotus zu der Einsicht, daß nur ein „übersetzungsfreundlich" geschriebenes Programm mit vertretbarem Aufwand für eine andere Sprache nutzbar gemacht werden kann. Die erste Lotus-1-2-3-Version entsprach dieser Vorstellung aber keineswegs, und die Übertragungsprobleme waren kaum zu bewältigen. Inzwischen hat man es jedoch geschafft. Symphony gibt es jetzt in einer französischen, einer deutschen und einer skandinavischen Version. Das Programm für Italien ist noch in Arbeit.

Die Schwierigkeit, den Text im Quellprogramm zu lokalisieren und mit dem vorhandenen Speicherplatz auszukommen, hat Lotus durch einen modularen Programmaufbau in den Griff bekommen. 1-2-3 besteht jetzt aus zwei Teilen. Der Textbereich wurde aus dem Quellprogramm herausgetrennt. Das Verfahren dieser Trennung wird „Localisation" genannt. Damit sind zwei Probleme gelöst: Durch die Speicherung des Textes in einem Extra-Segment kann von vornherein zusätzlicher Speicherplatz für unterschiedliche Wortlängen etc. reserviert werden. Außerdem kann der Text leichter vom Restprogramm isoliert werden. Ein Hilfsprogramm löst den Text aus dem Programm, danach kann er übersetzt und wieder in den Textbereich übertragen werden. Auch seine Anordnung läßt sich verändern. Damit kann man nun auch einer abweichenden nationalen Syntax gerecht werden.

Ein weiterer kritischer Punkt bei der Übersetzung des Textes ist, daß Symphony Befehle auch durch Eintippen des ersten Buchstabens eines Kommandos entgegennimmt. Daher muß jeder Befehl mit einem anderen Buchstaben beginnen. Und wenn die wörtliche Übersetzung zu lang und damit der Speicherplatz knapp wurde, mußten gelegentlich Kompromisse eingegangen werden, um die Bedeutung der Wörter zu erhalten.

Schwierigkeiten bereitet nach wie vor die Umsetzung eines nationalen Zeichensatzes. Ferner gibt es zahlreiche Varianten des Zeichencodes, die bei Übertragungen berücksichtigt werden müssen. Im Zeitalter des Lochstreifens war die 7-Bit-ASCII-Norm weltweit üblich. Seit die Acht-Bit-Computer mehr Möglichkeiten bieten, ist dieser Standard brüchig geworden. Jeder Hersteller „bastelt" heute an seiner eigenen Version des ASCII-„Standards".

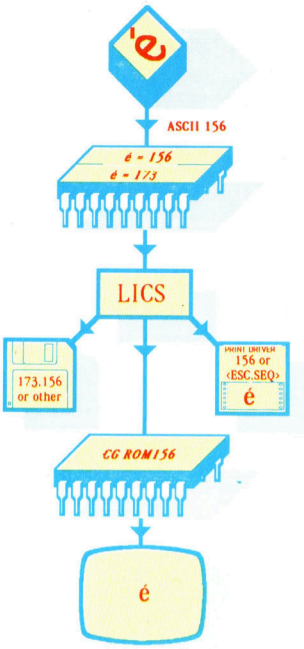
Übersetzungsprobleme

Ähnliche Entwicklungen gab es in den europäischen Staaten, wo vielfach unterschiedliche Tastaturanordnungen zur Norm erhoben worden sind. Damit wird Datenübertragung zwischen den nationalen Versionen des gleichen Computertyps oft zu einem enormen Problem. – Ein deutscher ASCII-Code ist längst nicht dasselbe wie ein spanischer!

Eine große Hürde bei der Übersetzung von Symphony entstand auch dadurch, daß viele Befehle mit der @-Taste ausgelöst wurden, die auf nicht-englischen Tastaturen fehlt.

Bei Lotus hat man dafür eine Lösung gefunden: Es wurde ein eigener Zeichensatz (LICS = Lotus International Character Set) entwickelt. In den insgesamt 250 Zeichen, die in jeder Symphony-Version zur Verfügung stehen, sind alle in Europa üblichen Buchstaben enthalten. Auch die Übertragung der Eingabe von einer nationalen Tastatur in LICS ist möglich. Erst dadurch kann die Eingabe vom Programm verstanden und genutzt werden. Mit der ALT-Taste und einer Zahl zwischen 0 und 9 können auch die Buchstaben erzeugt werden, die auf der Tastatur fehlen.

Die Übersetzung eines kommerziellen Programms ist nicht nur zeitaufwendig, sondern auch teuer. Ein Zeitaufwand von neun Monaten bei Kosten zwischen 30 000 Mark und 300 000 Mark ist kein Sonderfall. Dennoch lohnt sich der Mehraufwand – im kontinentalen Europa warten unzählige potentielle Kunden auf Software, die sie nicht mehr mit dem Lexikon in der Hand bedienen müssen.



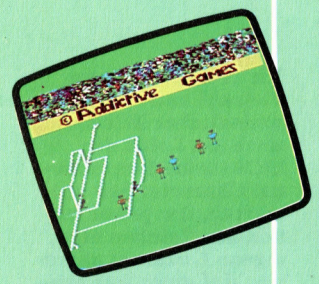
Zur Verarbeitung der Vielzahl national unterschiedlicher Zeichensätze hat Lotus den LICS (Lotus International Character Set) entwickelt, der jedem nationalen Zeichen eine eigene Codezahl zuordnet. Auf einer französischen Tastatur erzeugt etwa das é den Code 156, welcher der 173 im LICS entspricht. Vor der Übertragung auf den Bildschirm decodiert LICS ihn wieder zu 156. Falls der Drucker 156 nicht als Zeichen erkennt, kann das é auch durch eine Steuersequenz erzeugt werden, etwa <e>,<ESC>, <BSPACE>,<´>.

Fußballstrategen

Bei Strategiespielen wie dem „Football Manager" von Addictive Games Limited muß eine ganze Reihe von Faktoren beurteilt werden, und das gewünschte Ergebnis läßt sich nur durch sorgfältige Planung und ein bißchen Glück erreichen.

Der in den frühen achtziger Jahren auf dem Markt erschienene Football Manager ist erstaunlich langlebig. In einem Markt, in dem sich auch die erfolgreichsten Renner nur für begrenzte Zeit verkaufen lassen, nimmt dieses Spiel dadurch eine Sonderrolle ein. Das Konzept ist einfach. Sie sind der Kapitän einer Fußballmannschaft, die Sie in der Fußballiga zum Erfolg führen müssen.

Das Spiel beginnt natürlich am Anfang einer Saison. Auf einem Menü erscheinen Ihre Möglichkeiten: Sie können Ihre Tabellenposition abfragen, den Veranstaltungskalender ansehen oder die Eigenschaften der Spieler abwägen, die für Ihr Team in Frage kommen.



Das Bild zeigt eine Spielszene des Football Managers auf dem Commodore 64. An diesem Punkt hat der Spieler keine Kontrolle über die Abläufe auf dem Spielfeld, sondern kann nur die „Höhepunkte" beobachten. Dies ist vielleicht der aufreibendste Teil des Spiels, da Sie nur hoffen können, daß Sie die Situation vorher richtig beurteilt haben.

Es gibt drei Kategorien von Spielern: Verteidiger, Mittelfeldspieler und Stürmer. Ein umsichtiger Manager wird natürlich ein wohlausgewogenes Team ins Spiel schicken. Die Aufstellung kann vor jedem Treffen geändert werden. Da Sie in den frühen Spielstadien nur zwölf Spieler zur Verfügung haben, gibt es am Anfang noch nicht viel zu entscheiden. Beim Fortschreiten des Spiels kommen jedoch zusätzliche Spieler auf den Markt, die Sie „einkaufen" können.

Die Spieler haben drei grundlegende Eigenschaften, die beim Preisangebot berücksichtigt werden müssen. Jeder besitzt einen „Geschicklichkeitsfaktor", der mit einer Skala von eins bis fünf gemessen wird. Die Spieler haben außerdem einen „Wert", der sich zwar auf ihren Geschicklichkeitsfaktor bezieht (5000 für jeden Geschicklichkeitspunkt), sich im Verlauf des Bietens aber ändern kann. Schließlich gibt es noch die „Energie" der Spieler, die anhand einer Skala von eins bis zwanzig bewertet und nach jedem Spieleinsatz um eins verringert wird. Es ist daher nicht ratsam, bei jeder Austragung nur die besten Spieler einzusetzen. Durch geschickte Rotation erhalten diese Ruheperioden, in denen sie sich wieder mit Energie „aufladen" können.

Am Anfang jeder „Runde" können Sie für einen Spieler bieten. Wird Ihr Gebot angenommen, gehört der Spieler zu Ihrem Team. Ihr Gebot kann aber auch zurückgewiesen werden, obwohl Sie mehr als den angegebenen Wert zahlen möchten. Bei einem weiteren Gebot ist der Wert des gleichen Spielers dann gestiegen – nur weil Sie Ihr Interesse an ihm verstärkt gezeigt haben.

Nachdem Ihre Transaktionen auf dem Markt abgeschlossen sind, Sie – wenn nötig – Kredite arrangiert und Ihre augenblickliche Tabellenposition angesehen haben, können Sie ein Spiel austragen. Der Computer zeigt die relativen Stärken Ihres und des gegnerischen Teams an und gibt Ihnen die Möglichkeit, erschöpfte oder verwundete Spieler gegen frische auszutauschen. Danach zeigt der Computer „ausgewählte Höhepunkte" des Spiels.

Diese bestehen hauptsächlich aus Torkämpfen an beiden Enden des Spielfelds – die Dauer der Zeit, die an dem einen oder dem anderen Tor zugebracht wird, hängt von der relativen Stärke des Mittelfeld ab, während der Ausbau des gegnerischen Sturms und der Verteidigung bestimmen, ob Tore geschossen werden. Obwohl das Spiel durch die Grafik recht aufregend gestaltet wird, ist die Darstellung doch recht primitiv.

Am Ende des Spieles werden die Tore zusammen mit allen Ergebnissen dieser Liga angezeigt. Ihr Team steigt in der Tabelle entsprechend auf oder ab.

In diesem Spiel muß der Manager nicht nur die beste Strategie für das nächste Treffen entwerfen, sondern auch für zukünftige Austragungen planen und beispielsweise seine besten Spieler für die End- und Pokalspiele zurückhalten. Er muß entscheiden, ob seine besten Spieler ruhen (und Punkte verlieren) oder spielen (und Verletzungen und Energieverluste riskieren, die ihren Einsatz in wichtigen Spielen behindern könnten).

Obwohl das Spiel keine schnellen „Arcadeläufe" enthält, wird schnell deutlich, warum es so beliebt ist. Trotz einiger Einschränkungen ist die Simulation realistisch genug, um den Spieler wirklich in die Position eines Fußball-Managers zu versetzen. Wenn dann schließlich die Meisterschaft gewonnen ist, stellt sich ein echtes Erfolgserlebnis ein.

Football Manager: Für den ZX81, Sinclair Spectrum, Commodore 64, VC 20, Acorn B, Electron, Schneider und Dragon
Herausgeber: Addictive Games Ltd., 7A Richmond Hill, Bournemouth, Dorset BH2 6HE, Großbritannien
Autor: Kevin Toms
Format: Cassette
Joysticks: Nicht erforderlich

Der Unvollendete: Sanyo MBC-550

Der Sanyo MBC-550, eine 16-Bit-Maschine mit 8088-Prozessor, ist eine preisgünstige Alternative zu dem IBM PC. Die Hardwaregrenzen und das Fehlen von Software schränken die Einsatzmöglichkeiten dieses ansonsten guten Computers jedoch ein.

Trotz fallender Preise für 16-Bit-Prozessoren sind Computer mit 16-Bit-Technik (abgesehen vom Sinclair QL) immer noch sehr teuer. Sinclair Research hat gezeigt, daß eine 16-Bit-Maschine durchaus preisgünstig angeboten werden kann. Vergleichbare Rechner kosten meist um die 4000 Mark. Der Grund für die hohen Verkaufspreise ist vermutlich auch darin zu finden, daß 16-Bit-Maschinen hauptsächlich an Geschäftsleute verkauft werden, die mehr Geld zur Verfügung haben. Dies trifft besonders für Computer zu, die auf dem Intel-8088-Prozessor aufbauen und an die Verkaufsstrategie des Marktführers IBM gebunden sind.

Da der IBM PC mit einer Diskettenstation immer noch etwa 8000 Mark kostet, rechnen sich die meisten Hersteller vergleichbarer Geräte aus, daß ihre Kunden eine Preissenkung von mehreren hundert Mark schon als Gelegenheit ansehen werden. Den Interessenten, denen auch dieser Preis noch zu hoch war, blieb bisher nichts anderes übrig, als ein Gerät mit der Z80-Technik zu kaufen, das mit Disketten und dem Betriebssystem CP/M arbeitete. Inzwischen haben jedoch viele Hersteller dieses Käuferpotential erkannt und bieten preisgünstige 8088-Maschinen an.

Der Sanyo MBC-550 war eins der ersten Geräte für diesen Markt. Er ist nicht mit dem IBM PC kompatibel, arbeitet jedoch auch mit dem Betriebssystem MS-DOS. Der Rechner wird wahlweise mit einem oder zwei Diskettenlaufwerken geliefert.

Dreiblock-Tastatur

Wie die meisten kommerziellen Geräte besteht auch der MBC-550 aus zwei getrennten Elementen – der Tastatur und dem eigentlichen Computer. Beide befinden sich in robusten silbermetallfarbenen Gehäusen aus Metall und Plastik. Die Tastatur ist in drei Blöcke unterteilt. Die programmierbaren Funktionstasten auf der linken Seite können in Verbindung mit der Shift-Taste zehn Funktionen aufrufen, die sich entsprechend der aktuellen Anwendung definieren lassen. In BASIC sind diese Tasten mit häufig eingesetzten Befehlen wie LIST, LOAD und RUN belegt.

Der Sanyo MBC-550 ist eine preisgünstige 16-Bit-Maschine, die unter MS-DOS läuft und sich für den kommerziellen Einsatz eignet. Es gibt zwei Geräteversionen: Der MBC-550 ist mit einem Diskettenlaufwerk ausgerüstet, während der MBC-555 über zwei Stationen verfügt. Der im Bild gezeigte Monitor ist im Grundpreis nicht mit inbegriffen.

Die Schreibmaschinentastatur ist nach dem QWERTY-Standard ausgelegt. Die Return-Taste hat die vierfache Größe der übrigen Tasten. Rechts der Space-Taste befinden sich die Grafiktaste, mit der der Grafikzeichen-Modus aktiviert wird. Auf der rechten Seite der Tastatur liegt der Zehnerblock, der sich auch als Taschenrechner einsetzen läßt oder – über die Number-Lock-Taste – als Cursorsteuerung für den Bildschirmeditor in BASIC.

