Einsteigen - Verstehen - Beherrschen

DM 3,80 öS 30 sfr 3,80

computer kurs

Ein wöchentliches Sammelwerk

Heft 78

Bebilderte Tabellen

HOPE macht Hoffnung

Modems über alles

Service: Alles in Ordnung!

computer kurs

Heft 78

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

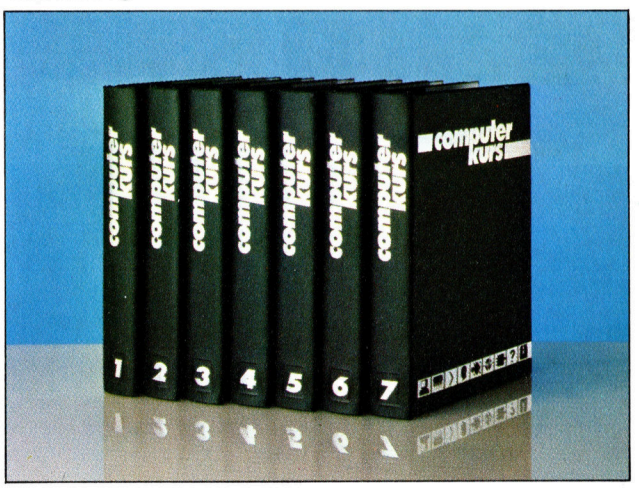
Schweiz: Der Sammelordner kostet sfr 15. Bitte wenden Sie sich an Ihren Kiosk; dort werden Sie jederzeit die gewünschten Exemplare erhalten.

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), Holger Neuhaus, Peter Aldick, 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

Motorolas Meisterstück

Der erweiterte Adreßbereich des 68000-Microprozessors reicht an viele Maschinen der Groß-EDV heran. In unserer Serie über den 68000 zeigen wir zunächst die Bereiche der Prozessorarchitektur, die für die Programmierung wichtig sind.

Durch die 68000-Serie erhalten die modernen Micros die Möglichkeiten von Minicomputern. Viele Hersteller haben diesen Vorteil erkannt.

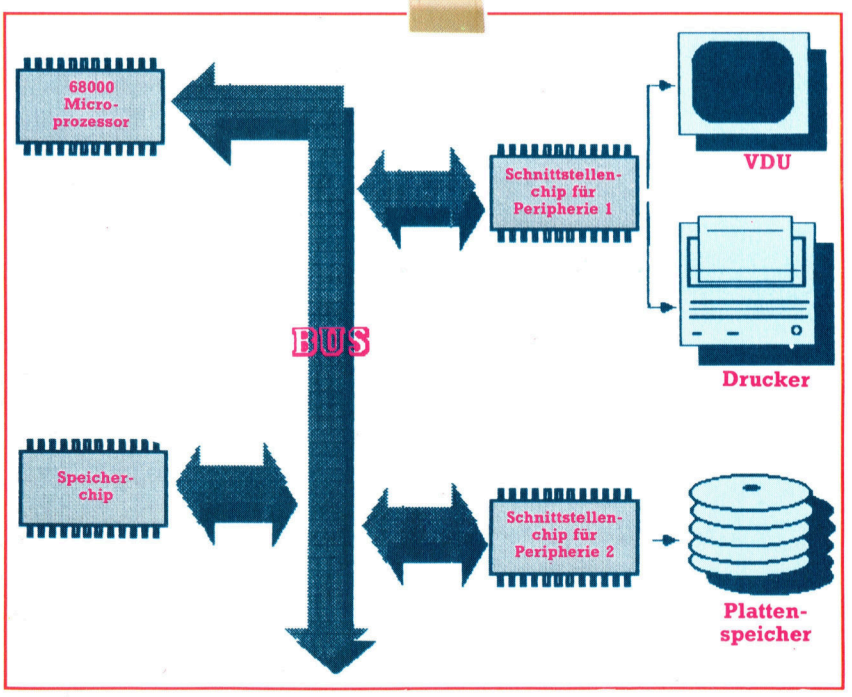
Atari 520 ST

Apple Macintosh

Sinclair QL

Apple Lisa

Die Befehle und Daten eines Einplatinencomputers werden über einen Bus aus dem Speicher geholt und dann verarbeitet. Danach wird das Ergebnis wieder auf den Bus gesetzt, der es zu Peripheriegeräten sendet.

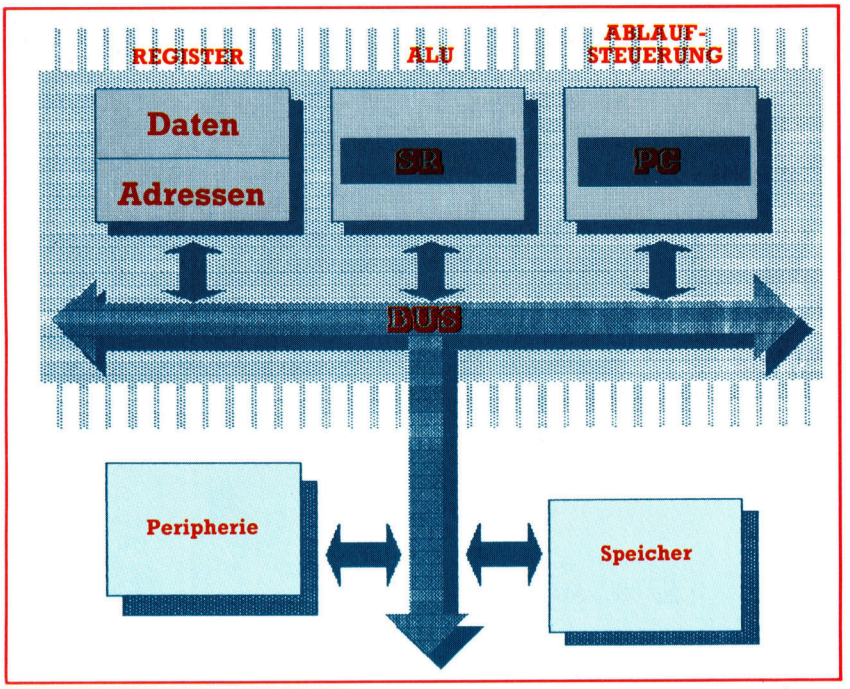


Der 68000 entwickelte sich aus der erfolgreichen 6800-Serie von Motorola, zu der auch der 6809-Microprozessor gehört. Mit der Familie der 68000-Chips baute Motorola seinen Erfolg weiter aus. Bei dem Prozessor stand die neue VLSI-Technik (Very Large Scale Integration) Pate, mit der mehr als 100 000 Logikelemente auf einem etwa fünf Zentimeter langen und mit 64 Kontakten versehenen Chip untergebracht wurden. Die Adressiermöglichkeiten reichen an die des DEC PDP-11 Minicomputers heran, wobei der 68000 den Komfort und die Flexibilität eines Befehlssatzes der Groß-EDV bietet.

Die Adreß- und Befehlsmöglichkeiten werden wir uns in späteren Folgen genauer ansehen. Zunächst wollen wir uns einen Überblick über die Architektur des 68000 verschaffen, um ein Modell für unsere Programmabläufe zu haben. Da wir nicht auf alle Aspekte des Chips eingehen können, beschränkt sich unser Modell nur auf die Bauteile, die für unsere Beispielprogramme wichtig sind. So brauchen wir beispielsweise für die Addition zweier Zahlen nicht die Arbeitsweise der Flip-Flop-Elemente zu verstehen, sondern müssen uns lediglich mit der Registergröße und dem Befehl ADD beschäftigen. In unserem Modell beschränken wir uns daher auf die Funktionen, mit denen der Microcomputer programmiert wird.

Einplatinensystem

Wenn wir die Funktionen der Kontakte beim 68000 untersuchen, stellen wir fest, daß damit digitale Busse gesteuert werden, über die der Prozessor mit seiner „elektronischen Umgebung" Daten austauscht. Mit den Bussen läßt sich ein vollständiges Microcomputersystem aufbauen. Unser erstes Bild zeigt ein typisches Einplatinensystem, in dem sich der Prozessor den nächsten Befehl aus dem Speicher holt, ihn ausführt und schließlich – falls gewünscht – ein Zeichen zum Schnittstellenchip sendet,



Der innere Aufbau des 68000 läßt sich in drei Hauptbereiche unterteilen: die Daten- und Adreßregister, die Arithmetik- und Logikeinheit und die Ablaufsteuerung. Jede Einheit kann über einen digitalen Datenbus Informationen austauschen.

der das Zeichen dann direkt auf den Bildschirm überträgt.

Dieser Hardwareaufbau ist Grundlage unseres Modells. Dabei ist zwar der zentrale Datenbus die wichtigste Komponente, doch spielen auch die internen Register des Microprozessors eine wesentliche Rolle.

Unser zweites Bild zeigt den Chip aus der Sicht des Programmierers. Wir werden uns die drei logisch voneinander getrennten Bereiche einmal genauer ansehen:

Die Register sind – wie auch in anderen Prozessoren – Arrays mit Speicherstellen, die zur Kurzspeicherung von Daten oder Zwischenergebnissen eingesetzt werden. Der 68000 verfügt über acht Daten- und acht Adreßregister, die je eine Breite von 32 Bit haben. Durch die Aufteilung in zwei Bereiche werden Maschinenabläufe und Programmierung doch schon sehr vereinfacht.