Der Computer selbst befindet sich in einem großen flachen Kasten mit den Ausmaßen eines Videorecorders, auf dem der (im Kaufpreis nicht enthaltene) Sanyo-Monitor Platz findet. Wie die Tastatur besteht auch das Monitorgehäuse fast vollständig aus Metall. Auf der Frontplatte des Computers befinden sich der Netzschalter und eine Aussparung für zwei Diskettenlaufwerke. Die Diskettenstationen des MBC-550 arbeiten mit Standarddisketten im 5¼-Zoll-Format.

Auf der Rückseite des Computers befinden

sich die Schnittstellen und das Netzteil. Links ist das Netzkabel zu sehen, in der Mitte die parallele Druckerschnittstelle im Centronicsformat, rechts eine RGB-Buchse für den Anschluß eines Farbmonitors und eine Composite-Video-Verbindung, die sich für den monochromen Sanyo-Bildschirm eignet. Darüber liegen die Aussparungen für den Einbau zusätzlicher Peripherieschnittstellen.

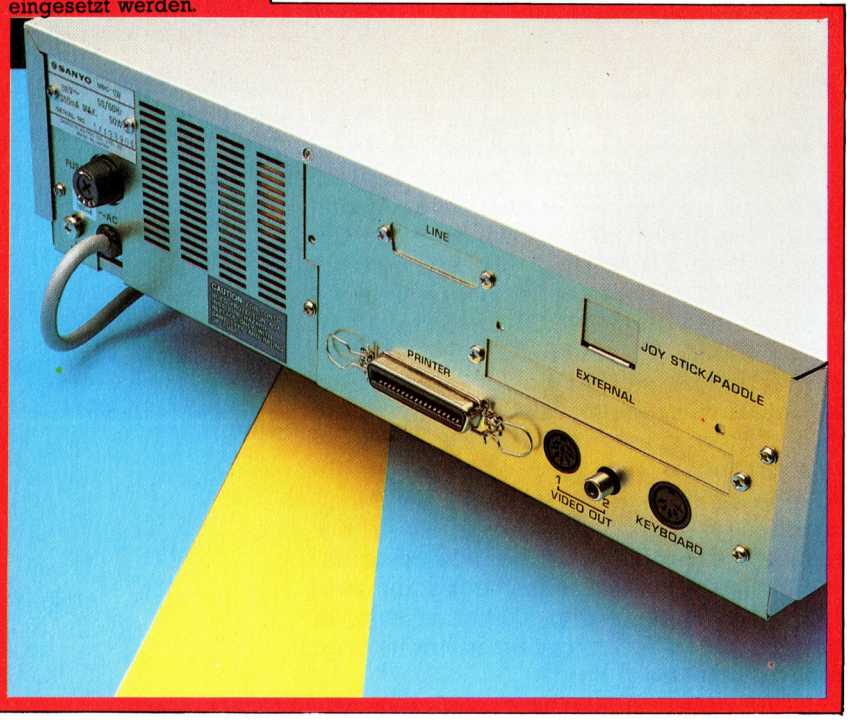
Der Line-Ausgang über dem Druckerinterface ist für eine serielle RS232C-Schnittstelle gedacht, die die Kommunikation mit anderen Computern (per Modem) oder den Anschluß von Peripheriegeräten (wie seriellen Druckern) ermöglichen soll. Rechts daneben befindet sich eine weitere Gehäuseöffnung für ein Joystick-Interface. Es sollen sich jedoch nach dem Einbau entsprechender Schnittstellen auch Trackballs, Paddles und andere Steuergeräte dieser Art anschließen lassen. Über die Erweiterungsleiste können – nach entsprechender Umrüstung des Geräts – weitere Peripheriegeräte betrieben werden.

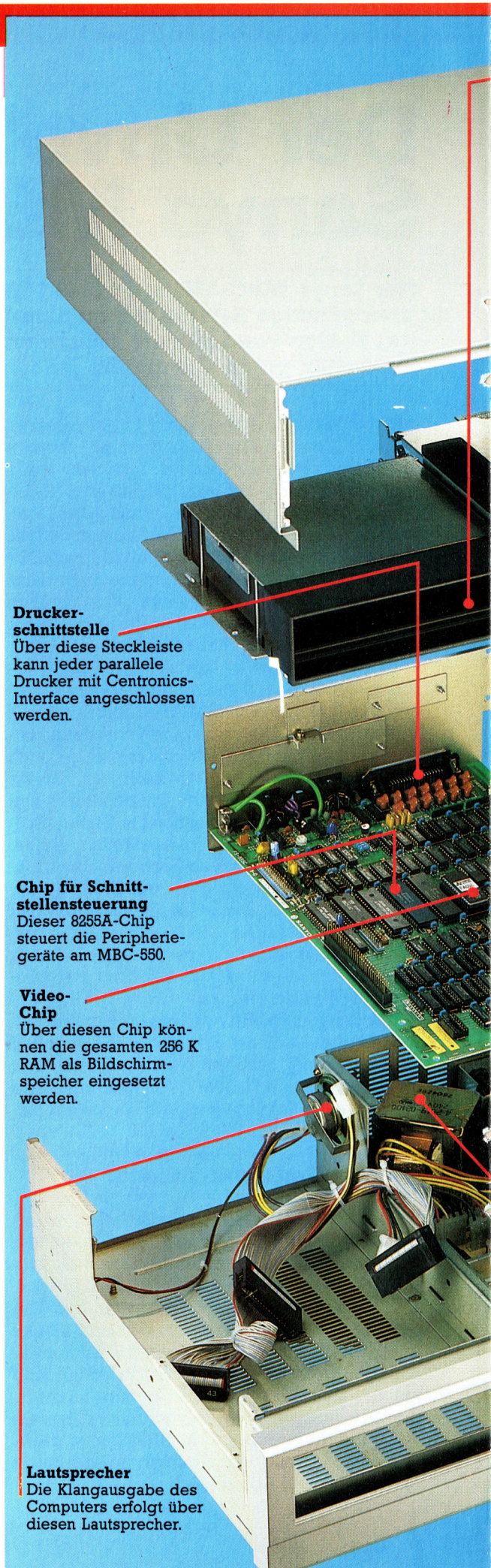
Gute Dokumentation

Das Sanyo-Handbuch enthält genaue Beschreibungen, wie man selbst die Peripherieschnittstellen, zusätzliche RAMs und das zweite Diskettenlaufwerk einbauen kann. Allerdings rät das Handbuch, doch lieber einen Fachmann hinzuzuziehen, wenn beim Einbau Fragen auftauchen. Die Dokumentation enthält ferner eine Übersicht über Computerbegriffe, eine vollständige Einführung in die Befehle des Sanyo-BASIC – darunter Instruktionen, wie sich das Diskettensystem vom BASIC aus steuern läßt – und viele technische Informationen.

Sanyo-BASIC – eine Variante des Microsoft-

Erweiterung möglich
Wenn der MBC-550 auch nicht die Vielzahl von Erweiterungsmöglichkeiten besitzt, so hat Sanyo doch schon zukünftige Erweiterungen berücksichtigt. Das Handbuch enthält genaue Anweisungen, wie Schnittstellen für Joysticks und serielle Peripheriegeräte problemlos eingesetzt werden.



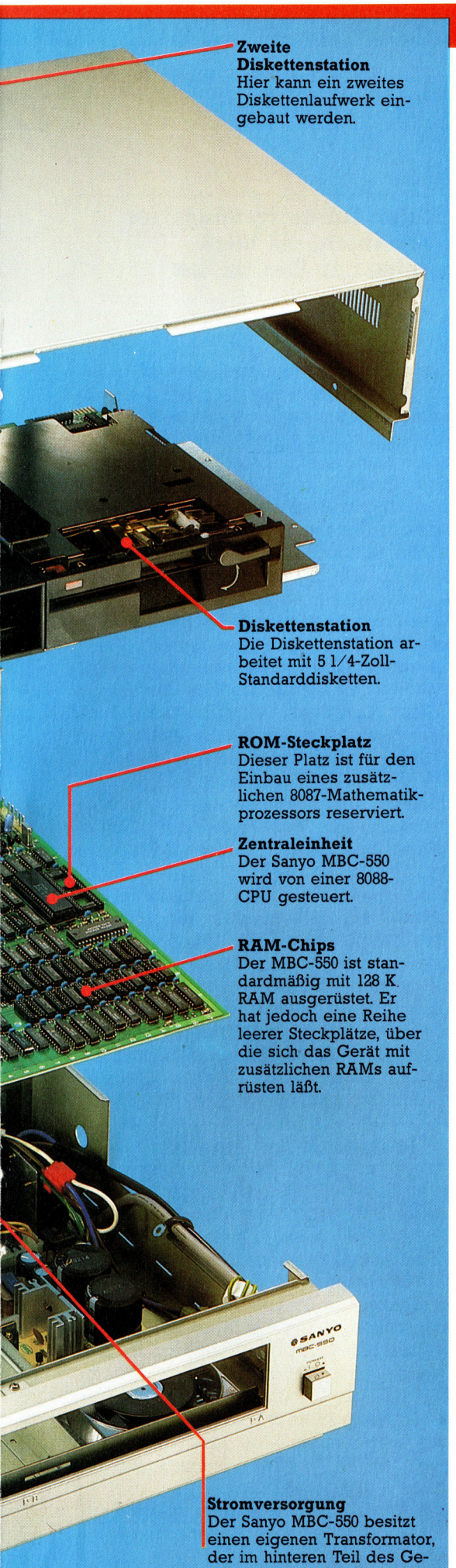


Drucker-schnittstelle
Über diese Steckleiste kann jeder parallele Drucker mit Centronics-Interface angeschlossen werden.

Chip für Schnittstellensteuerung
Dieser 8255A-Chip steuert die Peripheriegeräte am MBC-550.

Video-Chip
Über diesen Chip können die gesamten 256 K RAM als Bildschirmspeicher eingesetzt werden.

Lautsprecher
Die Klangausgabe des Computers erfolgt über diesen Lautsprecher.



BASIC – bietet viele Vorzüge, ist bei der Verarbeitung mathematischer Funktionen jedoch etwas langsam. Zwar ist schon seit längerem bekannt, daß der 8088-Chip auf mathematische Aufgaben recht träge reagiert (daher wird auch bei vielen Computern der Mathematikprozessor 8087 zusätzlich eingebaut), doch der MBC scheint sogar noch langsamer zu sein als andere 8088-Maschinen. Auch für das Zeichnen von Linien braucht der MBC-550 viel Zeit, da das Sanyo-BASIC keinen DRAW-Befehl besitzt. Alle Linien müssen mit PSET gezeichnet werden, wobei (langsame) Programmschleifen die einzelnen Pixel auf die gewünschte Farbe setzen.

Abgesehen von den Microsoft-ähnlichen Befehlen wir MID$, LLIST und CIRCLE kann auch mit Bildschirmfenstern gearbeitet werden. WHILE. . .WEND fördert die strukturierte Programmierung, und viele Schlüsselwörter des Sanyo-BASIC können außerdem in anderer Form mit zwei oder drei Tasten und der Control-Taste eingegeben werden. Der DIM-Befehl wird beispielsweise auch durch das gleichzeitige Drücken von CTRL, SHIFT und D abgerufen, während bei CTRL und P das Wort PRINT erscheint. Doch obwohl jede Vereinfachung der BASIC-Programmierung ausdrücklich zu begrüßen ist und viele Tastenfolgen sich eng an die Schlüsselwörter anlehnen, ist es nur schwer vorstellbar, daß alle 40 BASIC-Befehle auf diese Weise eingesetzt werden. Vermutlich wird dieses Eingabesystem in der Praxis nur für wenige oft verwandte Befehle angewendet werden.

Zum Lieferumfang des MBC-550 gehört auch eine Systemdiskette mit dem MS-DOS-Betriebssystem, dem Sanyo-BASIC, der weitverbreiteten Textverarbeitung WordStar und dem Kalkulationsprogramm CalcStar. Ein Handbuch liefert Beschreibungen all dieser Systeme.

Nicht IBM-kompatibel

Da der MBC-550 auf dem 8088-Chip aufbaut und mit MS-DOS arbeitet, wäre eigentlich zu erwarten, daß auch die breite Palette der IBM-PC-Software darauf funktioniert. Aufgrund der unterschiedlichen Hardwarespezifikationen lief jedoch keins der IBM-kompatiblen Pakete, die wir darauf testeten.

Auf dem Markt für preisgünstige MS-DOS-Maschinen war der Sanyo MBC-550 der erste Versuch, die Barriere von 5000 Mark zu durchbrechen. Seine Inkompatibilität mit der Software des IBM PC stellt das Gerät jedoch ins Abseits. Für einen Geschäftsmann, der nur eine Textverarbeitung und ein Kalkulationssystem braucht, hat der MBC-550 durchaus einen gewissen Wert. Wer aber Zugang zu einer größeren Softwarebasis haben will, sollte etwas mehr ausgeben oder auf IBM-PC-kompatible Geräte zurückgreifen.

Sanyo MBC-550

ABMESSUNGEN
375 × 355 × 108 mm

ZENTRALEINHEIT
Intel 8088 mit 3,6 MHz

BILDSCHIRMDARSTELLUNG
25 Zeilen mit je 80 Zeichen; 640 × 200 Pixel in der hohen grafischen Auflösung (acht Farben).

SCHNITTSTELLEN
Centronics parallel; eine RS232-Schnittstelle und eine Joystick-Buchse können nachgerüstet werden.

PROGRAMMIERSPRACHEN
Sanyo-BASIC

TASTATUR
65 Schreibmaschinentasten, fünf programmierbare Doppelfunktionstasten, Zahlenblock mit 19 Tasten.

DOKUMENTATION
Das Handbuch ist ausgezeichnet und enthält eine genaue Beschreibung der BASIC-Befehle. Leider fehlt eine ausführliche BASIC-Einführung. Vorhanden ist eine Einführung in MS-DOS, WordStar und CalcStar. Recht ungewöhnlich für eine kommerzielle Maschine sind die vielen technischen Informationen – darunter Anleitungen, wie der Anwender seine eigenen Erweiterungsplatinen einbauen kann.