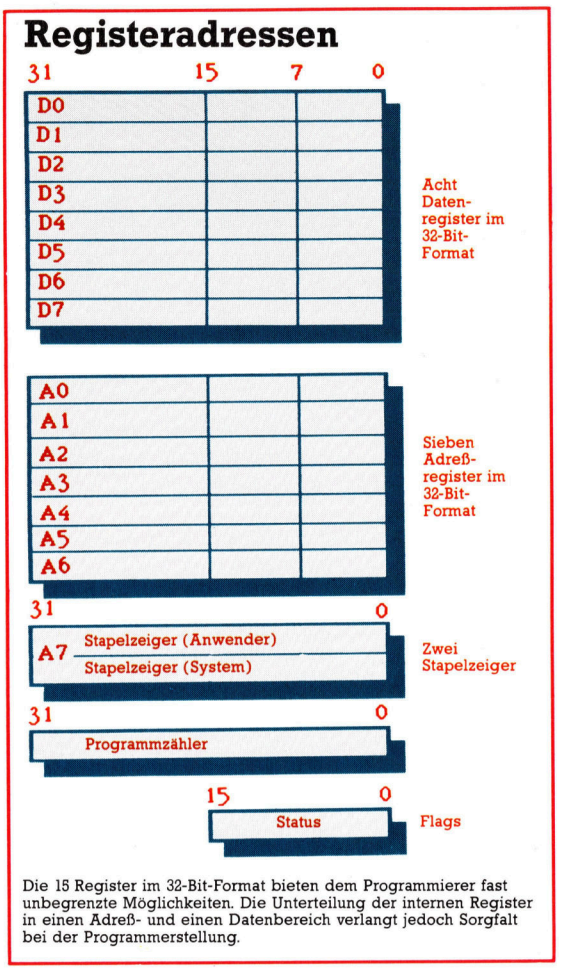
Wenn Sie beispielsweise den Inhalt einer Speicherstelle (deren Adresse sich bereits in einem der internen Register befindet) laden wollen, um einen arithmetischen Ablauf durchzuführen, muß zunächst der Pointer (oder die Speicheradresse) in ein Adreßregister geladen werden. Daraufhin können die Daten in einem Datenregister gespeichert werden. In der Assemblersprache wird daraus:

```
MOVE (A2),D4
```

MOVE ist der Befehl zum Übertragen (oder Kopieren) von Daten, A2 gibt als Quellenadresse den Inhalt des Adreßregisters 2 an, während D4 das Datenregister 4 als Ziel festlegt. Wenn die Adresse der Quelldaten in D4 läge, könnten wir jedoch nicht einfach

```
MOVE (D4),A2
```

schreiben, da eine Adressierung von Quelle und Ziel beim Befehl MOVE nicht zulässig ist.



Die 15 Register im 32-Bit-Format bieten dem Programmierer fast unbegrenzte Möglichkeiten. Die Unterteilung der internen Register in einen Adreß- und einen Datenbereich verlangt jedoch Sorgfalt bei der Programmerstellung.

Wichtig ist weiterhin, daß speziell die Datenregister fünf voneinander verschiedene Datentypen annehmen:

- Bit – eine Binärstelle
- BCD-Stelle – ein Nybble oder Halbbyte
- Byte – ein Acht-Bit-Zeichen
- Word – ein 16-Bit-Computerwort
- Long Word – ein 32-Bit-Computerwort.

Auf BCD-Stellen und Langwörter gehen wir später noch genauer ein, für den Augenblick wollen wir sie einfach als größere Dateneinheiten ansehen.

Das Befehlsattribut

Bei den meisten Befehlen kann der Datentyp als Befehlsattribut angegeben werden. Wenn Sie beispielsweise die Bits 0 bis 7 von D1 auf D2 kopieren wollen, schreiben Sie

```
MOVE.B D1,D2
```

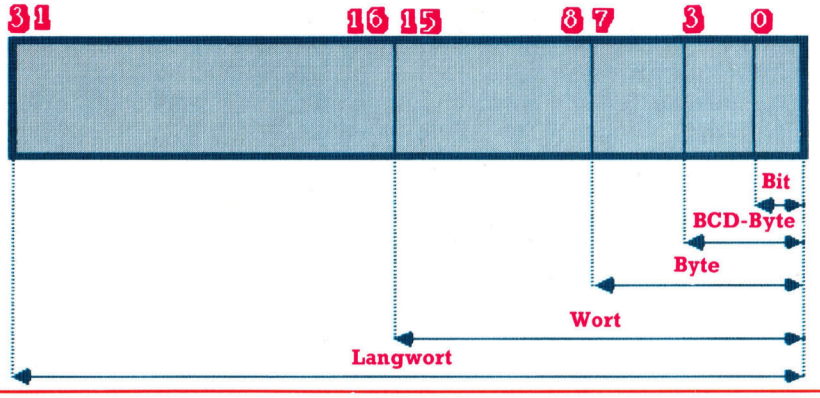
Alle 32 Bits lassen sich mit folgendem Befehl kopieren:

```
MOVE.L D1,D2
```

Das Attribut (oder der Code für die Datenlänge) eines 16-Bit-Wortes lautet .W. Dies ist die Standardeinstellung, wenn kein anderer Code angegeben wird.

Wortlängen

Der 68000 kann über Befehlserweiterungen Wörter mit fünf verschiedenen Längen adressieren und damit Daten in Gruppen von einem bis 32 Bits verarbeiten. Diese Möglichkeit gab es bei den älteren Acht-Bit-Chips nicht.



Noch ein Wort zum Registersatz. Das Adreßregister A7 wird als Pointer für den Systemstack eingesetzt – seien Sie daher beim Gebrauch dieses Registers äußerst vorsichtig.

Die ALU (Arithmetik- und Logikeinheit) ist der zweite große Bereich unseres Modells. Dieser Teil führt die arithmetischen und logischen Vorgänge aus und speichert das Ergebnis im Zieloperanden. So addiert die ALU bei folgendem Befehl D1 und D2 und stellt das Ergebnis in D2:

```
ADD D1,D2
```

Auch hier können Sie im Befehl die Datenlänge angeben und nur die niederwertigen Bytes addieren:

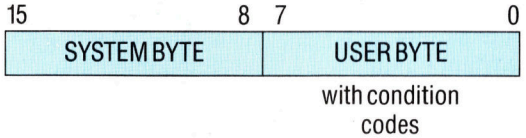
```
ADD.B D1,D2
```

In unserem Modell ist der ALU das Register SR oder Status Register beigeordnet, das Informationen über das Ergebnis des vorangegangenen Befehls liefert. So etwas wird beispielsweise für eine bedingte Verzweigung gebraucht, die vom Ergebnis des vorigen Befehls abhängt.

```
ADD D1,D2     D1 mit D2 addieren
BVS OFLOW     auf Überlauf prüfen
MOVE D1,D3    D1 auf D3 kopieren
```

Hier wird entweder zum Label OFLOW verzweigt, wenn das Überlaufbit in SR gesetzt ist, oder D1 auf D3 kopiert, wenn es nicht gesetzt wurde. Der Befehl BVS (Verzweigung bei Überlauf, englisch: „Branch if Overflow Set") prüft, ob ein Überlauf stattgefunden hat (oder Bit V des Statusregisters gesetzt ist). In diesem Beispiel könnte das Ergebnis des ADD-Befehls das Setzen des V-Bits verursachen. Überläufe entstehen, wenn das Ergebnis eines arithmetischen Vorgangs nicht mehr in die Wortlänge der Operanden paßt.

Das Statusregister besteht aus 16 Bits, und seine Bytes werden vom System wie folgt aufgeteilt:



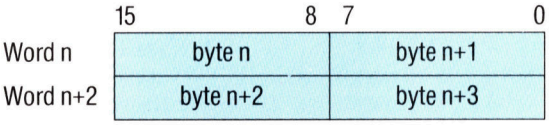
Die einzelnen Bits zeigen das Ergebnis der vorangegangenen Befehle an – wir gehen später darauf genauer ein.

Adressierung

Der Bereich für Ablaufsteuerung enthält den Programmzähler und ein Register, in dem die nächste Adresse steht, die aus dem Speicher geholt werden soll. Nach dem Laden wird ein Befehl erst einmal decodiert, um festzustellen, welche Art Ablauf die ALU ausführen soll, und wo die Quell- und Zieloperanden liegen.

Der Programmzähler ist zwar 32 Bit lang, aber nur mit 24 Kontakten an den Bus angeschlossen. Doch selbst mit 24 Bits steht ein riesiger Adreßbereich bis zu Hex FFFFFF Bytes zur Verfügung. Jede Hexadezimalstelle entspricht vier binären Bits, so daß die 24 Bits bis zu 16 777 216 Bytes direkt adressieren können. Da jedoch alle Befehle nur mit geraden Adressen beginnen dürfen, kommt ein Adreßbereich von 8 388 608 Computerwörtern der Wirklichkeit näher.

Interessant ist auch die Anordnung der Daten im Speicher. So lassen sich mit dem 68000 einzelne Bytes ansprechen, und die Adressiermethode unterscheidet sich erheblich von den bereits besprochenen Adressierungsarten. Die folgende Zeichnung macht deutlich, wie die Speicheradressierung des 68000 das höchstwertige Byte eines Wortes liefert.



Einige Befehle wirken direkt auf den Programmzähler und veranlassen so eine Programmverzweigung; entweder ohne Bedingung wie bei BRA ZURUECK (BRA – englisch „Branch Always" – bedeutet „Unbedingte Verzweigung" auf die Adresse, die das Label ZURUECK darstellt) oder bedingt wie in unserem letzten Beispiel BVS OFLOW, in dem die Verzweigung von dem Setzen des V-Bits im PS abhängig ist. Bei jeder Verzweigung oder Änderung des Programmflusses wird der Pro-

grammzähler auf die Adresse des nächsten Befehls gesetzt, der ausgeführt werden soll.

Computerhersteller preisen oft die große Zahl von Befehlen an, die auf ihren Maschinen zur Verfügung stehen. Diese Befehlsvielfalt kann jedoch sehr verwirren, wenn für jeden Operandentyp oder Adreßmodus ein anderer Befehl verwandt werden muß. Dies trifft besonders auf die älteren Acht-Bit-Prozessoren zu. Der 8085 besitzt beispielsweise 63 verschiedene MOV-Befehle, die die Register auf unterschiedliche Weise ansprechen.

Viel wichtiger als die Zahl der Befehlscodes ist der damit erfaßte Bereich, die Vielfalt der unterstützten Datenobjekte und die Flexibilität der Adreßarten.

Flexibler Befehlssatz

Motorola bietet einen außerordentlich flexiblen Befehlssatz für Datenobjekte. Die meisten Befehle sind für die Adressierung von Bytes, Wörtern und Langwörtern ausgelegt. Sie können binäre Multiplikation, Division und BCD-Arithmetik ausführen und verfügen über die übliche Befehlspalette für Logikabläufe, Programmsteuerung und das Kopieren von Daten.