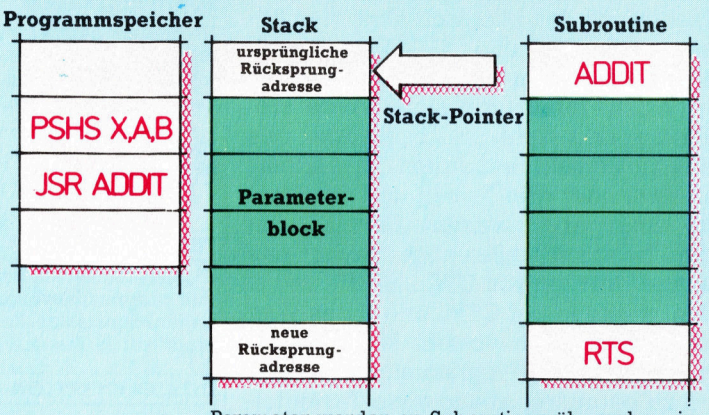
STÄRKEN
Der Sanyo MBC-550 bietet die 16-Bit-Technik für einen günstigen Preis.

SCHWÄCHEN
Da das Gerät nicht mit bestehender Software kompatibel ist, entsteht das Problem aller neuen Maschinen. – Es fehlen Programme.

Tiefstapeln

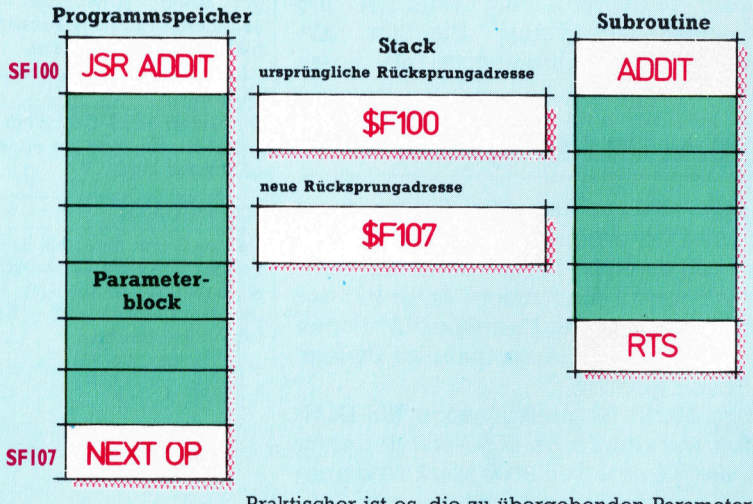
Zwar haben wir uns schon ausführlich mit den Adressierungsarten des 6809 beschäftigt, doch gibt es einige Varianten, die noch untersucht werden müssen, wie beispielsweise der Einsatz des Befehlszählers als Index.

Parameter übergeben



Parameter werden an Subroutinen übergeben, indem sie in Register geladen und dann auf den Stack geschoben werden. Die Subroutine zieht sie wieder vom Stack herunter. Dabei ist zu beachten, daß die Rücksprungadresse nach dem Laden an die richtige Stelle des Stacks gesetzt wird. Andernfalls verursacht der Stack einen Overflow.

Parameter einsetzen



Praktischer ist es, die zu übergebenden Parameter direkt hinter dem Aufruf (JSR) der Subroutine in das Programm einzusetzen. Die Subroutine kann nun die auf dem Stack gespeicherte Rücksprungadresse als Basisadresse des Parameterblocks einsetzen und mit der indizierten Adressierung darauf zugreifen. Vor dem RTS-Befehl muß die Rücksprungadresse verändert werden, damit sie auf den nächsten auszuführenden Befehl zeigt und nicht auf den Anfang des Parameterblocks.

Bisher wurden die beiden Register S und U (die Stack-Pointer) nur als zusätzliche Indexregister eingesetzt, und der sogenannte „Hardwarestack" zur Speicherung der Rücksprungadressen aus Subroutinen wurde nur nebenbei erwähnt. In diesem Artikel gehen wir daher zunächst genauer auf die Arbeitsweise des Stacks ein.

Der Stack läßt sich als eine spezielle „Liste" beschreiben. Wie Sie bereits wissen, besteht eine Liste aus einer Folge von Datenelementen. Diese Folge kann nach bestimmten Eigenschaften der Daten geordnet werden (eine Zahlenserie in numerischer Reihenfolge, eine Zeichenkette in alphabetischer Sortierung etc.). Eine Anordnung entsteht aber auch durch die Reihenfolge, in der Datenelemente an die Liste angefügt werden. Bei all diesen Listen hat der Inhalt des folgenden und des vorhergehenden Datenelements große Bedeutung, insbesondere das erste (der „Listenkopf") und das letzte Element (der „Abschluß").

Wichtig ist ebenfalls die „dynamische Datenstruktur" einer Liste. In allgemeingültig definierten Listen können Datenelemente beliebig eingefügt oder herausgenommen werden. Eine Liste wird zu einem „Stack" (Stapel), wenn Daten nur an einem Ende herausgenommen oder angefügt werden können. Jedes neu hinzukommende Datenelement wird zum Listenkopf und kann auch nur als erstes Element wieder herausgenommen werden.

Gestapelte Information

Die Bezeichnung Stack gibt schon Auskunft über seine Funktionsweise. Stellen Sie sich einen Tellerstapel in einer Kantine vor: Die Teller werden immer nur vom oberen Ende des Stapels genommen, und neue Teller werden auf die Spitze des Stapels gesetzt. Es wäre zwar möglich, auch aus der Mitte des Stapels Teller herauszunehmen oder dort einzusetzen, doch würde der Vorgang dadurch unnötig kompliziert. Es ist jedoch möglich, jedes Element des Stapels zu überprüfen. Das Hinzufügen und Wegnehmen wird „auf den Stack schieben" (PUSH) bzw. „herunterziehen" (PULL oder POP) genannt.

Bei der Arbeit eines Stacks können zwei extreme Situationen eintreten: Erstens, der Stack

ist leer (wenn der nächste Stack-Befehl Daten anfügt, entsteht kein Problem, wenn nicht, gibt es Schwierigkeiten), und zweitens, der Stack fließt über. In unserer Analogie bedeutet dies: Wenn der Tellerstapel die Decke erreicht, läßt sich kein Teller mehr daraufsetzen. Diese beiden Extremsituationen heißen „Underflow" (Unterfluß) und „Overflow" (Überfluß).

Stacks können auf vielerlei Weise eingesetzt werden (in BASIC beispielsweise als Arrays). Für unsere Zwecke benötigen wir einen frei verfügbaren Speicherblock und ein Register als „Stack-Pointer". Dieser Pointer enthält die Adresse des Listenkopfes. Der Speicherstack kann jedoch nicht von außen untersucht werden, da sich eine Speicheradresse mit Stack-Element nicht von Speicherstellen mit anderen Daten unterscheidet. Wichtig ist auch, daß Daten nicht wirklich aus dem Speicher in ein Register „geladen", sondern nur kopiert werden. Sie werden nicht wirklich vom Stack „gezogen", sondern es wird nur der Pointer auf eine andere Adresse gesetzt.

Wie ein Verschiebebahnhof

Im Gegensatz zu unserem Tellerstapel „wächst" ein Speicher-Stack des 6809-Systems abwärts. Je mehr Elemente auf den Stack geschoben werden, desto niedriger wird die Adresse des Stack-Pointers – er „wächst gegen Null".

Zwei Befehle des 6809 lösen Stack-Vorgänge aus: PSH schiebt Daten auf den Stack, und PUL zieht sie wieder herunter. Da für diese Vorgänge die beiden Pointer S und U eingesetzt werden können, stehen die Befehle PSHS, PULS, PSHU und PULU zur Verfügung.

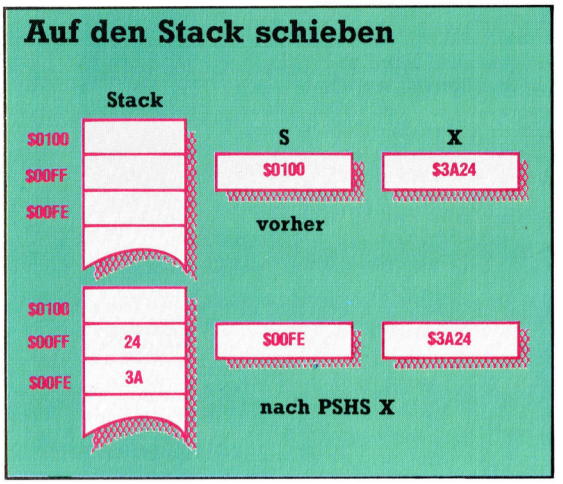
Um die Adresse des nächsten freien Stack-Speichers zu erhalten, dekrementiert der Befehl PSHS X zuerst den Stack-Pointer S um zwei (oder eins, wenn ein Acht-Bit-Register gespeichert wird) und legt dann den Inhalt von X auf dieser Adresse ab. Die erste Abbildung veranschaulicht diesen Vorgang. Beachten Sie, daß der 6809 das Adreßformat hi-lo einsetzt. Das höherwertige Byte ($3A) von X ist in $00FE gespeichert, also in einer niedrigeren Speicheradresse als das in $00FF untergebrachte niederwertige Byte ($24). Wenn Sie mit einem Assembler arbeiten, ist die In- oder Dekrementierung des Stack-Pointers nicht wichtig, da das Programm alle notwendigen Speichervorgänge berücksichtigt.

Der Befehl PULS X bewirkt das genaue Gegenteil: Der 16-Bit-Wert der in S gespeicherten Adresse wird in X geladen und der Inhalt von S dann um zwei inkrementiert. Das zweite Diagramm zeigt diese Veränderungen.

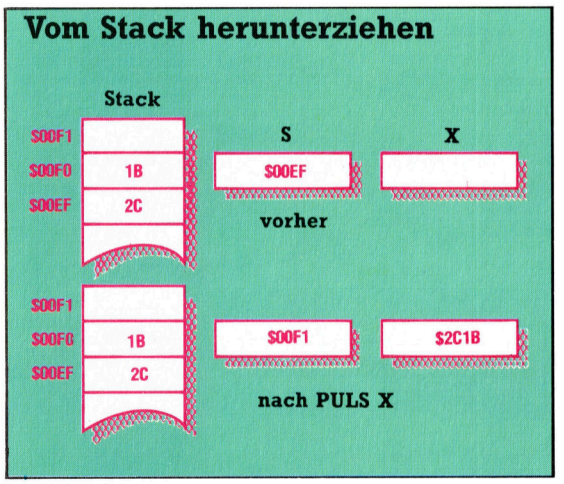
Es lassen sich auch mehrere Register gleichzeitig heraufschieben oder herunterziehen, beispielsweise:

PSHS X,Y,U,A

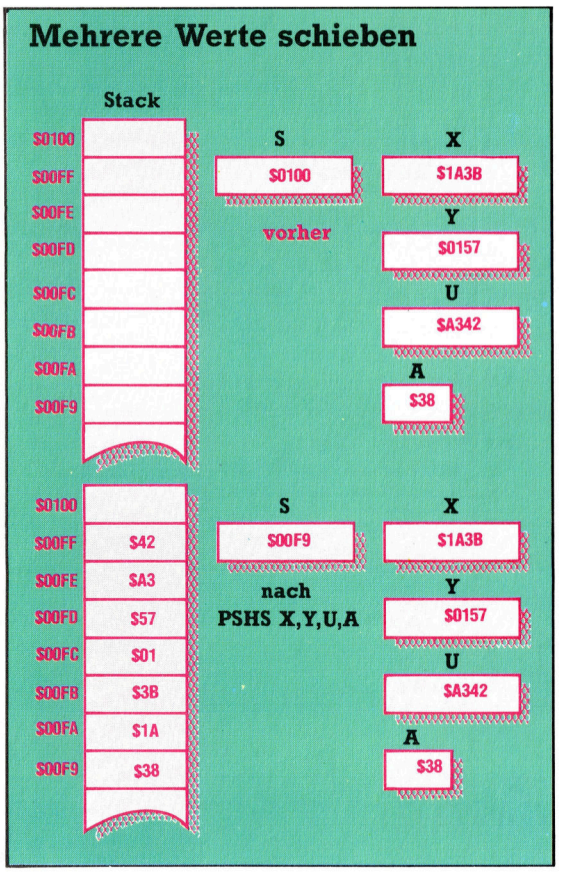
Die Anordnung der Register hat in diesem Fall



Der Stack-Pointer des 6809 adressiert immer die „Spitze" des Stacks – das heißt das Byte, in das zuletzt geschrieben wurde. Bei der Ausführung von PSHS X wird daher S um zwei dekrementiert und zeigt so auf die neue Stackspitze. Der Inhalt von X wird dann im Format hi-lo auf dieser Adresse gespeichert. Beachten Sie, daß der Stack „gegen Null wächst", das heißt, der Stack-Pointer zeigt beim Wachsen des Stacks auf immer niedrigere Adressen.



Bei der Ausführung von PULS X wird der Inhalt der zwei Bytes beim aktuellen Stack-Pointer nach X kopiert. Danach wird S um zwei dekrementiert, damit der Pointer auf die neue Stack-Spitze zeigt.



Bei einem Stack-Befehl, der mehrere Werte gleichzeitig umfaßt, werden die angesprochenen Register immer in der gleichen vorbestimmten Folge abgelegt: PC, U oder S, Y, X, DP, B, A, CC. Wenn daher PSHS X,Y,U,A ausgeführt wird, wird zuerst U gespeichert, dann Y, X und A.

Addieren mit mehrfacher Präzision

Die beiden Programmteile zeigen zwei Möglichkeiten, wie sich mit Stacks Additionen mit mehrfacher Präzision ausführen lassen. Das erste Beispiel speichert seine Parameter gleich hinter dem Subroutinenaufruf. Dabei werden die beiden Zahlen (in $100 und $104) addiert und das Ergebnis in $108 gespeichert:

```
        BSR     MPADD
        FCB     4           Länge jeder Zahl (in Bytes)
        FDB     $100        Adresse der ersten Zahl
        FDB     $104        Adresse der zweiten Zahl
        FDB     $108        Adresse des Ergebnisses
        ORG     $1000
MPADD   PSHS    X,Y,U,A,B,CC  Alle Register sichern. Dieser Befehl schiebt neun Bytes auf den Stack
        LDU     9,S         Dieser Befehl lädt die mit BSR auf den Stack geschobene PC-Rücksprungadresse in U
        PULU    X,Y,B       Die hinter dem Subroutinenaufruf gespeicherten Daten werden wie ein Stack behandelt, die Adressen der zwei Zahlen in X und Y und die Länge in B gespeichert
        LDU     ,U          Die Adresse des Ergebnisses wird in U abgelegt
        ANDCC   #%11111110  Übertragsflag löschen
LOOP    LDA     ,X+         Ein Byte auf das andere addieren (mit Übertrag)
        ADCA    ,Y+
        STA     ,U+         Ergebnis speichern
        DECB                Prüfen, ob Addition beendet
        BNE     LOOP        Falls nicht, LOOP
        LDU     9,S         Sonst Rücksprungadresse holen
        LEAU    7,U         und sieben addieren, damit die
        STU     9,S         sieben Parameterbytes übersprungen werden
        PULS    PC,U,X,Y,A,B,CC  Alle Register wieder in ihren ursprünglichen Zustand versetzen und statt eines RTS-Befehls den PC setzen
```

Das zweite Beispiel führt den gleichen Vorgang aus, schiebt die Parameter jedoch auf den Stack. Die aufrufende Routine sieht folgendermaßen aus:

```
        LDU     #$108       Adresse des Ergebnisses
        LDX     #$100       Adresse des Operanden
        LDY     #$104
        LDA     #4          Länge
        PSHS    U,X,Y,A     Parameter auf den Stack schieben
        BSR     MPADD
        ORG     $1000
MPADD   PSHS    X,Y,U,A,B,CC  Alle Register sichern
        LEAU    11,U        Das Sichern der Register belegt neun Bytes und die Rücksprungadresse zwei Bytes – insgesamt elf. U zeigt nun auf die Parameter
        PULU    X,Y,B
        ANDCC   #%11111110
LOOP    LDA     ,X+         Gleicher Ablauf wie im ersten Beispiel
        ADCA    ,Y+
        STA     ,U+
        DECB
        BNE     LOOP
        PULS    PC,U,X,Y,A,B,CC
```

keine Bedeutung, da immer in der gleichen Reihenfolge gespeichert wird: PC (der Befehlszähler), U oder S, Y, X, DP (das Seitenregister für die direkte Adressierung), B, A und CC (das Condition Code Register). Das Herunterziehen geschieht natürlich in umgekehrter Reihenfolge. Lediglich der eigene Stack-Pointer S oder U kann nicht auf den zugehörigen Stack geschoben werden.

Stacks werden in der Programmierung als schnelle Zwischenspeicher eingesetzt. Ihre größte Bedeutung haben sie jedoch bei der Behandlung von Interrupts (auf die wir später genauer eingehen) und Subroutinen. Wir haben bereits gezeigt, wie der Aufruf einer Subroutine den Inhalt des Befehlszählers automatisch auf den Stack schiebt und beim Rücksprung wieder herunterzieht (RTS entspricht PULS PC). Mit beiden Stacks – besonders aber mit S – lassen sich Parameter an Subroutinen übergeben.

Ohne Komplikationen

Diese Aufgabe haben wir bisher zwar mit Registern erledigt, doch hat diese Technik zwei schwache Punkte. Erstens kann der Fall eintreten, daß mehr Parameter übergeben werden müssen, als Register zur Verfügung stehen, und zweitens kann es Komplikationen geben, wenn die aufgerufene Routine ein Register mit einem Parameter verwendet, der noch gebraucht wird. Für die Parameterübergabe werden häufig die beiden folgenden Techniken eingesetzt:

1) Die Daten werden unmittelbar nach Aufruf der Subroutine mit FCB-, FDB- oder FCC-Anweisungen in der Mitte des Programms gespeichert. Der Wert des Befehlszählers, den der JSR-Befehl auf den Stack geschoben hat, enthält die Adresse der ersten Speicherstelle (der Befehlszähler zeigt immer auf das Byte, das der aktuellen Instruktion folgt). Die Daten lassen sich nun mit geeigneten Offsets lesen. Unser erstes Programmbeispiel demonstriert diese Technik. Dabei muß jedoch beachtet werden, daß RTS die Steuerung an den nächsten gültigen Befehl übergibt und nicht an ein Datenelement.

2) Die Daten werden in Register geladen und vor Aufruf der Subroutine auf den Stack geschoben. Die Subroutine zieht die Daten wieder herunter und verarbeitet sie. Dabei ist zu beachten, daß der Stack-Pointer beim Rücksprung (RTS) wieder auf den zuvor gespeicherten Wert des Befehlszählers zeigt.

Bei beiden Methoden werden S und U als Indexregister und auch als Stack-Pointer eingesetzt. Dadurch kann mit der indizierten Adressierung leicht auf die Stacks zugegriffen werden, und die Stack-Daten lassen sich auch leicht wieder vom Stack herunterziehen. Ferner ist sichergestellt, daß der Stack beim Rücksprung die korrekten Daten enthält.

Gut geplant ist halb getan

In der vorangegangenen Folge beschäftigten wir uns mit MacProject, das die CPA-Prinzipien für das langfristige Management von Projekten verwendet. In dieser Folge stellen wir eine ähnliche Applikation für MS-DOS-Rechner wie den IBM und kompatible vor.

Wenngleich Microsofts „Project"-Programm nicht über die Grafikfähigkeiten von „MacProject" oder dem Digital-Research-Programm „Graphics Environment Manager" verfügt, nutzt es doch die Grafikmöglichkeiten des IBM optimal. So erzeugt beispielsweise die Kombination von Gleichheitszeichen oder Strichlinien mit „größer als"-Symbolen folgende nützlichen Programmhilfen:

=======> ----------->

Der erste Pfeil kann dazu verwendet werden, kritische Pfade aufzuzeigen, wogegen der zweite die anderen Abhängigkeiten innerhalb des Projekts aufzeigt. Für die Anwendung sind allerdings die folgenden Fragen interessanter:

- Wie flexibel ist das Programm, und wie weit reagiert es auf Veränderungen, die sich im Zuge der Projektentwicklung ergeben?
- Wie deutlich stellt es Verknüpfungen zwischen den Aktivitäten dar, und ist es eine wirkliche Hilfe bei der Projektverwaltung?
- Kann man das Programm von Anfang an für die Planung verwenden, oder muß der Benutzer erst alle Ideen vollständig entwickeln?

Um die letzte Frage zuerst zu beantworten: Der Benutzer sollte, bevor er mit seinen Computeraktivitäten beginnt, tatsächlich erst das Projekt zu Papier bringen. Dazu stellt das Handbuch fest:

„Bevor Sie beginnen, mit ‚Micro Soft Project' zu arbeiten, sind folgende Informationen für die Aktivitäten erforderlich:

Worum geht es? (Beschreibung)
Wie lange dauert es? (Dauer)
Was muß vorher getan werden? (Basis)
Wann beginnt es? (Starttermin)."

Um die Wichtigkeit dieser Vorbereitungen zu verdeutlichen, enthält das Handbuch einen zehnseitigen Anhang „Aufgabendefinition durch Arbeits-Strukturierung", in dem unter anderem folgendes enthalten ist:

„Die erfolgreiche Projektverwaltung fängt bei der Planung an. Zunächst sind Ziele, Zweck, Inhalte und Spezifikationen des Projekts klar darzulegen. Dann müssen die spezifischen Aktivitäten definiert werden."

Wohl kaum jemand, der mit Management-Techniken vertraut ist, wird die Notwendigkeit angemessener Vorbereitung in Frage stellen. Wer aber den Computer als Erweiterung des Anwenderwissens sieht, könnte annehmen, daß die Aufgabe der Software darin besteht, unangemessene oder ungenau definierte Aktivitäten zu vermeiden. MicroSoft Project hilft dem Anwender jedoch, die Prioritäten richtig zu setzen. Es bietet dem Benutzer zusätzlich eine Fülle von Änderungsmöglichkeiten.

Nach dem Einschalten des Computers bestimmen das tatsächliche Datum und die Zeit – als Teil der Einschaltsequenz gesetzt – automatisch den im Programm verwendeten Kalender. Er läßt sich nach Eingabe mehrerer Aktivitäten leicht verändern. Werden mit Aktivitäten verknüpfte Zeiten verändert, rechnet das Wegdiagramm nach Eingabe neu und verändert manche Aktivitätspfeile in fette =======> (kritisch), andere in magere -------> (unkritisch). Das macht nicht nur die Beziehung der Aktivitäten leichter verständlich, sondern verdeutlicht so auch Veränderungen, die dem Benutzer vielleicht nicht aufgefallen wären. Es scheint, als habe MicroSoft hier die mit Multiplan erzielten Erfahrungen eingebracht.

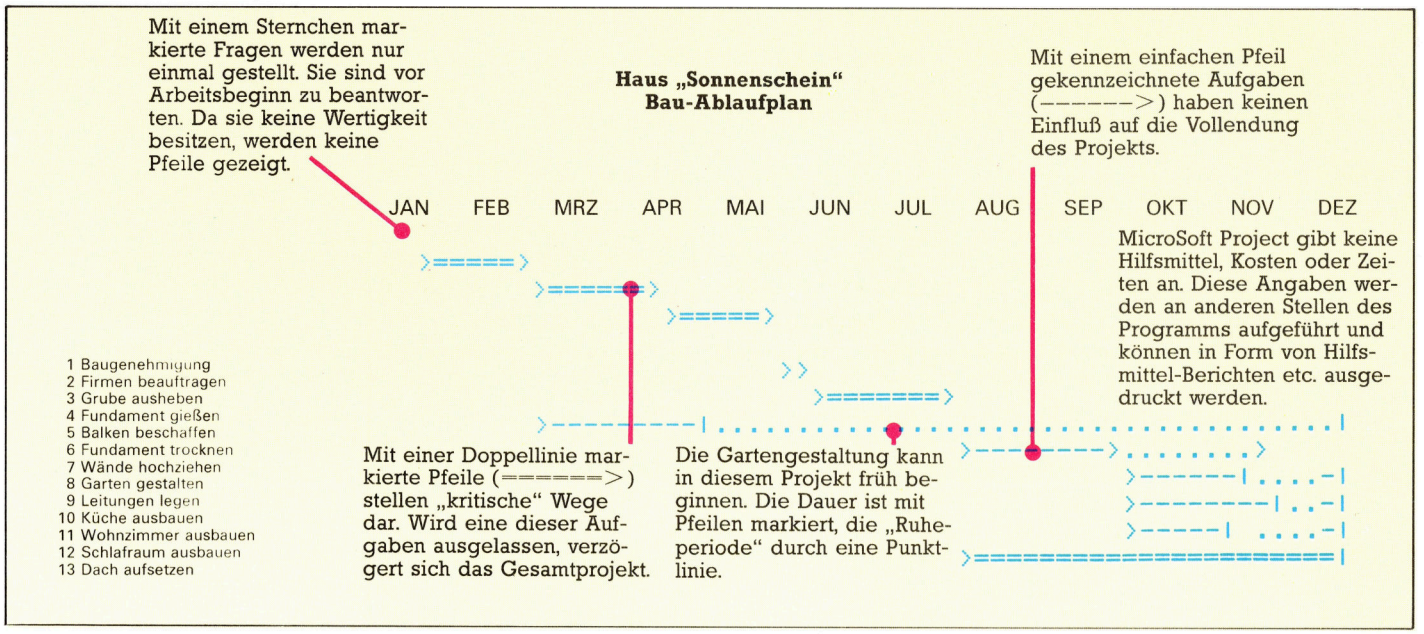
Spreadsheet-Parallelen

Die Benutzung von Microsoft Project entspricht denn auch tatsächlich der Anwendung eines Spreadsheets. Wer damit vertraut ist, wird mit Project sofort klarkommen.

Nach einer kurzen Darstellung des Microsoft-Logo erscheint der Aktivitäten-Bildschirm (ohne Spezifizierung eines File-Namens bleibt der Schirm leer). In der linken Spalte stehen Zahlen von eins bis 19. Am oberen Bildschirmrand befindet sich der Kalender, beginnend mit dem jeweiligen aktuellen Datum. Es kann über OPTIONS aus dem Menü am unteren Bildschirmrand geändert werden. Das Menü umfaßt außerdem:

BLANK CALENDAR DELETE EDIT GOTO
HELP INSERT MOVE PRINT QUIT
RESOURCE SORT TRANSFER XTERNAL

Unter der Legende „Select Option or type command letter" wird die derzeit benutzte Option dargestellt (anfangs ACTIVITY), und in der unteren rechten Ecke der Name des in Arbeit befindlichen Projekts, das automatisch den Na-



Dieses Diagramm stellt einen Plan für einen Hausbau dar. Die Monate erscheinen am oberen Diagramm-Rand, und jede Teilaufgabe erscheint entsprechend der Reihenfolge am linken Rand. Die Dauer jeder Einzelaufgabe (Aktivität) wird durch Pfeile dargestellt.

men TEMP erhält, bis man den Namen ändert.

Die Zeitskala kann nach Tagen, Wochen oder Monaten definiert werden. Eine Veränderung ist jederzeit möglich, um das ganze Projekt auf einen Blick zu übersehen. Hat man sich für eine Tagesskala entschieden, sind zum Beispiel nur die Daten zwischen dem 1. Januar und dem 2. März zu sehen, wogegen eine Monatsskala Überblick über zwei Jahre und drei Monate gibt. Die rechte und linke Cursor-Steuerungstaste werden für die seitliche Bewegung des ACTIVITY-Bildschirms verwendet. In der CALENDAR-Option programmiert der Benutzer zunächst Ferien und andere freie Tage; Samstage und Sonntage werden normalerweise als freie Tage eingegeben. Das läßt sich natürlich ändern, muß aber für jeden einzelnen Tag extra geschehen. Folgemonate lassen sich mitttels <PAGE DOWN>-Taste darstellen, vorangegangene Monate werden mit <PAGE UP> auf den Bildschirm geholt.

Wie bei den meisten Datum-Optionen bei IBM und kompatiblen Rechnern ist wegen der amerikanischen Datumsfolge von MM/DD/YY Sorgfalt geboten, (beispielsweise bedeutet 01/07/85 7. Januar 1985 und nicht 1. Juli 1985). Nach entsprechender Korrektur des Kalenders kann dieser unter Verwendung der TRANSFER-Option geSAVED werden. Das Programm fügt die Silbe .CAL an, um sie von Activitäts-Files (Silbe .ACT) und Quell-Files (.RES) zu unterscheiden.

Darauf beginnt der Benutzer mit der Eingabe der Aktivitäten und der dafür erforderlichen Zeit. Er setzt auch die vorangegangenen Aktivitäten, von denen die folgenden abhängen – seine „Ahnen“. Diese Mehrfachverkettungen werden durch Kommas getrennt. Die Spalte der Aktivitäts-Bezeichnungen ist auf eine Breite von 15 Zeichen begrenzt und kann nicht erweitert werden. Werden mehr als 15 Zeichen eingegeben, speichert der Rechner diese und gibt sie beim Ausdruck auch aus. Nach Eingabe jeder Aktivität – mit Dauer, Ahnen und Startdatum – wird der Bezug der unterschiedlichen Aktivitäten zueinander sofort dargestellt. Ruhezeit (die Aktivität kann ohne Verzögerung des Gesamtprojekts verzögert werden) wird mit Punkten dargestellt. In diesem Stadium werden auch sogenannte Hilfsmittel zugeordnet. Sobald ein solches eingegeben ist, kann man die Liste der Hilfsmittel aufrufen und die Ergebnisse auf dem Bildschirm darstellen lassen.