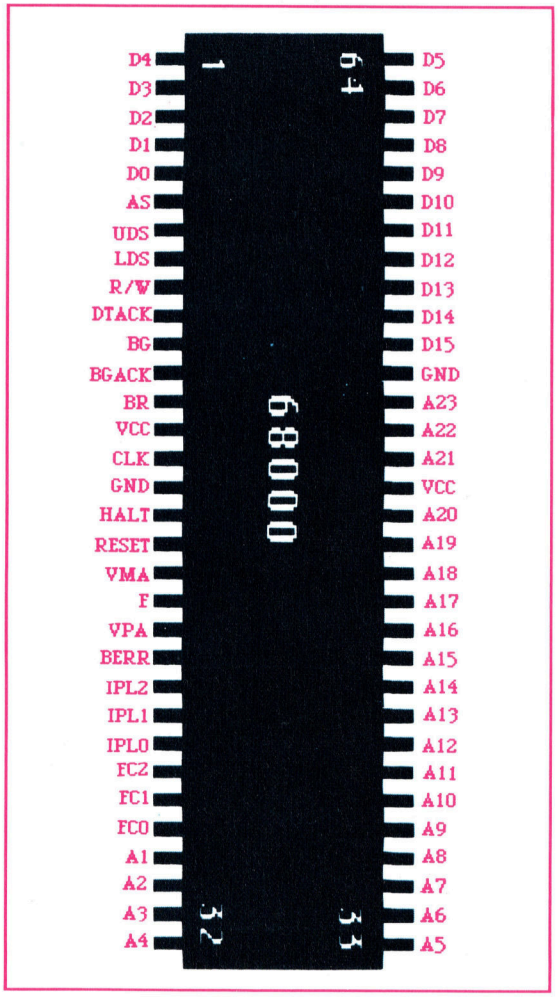
Die vielen Adreßarten lassen sich jedoch nicht in allen Befehlen einsetzen. enn der Zieloperand des MOVE-Befehls beispielsweise ein Adreßregister ist, können Sie nicht einfach MOVE D3,A6 schreiben, sondern müssen MOVEA oder LEA (MOVE A D3,A6) verwenden. Hier wurden einige Adreßmöglichkeiten zugunsten einer großen Zahl der möglichen Datenobjekte geopfert.

Im Vergleich zu den Befehlssätzen der Acht-Bit-Micros (und selbst einigen Computern der Mini- und Groß-EDV) hat der 68000 einen ungewöhnlich breiten Anwendungsbereich, kann eine Vielzahl von Datenobjekten verarbeiten, enttäuscht aber mit der wenig flexiblen Adressierung. Dieser Nachteil wird durch die Kompaktheit einiger Befehle (auch „Quick"-Befehle genannt) teilweise wieder ausgeglichen. Bei Quick-Befehlen ist der gesamte Ablauf in einem Maschinenwort enthalten. Wenn Sie zum Beispiel die Zahl 25 in ein Datenregister laden wollen, genügt

```
MOVEQ #25,D3
```

wonach 25 in D3 gespeichert ist. Der gesamte Befehl inklusive der Konstanten 25 belegt nur ein Wort.

In der nächsten Folge sehen wir uns Einsatz und Struktur der Adreßarten genauer an und geben einige Programmbeispiele.



Das Bild zeigt die Kontakte des 68000. Beachten Sie die 24 Adreßleitungen und die 15 Datenleitungen. Im Gegensatz zu seinem nahen „Verwandten", dem 68008 (der nur einen Acht-Bit-Adreßbus besitzt), kann der 68000 Speicheradressen von Hex 000000 bis FFFFFF ansprechen. Der nächstgrößere Prozessor dieser Serie – der 68020 – verfügt über einen vollständigen 32-Bit-Bus. Der 68000 ist das perfekte Beispiel eines modernen Microprozessors – der Chip mit seinen mehr als 100 000 Logikschaltungen besitzt die Verarbeitungsmöglichkeiten eines Minicomputers.

Im rechten Moment

Der 16-Bit-Prozessor 68000 erschien genau zu dem Zeitpunkt, als viele Hersteller eine neue Maschinengeneration entwarfen. Unterschiedliche Versionen von ihm wurden in den Sinclair QL, den Apple Macintosh und die ST-Reihe von Atari eingebaut.

Der Prozessor ist einfach zu programmieren: In den Chip wurden siebzehn 32-Bit-Register, ein 16-Bit-Datenbus, ein 24-Bit-Adreßbus und der Befehlssatz eingebaut.

Trotz der beeindruckenden Fähigkeiten dieser Geräte ist es nicht ganz einfach, das Potential des 68000 voll zu entfalten. Der 68000 ist der restlichen Chip-Technologie so weit voraus, daß viele Computerhersteller überhaupt nur einen Bruchteil seiner eigentlichen Fähigkeiten einsetzen. Ein extremes Beispiel ist der Sinclair QL, der mit der 68008-Version des Chips und damit auch nur mit einem Acht-Bit-Datenbus arbeitet. Viele Anwendungen des QL sind deshalb kaum schneller als bei den Acht-Bit-Vorläufern.