Nach Eingabe aller Daten läßt sich das komplette Raster seitenweise ausdrucken. Verwendet man die Grafikkarte und benutzt einen geeigneten Drucker, ist ein Langausdruck möglich. Das ist besonders nützlich, weil so die Hardcopies nicht geschnitten und zusammengeklebt werden müssen. Detaillierte Berichte über einzelne Aktivitäten sind ebenfalls erhältlich, wobei die frühesten und spätesten Start- und Endzeiten, die Hilfsmittel, Ahnen und Unterbrechungszeiten dargestellt werden. Dazu gibt es eine Übersicht des Gesamtprojekts mit eben diesen Informationen, allerdings weniger detailliert.

In der Praxis erweist sich der Mangel an ausgefeilter Grafik bei MicroSoft Project kaum als Nachteil. Aufgrund des ausgezeichneten Handbuchs ist es möglich, Informationen zu korrigieren. Für Leute, die mit der Speadsheetarbeit vertraut sind, ist das Programm sehr anwenderfreundlich. Es ist schneller als MacProject, dafür aber auch dreimal so teuer.

MicroSoft Project: Für MS-DOS-Rechner
Hersteller: MicroSoft Ltd., Piper House, Hatch Lane, Windsor, Berkshire, Großbritannien
Autoren: MAS, Seattle, USA
Format: Diskette

Ein maßvolles Gefährt

Es geht wieder weiter mit dem Ausbau unseres Roboters. In diesem Abschnitt soll ein Programm entwickelt werden, durch das er die Länge eines Gegenstands genau messen kann.

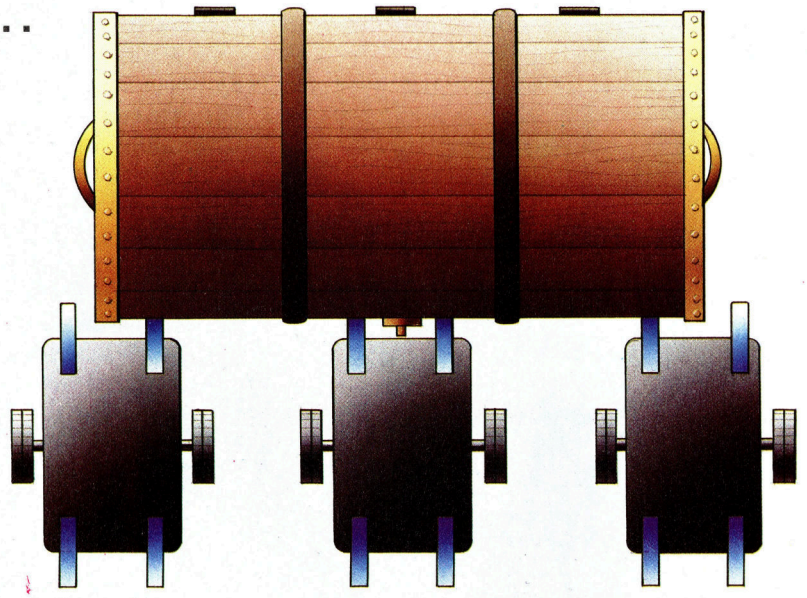
Damit der Roboter einen Gegenstand exakt lokalisieren und seine Kantenlänge feststellen kann, brauchen wir ein präzises Rechenprogramm. Als Sensor für die Messung werden die Microschalter genutzt. Die Aufgabe muß in Einzelschritten gelöst werden:
1) Gegenstand finden.
2) Eine Ecke des Gegenstands erkennen.
3) Seite des Gegenstands nach der zweiten Ecke absuchen.
Der erste Schritt ist noch relativ einfach: Wir gehen davon aus, daß der Roboter beim Programmstart unser Meßobjekt berührt. Was aber, wenn nur einer der beiden Microtaster und nicht beide zusammen geschlossen sind? Die möglichen Variationen sind in der Zeichnung dargestellt. Es läßt sich einfach feststellen, ob der rechte oder der linke Sensor geschlossen ist. Auf diese Information können wir die weitere Handlungsstrategie des Roboters aufbauen.

Voraussetzung für den korrekten Ablauf ist zunächst, daß der Roboter beim Programmstart in einem Winkel von 90 Grad zum Meßobjekt steht. Den Fall einer Berührung im schiefen Winkel werden wir nicht berücksichtigen.

Auch der zweite Schritt des Lösungsweges kann vereinfacht werden: Unser Roboter soll sich immer zuerst zur rechten Ecke des Gegenstands begeben, bevor er mit dem Messen anfängt. Dazu muß er sich so lange nach rechts vortasten, bis nicht mehr beide, sondern nur noch der linke Microtaster Kontakt meldet. Dieses Vortasten erfordert eine komplizierte Bewegungsfolge aus fünf Einzelschritten. Wenn der Roboter das Meßobjekt anfangs berührt, sieht sie so aus: Zurückfahren, Drehung um 90 Grad nach rechts, ein kleines Stück vorwärts, um 90 Grad zurückdrehen und danach so weit vorfahren, bis die Sensoren wieder Kontakt mit dem Meßobjekt melden. Als „Schrittweite" (Abstand zwischen zwei Berührungspunkten an der Seite des Gegenstands) bezeichnen wir die Länge eines einzelnen parallel zum Meßobjekt gefahrenen Streckenabschnitts.

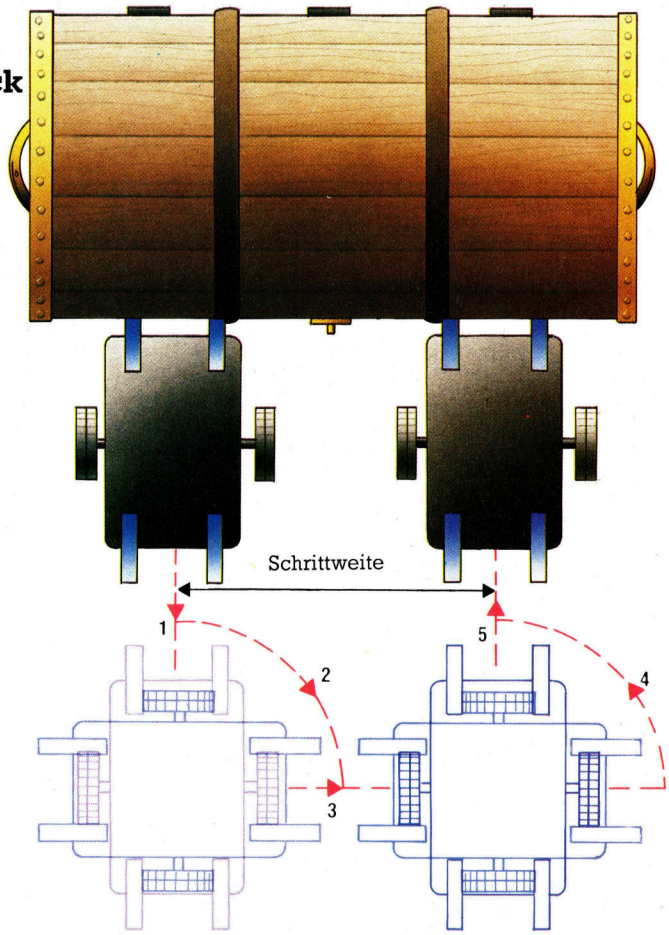
Steuern „nach Gefühl" ...

In der Zeichnung sind die drei verschiedenen Zustände dargestellt, die beim Kontakt der Tasthebel mit einem Gegenstand auftreten können. Wenn nur der rechte Sensor geschlossen ist, hat der Roboter die linke Ecke des Hindernisses erreicht. Geben beide Microtaster Kontakt, muß er sich irgendwo in der Mitte zwischen den Ekken befinden. Ein Schließen des linken Sensorkontaktes allein signalisiert, daß er die rechte Ecke erreicht hat.



Nur rechter Taster geschlossen | Beide Taster geschlossen | Nur linker Taster geschlossen

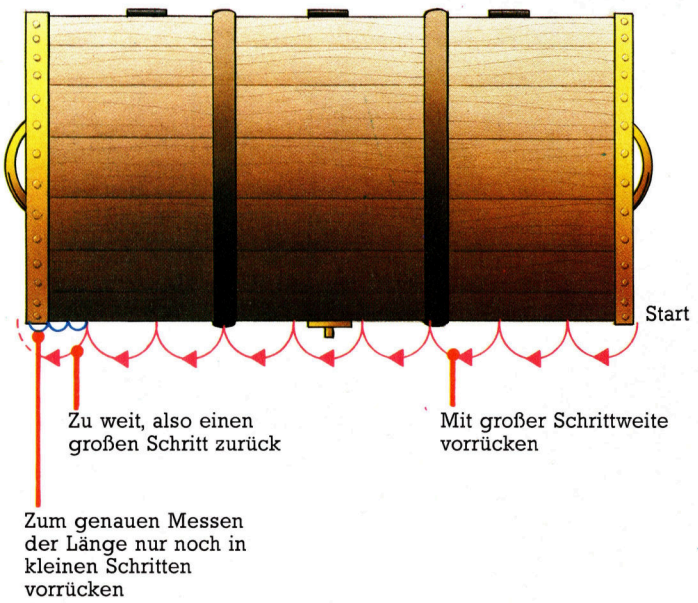
Vor und Zurück



Oben: Nach dem Schließen beider Taster setzt der Roboter zurück (1), damit er sich ohne Berührung mit dem Hindernis um 90 Grad nach rechts drehen kann (2). Darauf folgt ein Schritt vorwärts (3) und ein Richtungswechsel von 90 Grad nach links (4). Mit abschließendem Vorwärtsfahren (5) kann der Roboter dann testen, ob er noch vor dem Gegenstand steht.

Der beschriebene Ablauf wird wiederholt, bis eine Ecke erreicht wird. Dabei geht der Roboter zuerst mit großen Schritten voran. Wenn nur noch einer oder kein Sensor mehr Kontakt meldet, wird der letzte Schritt rückwärts ausgeführt und das Ganze mit kleinen Schritten wiederholt.

Testen durch Tasten



Für eine möglichst genaue Messung erscheint auf den ersten Blick eine Schrittweite von nur wenigen Millimetern erstrebenswert. Es geht aber auch anders: Der Roboter rückt in größeren Schritten so lange weiter, bis er über das Ende des Gegenstands hinausfährt. Von dort kehrt er zur Position vor dem letzten Schritt zurück und arbeitet sich erst dann mit kleineren Schritten an die rechte Ecke heran. Der Abstand zwischen den beiden Tasthebeln der vorderen Sensoren – circa 60 mm – ist als Schrittweite besonders günstig. Gerät der Roboter über die Ecke hinaus, wird beim neuerlichen Heranfahren an das Meßobjekt immer noch einer der beiden Taster geschlossen.

Im dritten Teil unseres Lösungsansatzes wird die eigentliche Messung vorgenommen. Dazu tastet der Roboter Schritt für Schritt von der rechten zur linken Ecke. Die Zahl der für den Weg benötigten Schritte wird in einer Variablen gespeichert.

Programmiertes Abtasten

Das Verhältnis Impuls/zurückgelegte Strecke bzw. Impuls/Winkel – im letzten Abschnitt des Projekts ermittelt – richtet sich nach den Eigenschaften Ihres Roboters. Durch die Einsatzmöglichkeit von Prozeduren (PROC) ist das BASIC des Acorn B für unsere Aufgabe besonders gut geeignet. Das Programm kann gut durchstrukturiert werden und die Bewegungen des Roboters durch Aufrufen einzelner Prozeduren steuern.

Nachdem die wichtigsten Aufgaben des Programms definiert wurden, lassen sich die Einzelbewegungen des Roboters zu einem kompletten „Tastschritt" zusammenstellen. Der gesamte Meßvorgang setzt sich wiederum aus mehreren dieser Tastschritte zusammen. In diesem Beispiel werden Prozeduren auf ganz unterschiedlichem Niveau eingesetzt – im einfachsten Fall für die Versorgung des Motors mit Steuerimpulsen, im schwierigsten für die Überwachung des Meßvorgangs selbst.

Probleme gibt es, wenn der Roboter beim Programmstart nicht genau im rechten Winkel zum Meßobjekt steht. Die Logik des Programms setzt das Schließen eines einzelnen Sensors immer mit dem Erreichen einer Ecke des Gegenstands gleich. In diesem Fall helfen nur die BREAK-Taste und ein neuer Programmstart mit exakt aufgestelltem Roboter.

Zuvor müssen jedoch noch eventuelle Fehlerquellen eliminiert werden: Zwei 5 mm breite Tasthebel können beim Erkennen einer Ecke einen „Gesamtfehler" von maximal 10 mm erzeugen. Beim genauen Suchen nach den Ekken arbeitet der Roboter mit einer Schrittweite von 5 mm, was einen zusätzlichen Meßfehler von 10 mm hervorruft. Der Test mit unserem Prototyp ergab beim Messen eines 410 mm langen Gegenstandes Fehler von 20 mm. Die Abweichung blieb also unter fünf Prozent.

Acorn-Listing

```
1000 REM **** BBC ROBOT MEASURE ****
1010 MODE 7
1020 PROCinitialise
1030 PROCmeasure
1040 PROCprintout
1050 END
1060 DEF PROCmeasure
1070 PROCfind
1080 REM ** ONE BUMPER ONLY ? **
1090 PROCtest_bumpers
1100 REM **** FIND END ****
1110 REPEAT:PROCprobe(right,width)
1120 UNTIL (?DATREG AND 192)=left_bumper
1130 REM ** GO BACK AND INCH TO END **
1140 PROCprobe(left,width)
1150 REPEAT:PROCprobe(right,small_width)
1160 UNTIL (?DATREG AND 192)=left_bumper
1170 PRINT"FOUND RIGHT-HAND END"
1180 PRINT"STARTING TO MEASURE"
1190 REM ** START TO MEASURE **
1200 count=width
1210 REPEAT:PROCprobe(left,width)
1220 count=count+width
1230 UNTIL (?DATREG AND 192)=right_bumper
1240 REM ** GO BACK AND INCH TO END **
1250 count=count-width
1260 PROCprobe(right,width)
1270 REPEAT:PROCprobe(left,small_width)
1280 count=count+small_width
1290 UNTIL (?DATREG AND 192)=right_bumper
1300 ?DATREG=0
1310 ENDPROC
1320 :
1330 DEF PROCprintout
1340 CLS
1350 PRINTTAB(5,12)"OBJECT SIDE MEASURED AT
     ";count;" mm"
1360 ENDPROC
1370 :
1380 DEF PROCinitialise
1390 DDR=&FE62:DATREG=&FE60
1400 ?DDR=15:REM LINES 0-3 OUTPUT
1410 ?DATREG=1:REM TURN ON RESET BIT
1420 forwards=4:backwards=2:left=6:right=0
1430 pd_ratio=3.34446:pa_ratio=375/90
1440 right_bumper=128:left_bumper=64
1450 both_bumpers=0:neither_bumpers=192
1460 width=60:small_width=5
1470 ENDPROC
1480 :
1490 DEF PROCsearch(sense)
1500 REPEAT:PROCprobe(sense,width)
1510 UNTIL (?DATREG AND 192)=both_bumpers
1520 ENDPROC
1530 :
1540 DEF PROCfind
1550 REPEAT:PROCmove(forwards,8)
1560 UNTIL(?DATREG AND 192)<>neither_bumpers
1570 ENDPROC
1580 :
1590 DEF PROCtest_bumpers
1600 IF (?DATREG AND 192)=right_bumper THEN
     PROCsearch(right):ENDPROC
1610 IF (?DATREG AND 192)=left_bumper THEN
     PROCsearch(left):ENDPROC
1620 ENDPROC
1630 :
1640 DEF PROCprobe(way,step)
1650 IF way=right THEN opp_way=left ELSE
     opp_way=right
1660 PROCmove(backwards,30)
1670 PROCturn(way,90)
1680 PROCmove(forwards,step)
1690 PROCturn(opp_way,90)
1700 REPEAT:PROCmove(forwards,8)
1710 UNTIL (?DATREG AND 192)<>neither_bumpers
1720 ENDPROC
1730 :
1740 DEF PROCmove(dir,distance)
1750 ?DATREG=(?DATREG AND 1)OR dir
1760 pulses=pd_ratio*distance
1770 FOR I=1 TO pulses:PROCpulse:NEXT I
1780 ENDPROC
1790 :
1800 DEF PROCturn(dir,angle)
1810 ?DATREG=(?DATREG AND 1)OR dir
1820 pulses=pa_ratio*angle
1830 FOR I=1 TO pulses:PROCpulse:NEXT I
1840 ENDPROC
1850 DEF PROCpulse
1860 ?DATREG=(?DATREG OR 8)
1870 ?DATREG=(?DATREG AND 247)
1880 ENDPROC
```

Commodore-64-Listing

```
10 REM **** CBM ROBOT MEASURE ****
20 GOSUB1000:REM INITIALISE
30 GOSUB2000:REM MEASURE
40 GOSUB3000:REM PRINTOUT
50 END
60 :
1000 REM **** INITIALISE ****
1010 DDR=56579:DATREG=56577
1020 POKE DDR,15:REM LINES 0-3 OUTPUT
1030 POKE DATREG,1:REM TURN ON RESET BIT
1040 FW=4:BW=2:LF=6:RT=0
1050 PD=3.34446:PA=375/90
1060 RB=128:LB=64:BB=0:NB=192
1070 WD=60:SW=5
1080 RETURN
1090 :
2000 REM **** MEASURE ****
2010 GOSUB3500:REM FIND OBJECT
2020 GOSUB4000:REM TEST BUMPERS
2030 REM ** FIND END **
2040 WY=RT:SP=WD:GOSUB6000:REM PROBE
2050 IF(PEEK(DATREG)AND192)<>LB THEN 2040
2060 REM ** GO BACK AND INCH TO END **
2070 DR=LF:DS=WD:GOSUB6000:REM PROBE
2080 DR=RT:DS=SW:GOSUB6000:REM PROBE
2090 IF(PEEK(DATREG)AND192)<>LB THEN 2080
2100 PRINT"FOUND RIGHT-HAND END"
2110 PRINT"STARTING TO MEASURE"
2120 REM ** START TO MEASURE **
2130 CC=WD
2140 DR=LF:DS=WD:GOSUB6000:CC=CC+WD
2150 IF(PEEK(DATREG)AND192)<>RB THEN 2140
2160 REM ** GO BACK AND INCH TO END **
2170 CC=CC-WD
2180 DR=RT:DS=WD:GOSUB6000:CC=CC+SW
2190 IF(PEEK(DATREG)AND192)<>RB THEN 2180
2200 POKE DATREG,0
2210 RETURN
2220 :
3000 REM **** PRINT OUT ****
3010 PRINTCHR$(147)
3020 PRINT"OBJECT MEASURED AT ;"CC;"MM"
3030 RETURN
3040 :
3500 REM **** FIND ****
3510 DR=FW:DS=5:GOSUB7000:REM MOVE
3520 IF(PEEK(DATREG)AND192)=NB THEN 3510
3530 RETURN
3540 :
4000 REM **** TEST BUMPERS ****
4010 IF(PEEK(DATREG)AND192)=RB THEN SS=RT:
     GOSUB5000:RETURN
4020 IF(PEEK(DATREG)AND192)=LB THEN SS=LF:
     GOSUB5000:RETURN
4030 RETURN
4040 :
5000 REM **** SEARCH (SS) ****
5010 DR=FW:DS=8:GOSUB7000:REM MOVE
5020 IF(PEEK(DATREG)AND192)=NB THEN 5010
5030 RETURN
5040 :
6000 REM **** PROBE (WY,SP) ****
6010 IF WY=RT THEN OW=LF
6020 IF WY=LF THEN OW=RT
6030 DR=BW:DS=30:GOSUB7000:REM MOVE
6040 DR=WY:AG=90:GOSUB7500:REM TURN
6050 DR=FW:DS=SP:GOSUB7000:REM MOVE
6060 DR=OW:AG=90:GOSUB7500:REM TURN
6070 DR=FW:DS=8:GOSUB7000:REM MOVE
6080 IF(PEEK(DATREG)AND192)=NB THEN 6070
6090 RETURN
6100 :
7000 REM **** MOVE (DR,DS) ****
7010 POKE DATREG,(PEEK(DATREG)AND 1)OR DR
7020 PL=PD*DS
7030 FOR I=1 TO PL:GOSUB8000:NEXT I
7040 RETURN
7050 :
7500 REM **** TURN (DR,AG) ****
7510 POKE DATREG,(PEEK(DATREG)AND 1)OR DR
7520 PL=PA*AG
7530 FOR I=1 TO PL:GOSUB8000:NEXT I
7540 RETURN
7550 :
8000 REM **** PULSE ****
8010 POKE DATREG,PEEK(DATREG)OR 8
8020 POKE DATREG,PEEK(DATREG)AND 247
8030 RETURN
```

Commodore-Grafik

In den letzten zwei Kapiteln wurden Bildschirmdarstellungen für zwei spezielle Orte in Digitaya für den Spectrum und Acorn B ausgearbeitet. Heute wollen wir diese Darstellungen auf dem Commodore 64 ausarbeiten und programmieren.

Die Ausarbeitung der Abenteuer-Grafik für den Acorn B und Spectrum verlief ähnlich. Beide Computer haben hochauflösende Grafiken und einfache PRINT-Formate. Unterschiede lagen vor allem in den Bildschirmdimensionen und BASIC-Befehlswörtern, die für das hochauflösende Zeichnen gelten.

Beim C 64 dagegen gibt es keine BASIC-Befehle für hochauflösende Grafik, und die Erstellung der relevanten PEEKs und POKEs in BASIC für hochauflösende Darstellungen ist für unsere Zwecke zu aufwendig. Daher müssen die einfacher verwendbaren Möglichkeiten des C 64 eingesetzt werden.

Aus Grafikzeichen oder Kombinationen hieraus können große Buchstaben oder andere Darstellungen erstellt werden. Mit Sprites können hochauflösende Formen auf den Bildschirm gebracht werden. Grafikzeichen können entweder durch eine Reihe von PRINT-Anweisungen oder durch POKEn des entsprechenden Codes auf dem Bildschirm positioniert werden.

In Zeile 8020 bis 8170 des „Joystick-Port"-Listings werden die Daten für die zwei in dieser Routine verwendeten Sprites eingelesen. Sprite 0 wird durch die ersten 63 Zahlen in der Gruppe von DATA-Anweisungen zwischen Zeile 8450 und 8497 definiert und repräsentiert den Joystick-Port. Oftmals werden Sprite-Daten innerhalb eines BASIC-Programms ganz am Anfang gesetzt. Bei einem umfangreichen Listing besteht aber das Risiko, daß diese Daten überschrieben werden.

Sprite 0 wird auf doppelte Breite gestreckt, um die zuletzt dargestellte Form durch Setzen von Bit 0 des horizontalen Ausdehnungsregisters in Zeile 8170 zu erstellen. Beachten Sie, daß sich alle Register für die Sprite-Definitionen, wie zum Beispiel Farbe, Position und Ausdehnung, auf die Startadresse für den Videokontrollchip beziehen. Die Adresse VIC+29 des horizontalen Ausdehnungsregisters ist einfacher im Gedächtnis zu behalten als seine aktuelle Speicherstelle (53277). Einige Sprite-Attribute – wie die X- und Y-Koordinatenregister – belegen ein komplettes Register für jedes Sprite. Sprite 1 wird durch die verbleibenden 13 Zahlen in den DATA-Anweisungen definiert und repräsentiert ein Objekt, das aus dem Joystick-Port abgeschossen wird.

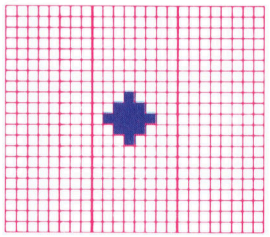
Da der „feste" Teil von Sprite 1 klein ist (er entspricht einem Projektil), ist es schneller, die Daten einzugeben, als sie in zwei Schritten zu definieren. Zuerst werden 63 Nullen in den definierten Bereich gePOKEt und dann die wenigen Zahlen für die Form-Definition mit READ und POKE eingebracht. So können wir auf die Anzahl von Nullen verzichten, die sonst erforderlich wäre.

Die Zeilen 8190 bis 8220 erstellen Strings, die aus einer Reihe von Grafikzeichen bestehen. LE$ bildet eine horizontale Linie in Bildschirmbreite durch Kombination von 40 speziellen Grafikzeichen. DW$ ist eine Reihe von Abwärtspfeil-Zeichen. LS$ und RS$ sind Gruppen rechter und linker Diagonalen, die für die Erstellung eines Rasters im Vordergrund verwendet werden.

Die Shoot-Routine in Zeile 8310 wählt einen zufälligen Punkt am unteren Bildschirmrand und lenkt Sprite 1 so lange zu diesem Punkt,

Der „Schuß" aus dem Joystick-Port wird über ein als Projektil definiertes Sprite erreicht. Die Umrisse werden durch POKEn von Nullen in den definierten Bereich gezogen. Die „festen" Flächen werden aus DATA-Anweisungen gelesen und an die korrekte Sprite-Position gePOKEt.

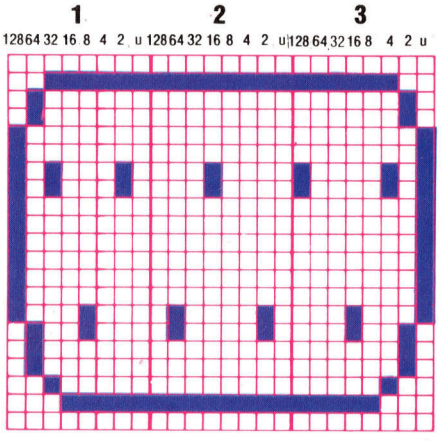
Projektil – Sprite 1



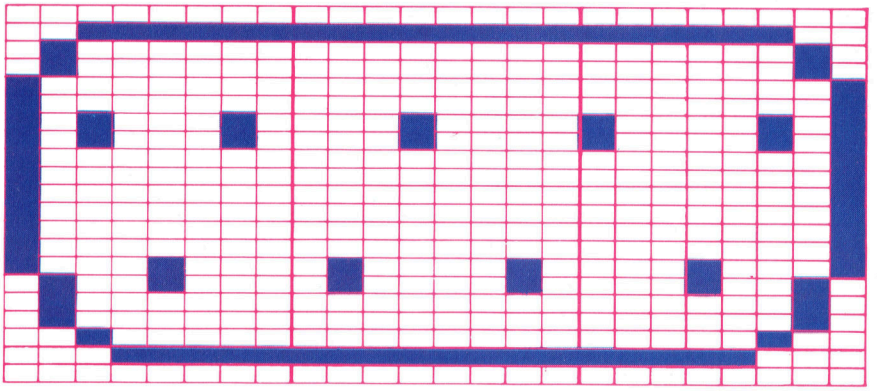
0	16	0
0	56	0
0	124	0
0	56	0
0	16	0

Die aufgeführten Werte für das Joystick-Sprite werden aus DATA-Anweisungen gelesen und an die entsprechenden Sprite-Plätze gePOKEt. Durch Verändern der Werte des horizontalen Ausdehnungsregisters kann die Darstellung horizontal gedehnt werden.

Joystick-Port – Sprite 0



1	2	3
0	0	0
63	255	252
64	0	2
64	0	2
128	0	1
128	0	1
162	16	133
162	16	133
128	0	1
128	0	1
128	0	1
128	0	1
128	0	1
128	0	1
136	66	17
72	66	18
64	0	2
64	0	2
32	0	4
31	255	248
0	0	0



bis der Spieler eine Taste drückt. Die Bildschirmfarben werden auf Normal zurückgesetzt, der Bildschirm wird gelöscht, und die Sprites werden vor der Rückkehr zum Hauptprogramm abgeschaltet. Für den Einsatz dieser Unterroutine in Digitaya wird folgende Zeile eingefügt:

```
3845 GOSUB 8000: REM JOYSTICK PORT
     PICTURE
```

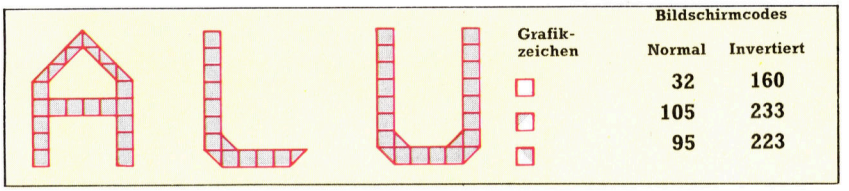
Das andere Listing liefert eine Grafik für den ALU-Ort in Digitaya und zeigt verschiedene Methoden der Zeichendarstellung auf dem Schirm. Die Zeilen 7040 bis 7090 lesen eine Anzahl von DATA-Anweisungen und POKEn die Werte direkt in den Bildschirmbereich. Hier ist der Farbcode 2, um die Zeichen in Rot darzustellen.