Die Zukunft ist der 68010, der den 16-Bit-Adreßbus beibehält, aber einen virtuellen Speicher bietet. Das bedeutet, daß Programme mit logischen (virtuellen) Adressen arbeiten können, ohne sich darum kümmern zu müssen, ob diese Adressen auch physisch vorhanden sind. Damit läßt sich der gesamte RAM-Bereich einsetzen, ohne in das Gerät eingebaut sein zu müssen. Ein weiterer Chip von Motorola, der 68020, wird als 32-Bit-Computer angepriesen, der 97 % der Fähigkeiten einer Groß-EDV-Anlage besitzt.

Zellaktivierung

Den generellen Aufbau unseres Tabellenkalkulationsprogrammes besprachen wir bereits. Wir lenken unsere Aufmerksamkeit nun auf die Programmierung der Bildschirmdarstellung für den Acorn B, den Schneider CPC und den Sinclair Spectrum.

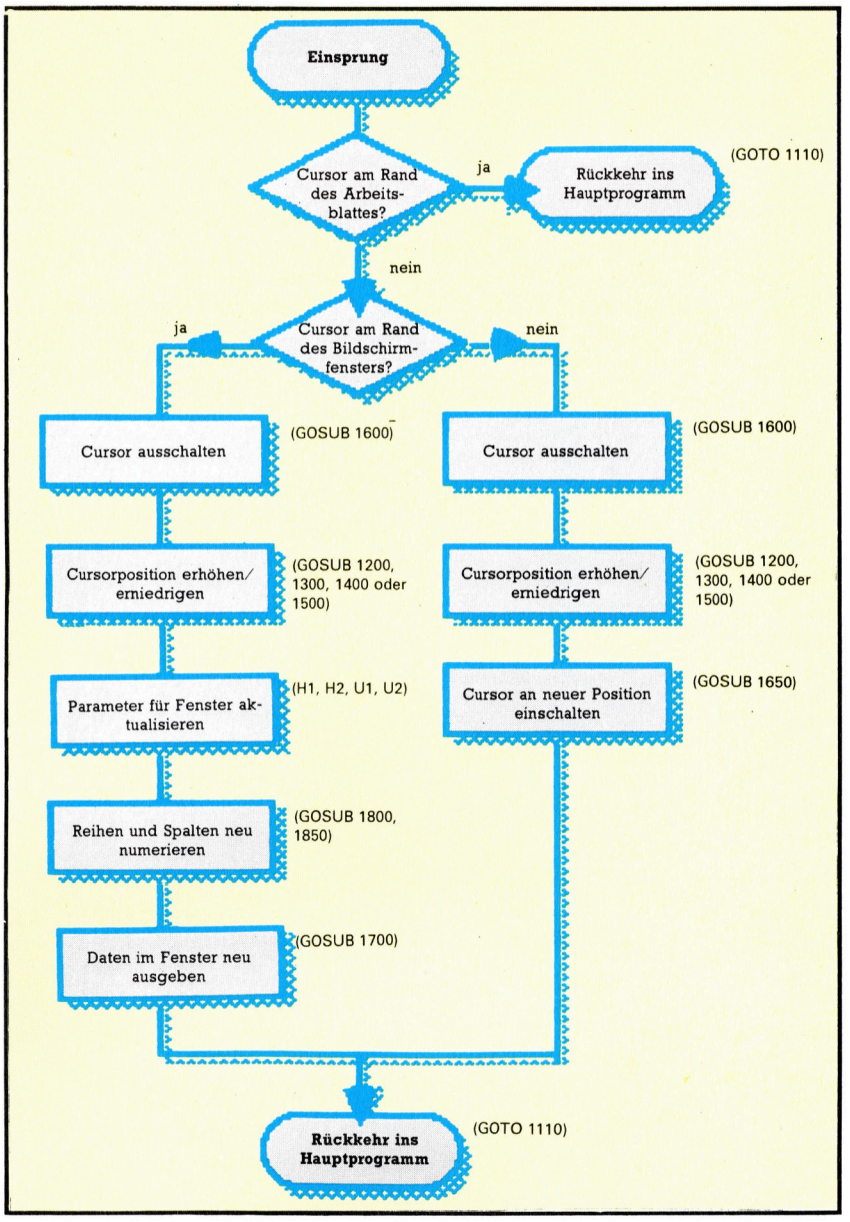
Die Mikros, für die das Kalkulationsprogramm geschrieben wurde, produzieren ihre Bildschirmgrafiken auf unterschiedliche Weise. Wir stellen daher die Grafikroutinen für jeden Rechner gesondert vor. (Die Routinen für den Commodore 64 finden Sie auf Seite 2132.) Ihre Hauptfunktionen sind die Ausgabe der Tabellenmatrix und die Steuerung des Cursors innerhalb der Matrix.