Das Wort ALU wird mit einem ungewöhnlichen Trick über den Bildschirm gerollt. Die erste Zeile mit Grafikzeichen-Codes, aus denen das Wort ALU gebildet wird, wird in die zweite Bildschirmzeile gePOKEt. Dann wird die Unterroutine in Zeile 7680 aufgerufen, durch die die Darstellung eine Zeile nach unten geschoben wird. Die zweite Code-Zeile wird in denselben Bereich gePOKEt wie die erste, und die Unterroutine wird erneut aufgerufen. Eine Wiederholung für alle acht Code-Zeilen bewirkt ein Abwärtsrollen des Wortes ALU.

Nachstehend werden zwei weitere Methoden zur Darstellung von Zeichendaten demonstriert. Zeichen können mit PRINT direkt dargestellt (Zeile 7130 und 7140) oder als Datenstrings gelesen werden (Zeile 7170 und 7590 bis 7670). Diese zweite Methode erleichtert die Ausarbeitung in DATA-Anweisungen. Dazu wird folgende Zeile eingefügt:

```
4565 GOSUB 7000:ALU PICTURE
```

Der ALU-Ort in Digitaya wird aus drei Grafikzeichen in niedriger Auflösung erstellt. Die großen Buchstaben scheinen vom oberen Bildschirmrand an ihre Endposition zu rollen.



ALU-Programm

```
7000 REM **** ALU PICTURE S/R ****
7010 VIC=53248:CS=55296:SC=1024
7020 PRINT CHR$(147):REM CLEAR SCREEN
7030 POKE VIC+32,0:POKE VIC+33,0:REM SET SCREEN/BORD
7040 CC=0
7050 FOR J=1 TO 8:GOSUB7680:REM SCROLL
7060 FOR I=47 TO 72
7070 READ A:CC=CC+A:POKE SC+I,A:POKE CS+I,2
7080 NEXT I,J
7090 READ CS:IF CS<>CC THEN PRINT"CHECKSUM ERROR":STOP
7100 GOSUB7680:REM SCROLL
7110 PRINTCHR$(158):REM TEXT YELLOW
7120 FOR I=1 TO 8:PRINT:NEXT I:REM MOVE DOWN
7130 PRINTTAB(9)"AND";SPC(7);"OR";SPC(8);"NOT"
7140 PRINTTAB(10)"0";SPC(9);"0";SPC(9);"0"
7150 PRINTCHR$(28):REM TEST RED
7160 REM ** QUESTION MARK **
7170 FOR I=1 TO 9:READ Q$:PRINTTAB(16)Q$:NEXT I
7180 REM **** AWAIT KEY AND RESET ****
7190 GET A$:IF A$="" THEN 7190
7200 POKE VIC+32,14:POKE VIC+33,6:REM SCREEN/BORD
7210 PRINTCHR$(154):REM LT BLUE TEXT
7220 PRINTCHR$(147):REM CLEAR SCREEN
7230 RETURN
7240 REM **** SCREEN DATA ****
7250 REM ** ROW 1 **
7260 DATA 160,32,32,32,32,160,32,32,32,32
7270 DATA 95,160,160,160,160,105
7280 DATA 32,32,32,32,95,160,160,160,160,105
7290 REM **ROW 2 **
7300 DATA 160,32,32,32,32,160,32,32,32,32
7310 DATA 160,223,32,32,32,32,32,32,32,32
7320 DATA 160,223,32,32,233,160
7330 REM ** ROW 3 **
7340 DATA 160,32,32,32,32,160,32,32,32,32
7350 DATA 160,32,32,32,32,32,32,32,32,32
7360 DATA 160,32,32,32,32,160
7370 REM ** ROW 4 **
7380 DATA 160,160,160,160,160,160,32,32,32,32
7390 DATA 160,32,32,32,32,32,32,32,32,32
7400 DATA 160,32,32,32,32,160
7410 REM ** ROW 5 **
7420 DATA 160,32,32,32,32,160,32,32,32,32
7430 DATA 160,32,32,32,32,32,32,32,32,32
7440 DATA 160,32,32,32,32,160
7450 REM ** ROW 6 **
7460 DATA 233,105,32,32,95,223,32,32,32,32
7470 DATA 160,32,32,32,32,32,32,32,32,32
7480 DATA 160,32,32,32,32,160
7490 REM ** ROW 7 **
7500 DATA 32,233,105,95,223,32,32,32,32,32
7510 DATA 160,32,32,32,32,32,32,32,32,32
7520 DATA 160,32,32,32,32,160
7530 REM ** ROW 8 **
7540 DATA 32,32,233,223,32,32,32,32,32,32
7550 DATA 160,32,32,32,32,32,32,32,32,32
7560 DATA 160,32,32,32,32,160
7570 DATA 14463:REM CHECKSUM
7580 REM ** QUESTION MARK DATA **
7590 DATA "  ?????  "
7600 DATA "?      ?"
7610 DATA "?      ?"
7620 DATA "     ??  "
7630 DATA "    ?    "
7640 DATA "   ?     "
7650 DATA "  ?      "
7660 DATA "         "
7670 DATA "    ●    "
7680 REM **** SCROLL SCREEN S/R ****
7690 POKE 218,160
7700 PRINTCHR$(19);CHR$(17);CHR$(157);CHR$(148)
7710 RETURN
```

Buchsen-Darstellung

```
8000 REM **** JOYSTICK PICTURE S/R ****
8010 PRINTCHR$(147):REM CLEAR SCREEN
8020 S0=832:S1=S0+64:REM SPR DATA START ADDR
8030 CC=0:VIC=53248:REM START OF VIC CHIP
8040 FOR I=S0 TO S0+62:READ A:CC=CC+A:POKE I,A:NEXT
8050 FOR I=S1 TO S1+62:POKE I,0:NEXT I
8060 FOR I=S1+25 TO S1+37:READ A:CC=CC+A:POKE I,A:NEXT I
8065 READ CS:IF CS<>CC THEN PRINT"CHECKSUM ERROR":STOP
8070 POKEVIC+33,0:POKEVIC+32,0:REM SET SCREEN/BORDER
8080 REM ** SET SPRITE POINTERS **
8090 POKE 2040,S0/64:POKE 2041,S1/64
8100 REM ** SET VIC SPRITE CONTROL REGS **
8110 POKE VIC+39,7:REM SET SPR 0 COLOUR
8120 POKE VIC+40,7:REM SET SPR 1 COLOUR
8130 POKE VIC,65:REM SET SPR 0 X COORD
8140 POKE VIC+1,70:REM SET SPR 0 Y COORD
8150 POKE VIC+2,74:REM SET SPR 1 X COORD
8160 POKE VIC+3,70:REM SET SPR 1 Y COORD
8170 POKE VIC+29,1:REM EXPAND SPR 0 HORIZ
8190 LE$="":FOR I=1 TO 40:LE$=LE$+CHR$(195):NEXT I
8200 DW$="":FOR I=1 TO 25:DW$=DW$+CHR$(17):NEXT I
8210 LS$="":FOR I=1 TO 11:LS$=LS$+CHR$(206)+" ":NEXT I
8220 RS$="":FOR I=1 TO 11:RS$=RS$+CHR$(205)+" ":NEXT I
8230 PRINTCHR$(158):REM TEXT YELLOW
8240 PRINTCHR$(19):PRINT TAB(2)"JOYSTICK PORT"
8250 PRINTCHR$(154):REM TEXT LT BLUE
8260 PRINTCHR$(19);LEFT$(DW$,17);LE$
8270 PRINT CHR$(145);
8280 PT$=LEFT$(LS$,19)+LEFT$(RS$,21)
8290 PT$=PT$+RIGHT$(LS$,19)+RIGHT$(RS$,21)
8300 FOR I=1 TO 3:PRINT PT$;:NEXT I
8305 POKE VIC+21,3:REM TURN ON SPR 0 & 1
8310 REM ** SHOOT **
8320 YB=240:Y1=70
8330 X1=74:X=INT(RND(1)*150)+24
8340 G=2*(X-X1)/(YB-Y1)
8350 FOR Y=Y1 TO YB STEP 2
8360 X1=X1+G:POKE VIC+2,X1:POKE VIC+3,Y
8370 NEXT Y
8380 GET A$:IF A$="" THEN 8330
8390 REM ** RESET SCREEN **
8400 POKE VIC+21,0:REM TURN SPRITES OFF
8410 POKE VIC+32,14:POKEVIC+33,6:REM RESET SCREEN/BORD
8420 PRINT CHR$(147):REM CLEAR SCREEN
8430 RETURN
8440 REM **** SPRITE DATA ****
8450 DATA 0,0,0,63,255,252,64,0,2,64,0,2
8460 DATA 128,0,1,128,0,1,162,16,133,162,16,133
8470 DATA 128,0,1,128,0,1,128,0,1,128,0,1
8480 DATA 128,0,1,128,0,1,136,66,17,72,66,18
8490 DATA 64,0,2,64,0,2,32,0,4,31,255,248
8495 DATA 0,0,0,16,0,0,56,0,0,124,0,0,56,0,0,16
8497 DATA 3701:REM CHECKSUM
```

Voll im Bild

Eine unzulängliche Bildwiedergabe ist häufig Ursache für ein schnelles Ermüden der Augen. In diesem Artikel werden die grundlegenden Unterschiede zwischen Monitoren und Fernsehgeräten aufgezeigt.

Der Bildschirm als Peripheriegerät hat beim Heimcomputer besondere Bedeutung. Häufig wird einfach der häusliche Fernseher benutzt oder ein extra gekauftes billiges Schwarzweißgerät. Dies erfüllt den Zweck, aber die Bildqualität ist keineswegs optimal.

Ursache ist vor allem eine mehrfache Signalumformung auf dem Weg vom Bildspeicher des Rechners bis zum Fernsehschirm, die auch bei aufwendiger Schaltungstechnik stets Qualitätseinbußen bedingt. Im Endergebnis ist die Wiedergabe unscharf und schwer lesbar, oft kommen noch unerfreuliche Flimmereffekte hinzu.

Voraussetzung für ein wirklich hochwertiges Bild ist daher das Ausschalten der Signalverzerrungen, die bei der Umsetzung in den Modulator- und Demodulatorschaltungen auftreten, und zwar dadurch, daß diese Stufen aus der Übertragungskette herausgenommen werden. Neben der höheren Bandbreite ist dies das wesentliche „Geheimnis" eines Monitors.

Da die Empfängerschaltung fehlt, kann der Monitor nicht an der Antennenbuchse Ihres Rechners angeschlossen werden. Sie brauchen einen Video- oder RGB-Ausgang. Fehlt an Ihrem Computer eine eindeutige Beschriftung, schauen Sie in der Anleitung nach, ob das unmodulierte Bildsignal herausgeführt wird (das ist entscheidend).

Das Zustandekommen des Bildes auf dem Schirm ist weitgehend eine Frage der Synchronisation, d. h. der richtigen zeitlichen Steuerung der Vorgänge. Das eigentliche Problem ist dabei, daß die horizontale und die vertikale Ablenkspannung für die Schirmabtastung durch den Elektronenstrahl innerhalb des Monitors erzeugt wird und daß der Videobaustein im Rechner darauf leider nur indirekt Einfluß nehmen kann.

Wird der Monitor bei offenem Eingang eingeschaltet, tastet der Strahl 50mal pro Sekunde den Bildschirm ab und leuchtet dabei das ganze Feld gleichmäßig aus. Soll ein stehendes Bild erscheinen, muß der Elektronenstrahl

Monochrom-Monitore
Mit den Einfarb-Geräten kommen Sie am billigsten weg. Diese Monitore sind für die Textverarbeitung und andere kommerzielle Anwendungen ebenso einsetzbar wie für die Wiedergabe einfacher Grafiken.

Farbmonitore
Der CUB von Microvitec ist ein ausgezeichnetes Beispiel aus der Gruppe der preisgünstigen Farbmonitore für Heimcomputer. Die Auflösung ist zwar nicht absolute Spitze, aber das Gerät ist für sehr viele Einsatzbereiche vollständig ausreichend, einschließlich der meisten Grafikanwendungen.

Japanische Vielseitigkeit
Der TM90PSN kann erkennen, welcher Norm das Eingangssignal entspricht, und schaltet dann auf die zugehörige Decodierung um. Er läßt sich als Datensichtgerät für den Rechner und auch in Verbindung mit Videorecordern oder Bildplattenspielern verwenden.

Heimfernseher
Bei den Heimfernsehgeräten wurde die Bildqualität in den letzten Jahren durch Fortschritte in der Bildröhrentechnologie, speziell bei der Justierung der Elektronenstrahlsysteme, erheblich gesteigert. Sony bietet die Möglichkeit, das Gerät auch als Monitor einzusetzen. Die Antennenbuchse ist an leicht zugänglicher Stelle im Frontrahmen.

dabei stets in genau den gleichen Rasterpositionen auf- und abgeblendet werden.

Das ganze Verfahren steht und fällt mit den Synchronimpulsen, die gemeinsam mit dem Helligkeitssignal im Rechner erzeugt und an den Monitor weitergegeben werden: Zu jeder Zeile gehört ein Zeilensynchronimpuls, und zu jedem kompletten Bild (d. h. 50mal pro Sekunde) gibt es ein Bildsynchronsignal, das den Monitor nach dem zeilenweisen Aufbau des vollständigen Bildes anweist, den Elektronenstrahl zur linken oberen Schirmecke zurückzuführen und ein neues Bild zu beginnen. Der Synchronimpuls am Ende jeder Zeile veranlaßt nur den Rücklauf des Strahls zum linken Bildrand und das Ansetzen einer neuen Zeile. Bei Zeilen- und Bildrücklauf muß der Strahl abgeblendet („ausgetastet") werden.

BAS und FBAS

Bei der Vielfalt der Monitore ist im wesentlichen nach Farb- und Monochromgeräten zu unterscheiden. Wichtig ist noch die Art des Eingangs. Die Monochrom-Monitore akzeptieren fast ausschließlich ein Composite Video (deutsch: BAS = Bild-Austast-Synchron)-Signal. Dabei kommen der Bildhelligkeits (Video)-Anteil und die Synchronimpulse als Misch-(Composite)-Signal an, das der Monitor für die Bilderzeugung wieder zerlegt.

Bandbreite

Bei der Auswahl Ihres Monitors ist die Bandbreite ein wesentliches Kriterium. Je höher die Bandbreite, desto feinere Details sind darstellbar.

Als erstes müssen Sie ermitteln, welche Bandbreite Sie für Ihren Computer mindestens brauchen. Dazu berechnen Sie zunächst die maximale Zeichenanzahl auf dem Bildschirm, indem Sie die höchstmögliche Zeilenanzahl und die maximale Anzahl von Zeichen pro Zeile miteinander malnehmen. Wenn Sie das Ergebnis noch mit der Anzahl der Punkte in der Zeichenmatrix multiplizieren, wissen Sie, wie viele Punkte auf dem Schirm im Höchstfall dargestellt werden müssen.