Die Routinen zur Cursorsteuerung enthalten Sektionen, die den Cursor nach allen vier Seiten durch das Kalkulationsschema bewegen. Auf dem Bildschirm ist immer nur ein Teil der Matrix sichtbar. Die Steuerungsroutinen müssen also auch die Bewegung des Bildausschnitts über das Arbeitsblatt übernehmen. Die Zeilen 1000 bis 1080 bilden eine Unterroutine zum Darstellen des Kalkulationsblattes. Der bei Zeile 1100 beginnende Abschnitt ist ein wichtiger Kontrollteil des Hauptprogramms. Er prüft, ob die Tastatur betätigt wurde. Mit ihrer Hilfe bewegen Sie den Cursor und wählen Funktionen an, etwa die Eingabe einer Berechnungsformel in eine bestimmte Zelle. Anschließend wird die entsprechende Unterroutine aufgerufen.

Den Cursor bewegen

Die meisten Cursor-Funktionen sind Bestandteile später folgender Programmabschnitte. Der Versuch, eine dieser Funktionen bereits jetzt anzuwählen, führt zu einem Absturz des Programms. Mit den Routinen aus diesem Teil können Sie immerhin schon den Cursor über den Bildschirm bewegen. In den Zeilen 1200 bis 1600 stehen die Unterprogramme zur Steuerung nach rechts, links, oben und unten. Sie sind prinzipiell für jede Bewegungsrichtung gleich aufgebaut. Wir betrachten lediglich das erste, um einen Eindruck der Arbeitsweise aller zu bekommen.

Nach Betätigen der Steuertaste „Cursor rechts" verzweigt das Hauptprogramm zur Subroutine bei Zeile 1200. Zeile 1210 prüft, ob der Cursor bereits den rechten Rand des Arbeitsblatts erreicht hat. Dazu wird getestet, ob die X-Koordinate bereits den maximalen Wert 15 erreicht hat. Bei Gleichheit der Werte wird die Subroutine abgebrochen und das Hauptprogramm weiter abgearbeitet. Andernfalls folgt ein Vergleich, ob sich der Cursor am Rand des Bildschirmfensters befindet. Die Variablen H1 und H2 bezeichnen die äußersten noch innerhalb des Fensters sichtbaren horizontalen Zellen. Wenn etwa H1 gleich 2 und H2 gleich 6 ist, sind die Spalten 2 bis 6 der Matrix auf dem Bildschirm sichtbar. Übereinstimmende Werte in X und H2 lösen folgende Aktionen aus: Der Cursor wird ausgeschaltet, der Wert in X um Eins erhöht, ebenso die Werte in H1 und H2. Die Subroutine bei 1800 schreibt



Unser Flußdiagramm zeigt, wie das Kalkulationsprogramm den Cursor auf dem Arbeitsblatt steuert. Die Routinen bewegen den sichtbaren Ausschnitt auf den nächsten Bereich, wenn der Cursor den oberen, unteren, linken oder rechten Rand des Fensters erreicht.

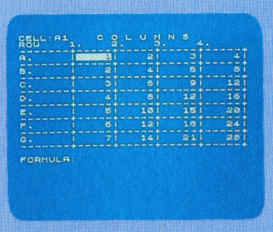
neue Zeilen- und Spaltennummern an die Ränder der Matrix, und die neuen Daten der jetzt sichtbaren Zellen werden auf dem Bildschirm ausgegeben. Schließlich wird der Cursor wieder eingeschaltet.

Die Unterprogramme bei Zeile 1600 und 1650 schalten den Arbeitsblatt-Cursor ein und aus. Sie sind wieder für jeden Rechnertyp gesondert erstellt. Alle Versionen zeigen den Cursor durch Invertieren der Farben in der angegebenen Zelle.

Auf dem Spectrum werden die Vorder- und Hintergrundfarbe zu jeder Zeichenposition des Bildschirms durch ein Byte in einer Tabelle, der „Attribute Map", bestimmt. Die unteren drei Bits geben die Vordergrundfarbe (INK) an, die Bits drei bis fünf die Hintergrundfarbe (PAPER). Um eine Zelle zu invertieren, müssen wir die ihr zugehörigen Bytes in der Attributtabelle lokalisieren und die Bit-Trios miteinander vertauschen.

Die Versionen für den Acorn B und den Schneider CPC arbeiten etwas komplizierter, da auf diesen Rechnern der Inhalt einer Zelle mit den veränderten Farbinformationen neu ausgegeben werden muß. Die Feldvariable M(y,x) enthält die Werte der Berechnungszellen des Arbeitsblatts. Die Routinen ermitteln den derzeit gültigen Inhalt der zu invertierenden Zelle aus diesem Array und wandeln ihn in eine PRINT-reife Zeichenkette um. Auf dem Acorn B vertauschen wir mit dem Befehl COLOUR die Farben. Beim Schneider CPC genügt das Steuerzeichen CHR$(24), um beim Print-Befehl die Farbwerte für PEN und PAPER auszutauschen.