Sie erhalten eine Zahl zwischen 10 000 und 1 500 000. Beim üblichen 24-Zeilen-Format mit 80 Zeichen/Zeile und 7 × 9-Matrix sind es z. B. 120 960 Punkte. Jeder dieser Punkte wird vom Abtaststrahl 50mal pro Sekunde (nach US-Norm 60mal) überstrichen. Mit diesem Faktor ist die Bildpunktzahl also noch zu multiplizieren, um herauszubekommen, wie oft die Elektronik den Strahl innerhalb einer Sekunde auf- und abblenden muß. Dann ergeben sich zwischen 500 000 und 75 Millionen Schaltvorgänge je Sekunde, entsprechend einer Frequenz von 0,5–75 MHz (Megahertz). Im obigen Beispiel beträgt die Bildpunktfrequenz, d. h. die gesuchte Mindestbandbreite, 6,048 MHz (bei 50 Bildern/s).

Die Bandbreiten gängiger Monitore liegen zwischen 4 und 22 MHz, wobei die Preise mit der Bandbreite steigen – für fünfstellige Summen bekommen Sie auch 100-MHz-Monitore. Ist die Bandbreite für Ihren Rechner zu niedrig gewählt, erscheinen Texte verschwommen und kaum leserlich. Das Bild wird mit wachsender Bandbreite schärfer, aber eine zu hohe Auflösung kann speziell bei der Farbwiedergabe auch unerwünscht sein – das Bild „fällt auseinander", weil die einzelnen Pixel zu deutlich voneinander getrennt sind.

Nach einem ähnlichen Verfahren arbeitet ein Teil der Farbmonitore. Das Composite-Video-(FBAS = Farb-BAS)-Signal ist dabei wegen der zusätzlichen Farbinformation aber komplizierter aufgebaut und stark an den Verhältnissen bei Farbfernsehern orientiert. Es gibt im wesentlichen PAL-, SECAM- und NTSC-Systeme, entsprechend den weltweit gebräuchlichen Fernsehnormen. Dabei werden Helligkeit, „additive" Farbmischung (aus Rot, Grün und Blau) und Synchronsignale unterschiedlich verschlüsselt.

Farbmischungen

Besser ist es aber, dem Monitor Informationen getrennt zuzuführen – das geschieht bei den RGB(Rot-Grün-Blau)-Verfahren. Beim TTL-RGB kann jedes der drei Farbsignale nur die TTL(Transistor-Transistor-Logik)-Pegel 0 und 5 Volt annehmen; die Farben werden pixelweise ein- und ausgeschaltet. Das ermöglicht bei additiver Dreifarbenmischung einschließlich Schwarz (3 × „Aus") und Weiß (3 × „Ein") acht Farbtöne.

Beim Analog-RGB kann die Helligkeit der Grundfarben feinstufig verändert werden – das ermöglicht bei 16 Stufen schon $16^3 = 4096$ Farbtöne. Die Bezeichnung „analog" ist wegen der digitalen Abstufung hier eigentlich nicht so ganz korrekt.

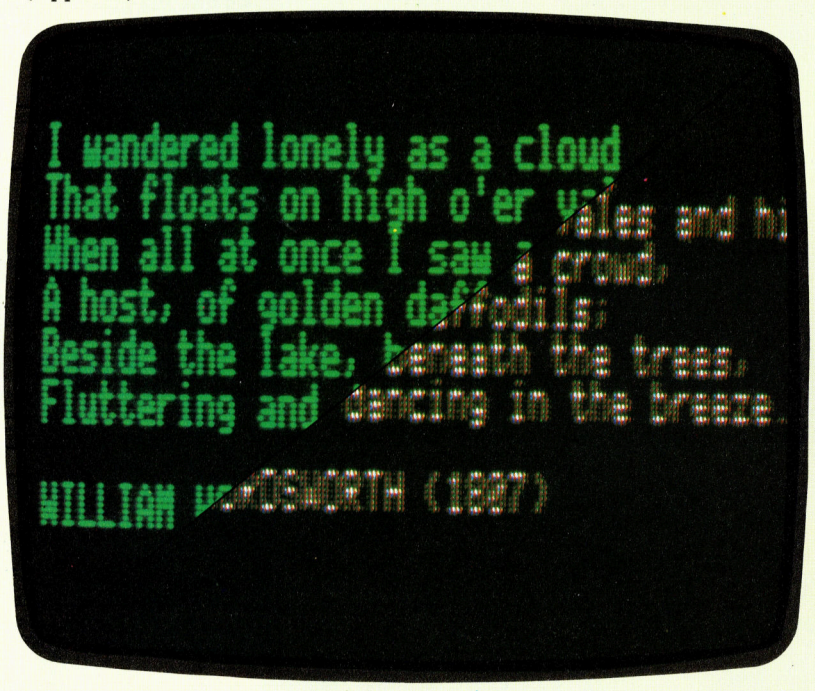
Die Preisskala beginnt bei etwa 250 Mark für ein 9-Zoll-Schwarzweißgerät; 12-Zoll-Monochrommonitore kosten zwischen 350 und 600 Mark. Für einen Farbmonitor zahlen Sie wegen der teureren Bildröhre 800 bis 1300 Mark, je nach Schirmdiagonale und Bandbreite. Sie können meist zwischen TTL- und Analog-Eingang wählen (bei geringer Preisdifferenz).

Interessant ist zum Beispiel der TM90PSN von JVC aufgrund seiner Vielseitigkeit. Für den kleinen 10-Zoll-Schirm und die durchschnittliche Bandbreite (d. h. begrenzte Auflösung) ist dies Gerät mit rund 1200 Mark nicht billig, aber es nimmt fast jedes Eingangssignal an, vom gewöhnlichen Monochrom-BAS über TTL- und Analog-RGB bis zu sämtlichen Farb-BAS-Signalen – PAL, SECAM, NTSC 3.58 und NTSC 4.43. Eine interne Selektionsschaltung erkennt die Art des Eingangssignals und wählt automatisch die entsprechende Betriebsart.

Voraussichtlich werden derartige Monitore zunehmend Verbreitung finden, da sie für Computer, Videorecorder, Bildplattenspieler und ähnliche technische Geräte universell verwendbar sind.

Klassische Perfektion

Das Foto zeigt den drastischen Unterschied in der Bildqualität zwischen einem rechnerangepaßten Monochrom-Monitor (Apple III) und einem qualitativ hochwertigen Heimfernseher. Zu Demonstrationszwecken einige berühmte Verse von William Wordsworth.



Bildschirmfarbe und Nachleuchten

Neben der Bandbreite spielt beim Kauf eines Monochrom-Monitors die Art des Phosphors eine wichtige Rolle – das ist das Leuchtstoffgemisch, mit dem die Schirminnenseite beschichtet ist. Das Bild entsteht, indem der auftreffende Elektronenstrahl diese Substanz zur Lichtemission anregt.

Kriterien sind die Farbe und die „Nachleuchtdauer", die angibt, wie lange die Lichtemission noch anhält, nachdem der Elektronenstrahl einen Bildpunkt passiert hat. Für das Abklingen werden selten Zahlenwerte genannt, man spricht nur von „langem" oder „kurzem" Nachleuchten.

Die Leuchtstoffe in Fernsehapparaten und in praktisch allen Farbmonitoren (auch in der Mehrzahl der Monochrom-Geräte) haben ein kurzes Nachleuchten. Häufig sind für den Betrachter aber auch Abklingzeiten bis an den Sekundenbereich von Vorteil. Das Bildschirmflimmern, das die Augen sehr anstrengt, wird dadurch merklich reduziert. Extrem langsam sind z. B. Radarschirme, denen der „Nachzieheffekt" eine Art Gedächtnis verleiht, ohne daß dafür eine komplizierte Speicherelektronik nötig wäre.

Wenn Sie am Bildschirm dagegen mit einem Lichtgriffel hantieren wollen, würde langes Nachleuchten das Gerät völlig unbrauchbar machen. Wenn das Leuchten nämlich nicht schnell genug abklingt, kann der Rechner die Position des Griffels nicht feststellen, weil der Lichtdetektor an der Spitze ständig einen hohen Fotostrom abgibt, und nicht nur dann, wenn der Abtaststrahl gerade den anvisierten Bildpunkt passiert. Der Griffel muß aber den genauen Anregungszeitpunkt melden, denn der Rechner bestimmt daraus anhand des Bildrasters die Position der Stiftspitze.

Bei Monitoren werden je nach Verwendungszweck Phosphore verschiedenster Farbe und Nachleuchtdauer eingesetzt. Weißleuchtende Schirme mit kurzem Nachleuchten sind überall billig zu bekommen, aber ein bestimmter Grünton ist fast ebenso gängig und angenehmer für das Auge, genau wie Bernstein. Blau finden Sie häufig bei den Terminals von Reisebüros und in der Flugabfertigung. Rot wird in Radarzentralen und ähnlichen Räumen benutzt, in denen die Dunkelanpassung des Auges nicht beeinträchtigt werden soll.

Fachwörter von A bis Z

Hi-Res Graphics = Hochauflösende Grafik
Hi-Res ist die Abkürzung für High Resolution (hohe Auflösung) – damit bezeichnen die meisten Hersteller die höchste Grafik-Auflösung, die bei ihrer Maschine vorgesehen ist. Aus wie vielen Bildpunkten (Pixeln) die Grafik im Hi-Res-Betrieb tatsächlich aufgebaut wird, hängt also vom Rechner ab. Mit der Anzahl der Pixel nimmt die Auflösung zu, und das Bild wird detailreicher und brillanter. Spezielle Computergrafik-Systeme erreichen eine horizontale und vertikale Auflösung von mehreren tausend Punkten.

Hollerith Code = Hollerith-Code
Herman Hollerith (1860–1929) entwickelte im Jahr 1888 die heute noch gebräuchliche Verschlüsselung von Buchstaben, Ziffern und Zeichen auf Lochkarten. Die Karte ist in 12 Zeilen mit je 80 Stanzpositionen aufgeteilt, wobei in jeder senkrechten Spalte durch ein bis drei Lochungen ein einzelnes Zeichen dargestellt wird. Die Information ist durch Kartenleser abtastbar und wird von „Tabelliermaschinen" mechanisch verarbeitet.

Für die Auswertung der amerikanischen Volkszählung entwickelte Herman Hollerith 1888 das Lochkartenverfahren, das sich später auch in Büros durchsetzte. Er gründete eine Gesellschaft zur Herstellung von Lochkartengeräten („Tabelliermaschinen"), aus der später die Firma IBM hervorging.

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.

Holographic Memory = Holografischer Speicher
Als neueste Errungenschaft bei den Kredit- und Scheckkarten gilt die Fälschungssicherung durch ein eingeprägtes „Hologramm" – ein Interferenzmuster, das fotografisch mit Hilfe eines Laserstrahls erzeugt wird. Die Banken und Kreditkartengesellschaften hoffen, damit Betrügern das Handwerk zu legen, weil das Hologramm sehr schwer nachzuahmen ist. Viel interessanter als diese spezielle Anwendung sind aber eigentlich die Perspektiven, die sich für den Einsatz holografischer Systeme als Massenspeicher am Computer abzeichnen.

In den Entwicklungslabors existieren holografische Speicher schon seit Anfang der siebziger Jahre. Die binäre Information wird dabei als Hologramm verschlüsselt auf einer Fotoschicht festgehalten. Durch Projektion des Hologramms mit einem schwachen Laser lassen sich die Ausgangsdaten rekonstruieren. Holografische Speicher sind noch mehr als Laserplatten gegen Umgebungseinflüsse wie Staub und Temperaturschwankungen unempfindlich, ebenso gegen oberflächliche Kratzer. Ende der siebziger Jahre gelang es bereits, 200 Millionen Bits auf einer Kunststoffkarte von 10 × 15 cm unterzubringen.

Host Computer = Host-Computer
Als Host-Computer wird bei Verbundsystemen der Computer bezeichnet, der für die Ablauforganisation zuständig ist. Dazu gehören die verschiedensten Aufgaben. In einem Local Area Network sind es im wesentlichen Server-Funktionen: Der Host-Computer stellt Dateien bereit, steuert die Datenübertragung und erledigt das Ausdrucken für alle anderen Systemkomponenten. Bei leistungsfähigeren Anlagen, vor allem beim Großrechner-Verbund, kann der Host-Computer die Verwaltung des Time-Sharing übernehmen. In einem hierarchischen Kommunikationssystem mit Rechnern unterschiedlicher Rangstufen dient ein Computer unter Umständen in der einen Ebene als Host-Computer und bearbeitet gleichzeitig auf einer anderen Ebene selbst Programme.

Human Factors Engineering = Menschengerechte Gestaltung
Weil der Rechner von Menschen bedient werden soll, erfordert die konstruktive Gestaltung (Engineering) die Berücksichtigung von auf dem Menschen beruhenden Einflußgrößen. Das „Human Factors Engineering" zielt darauf, perfekte Mensch-Maschine-Systeme zu schaffen, indem die traditionelle technikorientierte Vorgehensweise mit arbeitspsychologischen und sozialwissenschaftlichen Ansätzen zum Gesamtkonzept verschmolzen wird.

Computer-Neulinge haben vielfach Hemmungen, die der Gestalter mit einem benutzerfreundlichen Konzept überwinden kann. Derartige Schritte sind auf der Softwareseite zum Beispiel Menütechnik und Dialogsprachen, auf der Hardwareseite „Benutzerschnittstellen" wie die Maus als Eingabegerät oder der berührungsempfindliche Bildschirm. All dies sind aussichtsreiche Versuche, die Furcht vor dem Computer abzubauen. Auch Gesichtspunkte wie die Anordnung von Arbeitsflächen und die Aufstellung der Rechner fließen in eine Gesamtkonzeption ein.

Bildnachweise
1401, 1408, 1409, 1410, 1421, 1422: Kevin Jones
1407, 1411, 1428: Ian McKinnell
1412: Liz Heaney
1413, 1426, 1427: Chris Stevens
1417, 1424, 1425: Liz Dixon

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

computer kurs

Allzweckgerät

Der neue CPC 6128 der Firma Schneider hat einen größeren Speicher und soll einen weiteren Teil des Computer-Marktes erobern.

Musterhaft

Weiter geht es mit der Muster-Erkennung in unserer Serie „Künstliche Intelligenz". Diesmal zusätzlich ein BASIC-Programm zur Lösung dieser Aufgabe.

Musikpaket

Für den Acorn B und Commodore C 64 gibt es „The Music System".

Vielseitig

Plotter, Maus und Bodenroboter – alle Funktionen will der „Penman Plotter" in sich vereinigen.