Sinclair Spectrum

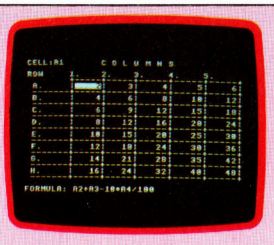


```
 100 GO SUB 3000
 110 GO SUB 1000
 120 GO SUB 1700
 130 GO SUB 1100
 999 STOP
1000 BORDER 1: PAPER 1: CLS : IN
K 7
1005 PRINT "            C O L U M
 N S"
1007 PRINT "ROW    1.    2.    3
.    4."
1010 PRINT "-------+-----+-----+
-----+-----+"
1020 FOR C=1 TO 6
1030 PRINT CHR$ (C+64);".      |
    |     |     |     |"
1040 PRINT "-------+-----+-----+
-----+-----+"
1050 NEXT C
1060 PRINT CHR$ (C+64);".      |
    |     |     |     |"
1070 PRINT "-------+-----+-----+
-----+-----+"
1080 RETURN
1100 PRINT AT 0,0;"CELL:";CHR$ (
Y+64);STR$ (X)
1110 LET A$=INKEY$: IF A$="" THE
N  GO TO 1110
1120 IF A$="8" THEN  GO TO 1200:
 REM MOVE RIGHT
1130 IF A$="5" THEN  GO TO 1300:
 REM MOVE LEFT
1140 IF A$="6" THEN  GO TO 1400:
 REM MOVE DOWN
1150 IF A$="7" THEN  GO TO 1500:
 REM MOVE UP
1155 IF A$="C" THEN  GO SUB 2300
: REM CALCULATE SHEET
1160 IF A$="F" THEN  GO SUB 2000
: REM INPUT FORMULA
1165 IF A$="E" THEN  GO SUB 2100
: REM ENTER NUMERIC DATA IN CELL
1168 IF A$="H" THEN  GO TO 6000:
 REM PRINT HELP SCREEN
1170 IF A$="S" THEN  GO SUB 5150
: REM STORE CURRENT SHEET IN MEM
ORY
1172 IF A$="G" THEN  GO SUB 5100
: REM GET PREVIOUS SHEET
1174 IF A$="Z" THEN  GO SUB 5000
: REM CLEAR CURRENT SHEET
1176 IF A$="R" THEN  GO SUB 5700
: REM REPLICATE FORMULA
1178 IF A$="T" THEN  GO SUB 5200
: REM TAB TO NEW CELL
1180 IF A$="D" THEN  GO SUB 7000
: REM LOAD/SAVE DATA/FORMULAS
1190 GO TO 1110
1200 REM ***** MOVE RIGHT *****
1210 IF X=15 THEN  GO TO 1100
1220 IF X=H2 THEN  GO SUB 1600:
LET X=X+1: LET H1=H1+1: LET H2=H
2+1: GO TO 1270
1230 GO SUB 1600: LET X=X+1: GO
SUB 1650: GO TO 1100
1270 GO SUB 1800: GO SUB 1700: G
O TO 1100
1300 REM ***** MOVE LEFT *****
1310 IF X=1 THEN  GO TO 1100
1320 IF X=H1 THEN  GO SUB 1600:
LET X=X-1: LET H1=H1-1: LET H2=H
2-1: GO TO 1370
1330 GO SUB 1600: LET X=X-1: GO
SUB 1650: GO TO 1100
1370 GO SUB 1800: GO SUB 1700: G
O TO 1100
1400 REM ****** MOVE DOWN *****B
1410 IF Y=15 THEN  GO TO 1100
1420 IF Y=V2 THEN  GO SUB 1600:
LET Y=Y+1: LET V1=V1+1: LET V2=V
2+1: GO TO 1470
1430 GO SUB 1600: LET Y=Y+1: GO
SUB 1650: GO TO 1100
1470 GO SUB 1850: GO SUB 1700: G
O TO 1100
1500 REM ***** MOVE UP *******
1510 IF Y=1 THEN  GO TO 1100
1520 IF Y=V1 THEN  GO SUB 1600:
LET Y=Y-1: LET V1=V1-1: LET V2=V
2-1: GO TO 1570
1530 GO SUB 1600: LET Y=Y-1: GO
SUB 1650: GO TO 1100
1570 GO SUB 1850: GO SUB 1700: G
O TO 1100
1600 REM ** TURN CURSOR OFF ***
1610 LET CU=22528+32*(V(Y+1-V1))
+H(X+1-H1)
1615 POKE CU,15: POKE CU+1,15: P
OKE CU+2,15
1620 POKE CU+3,15: POKE CU+4,15:
 RETURN
1650 REM *** TURN CURSOR ON ***
1660 LET CU=22528+32*(V(Y+1-V1))
+H(X+1-H1)
1665 POKE CU,56: POKE CU+1,56: P
OKE CU+2,56
1670 POKE CU+3,56: POKE CU+4,56
1690 GO SUB 1900: RETURN
1700 REM ** PRINT DATA IN SHEET
1710 FOR I=0 TO 6
1720 FOR J=0 TO 3
1730 LET P$=STR$ (M(I+V1,J+H1))
1740 PRINT AT V(I+1),H(J+1);"
  "
1745 PRINT AT V(I+1),(H(J+1)+5-L
EN (P$));P$
1750 NEXT J: NEXT I
1760 GO SUB 1650: RETURN
1800 REM *** PRINT COLUMN NOS **
1810 FOR I=H1 TO H2
1820 PRINT AT 1,7+6*(I-H1);I;".
"
1830 NEXT I
1840 RETURN
1850 REM ** PRINT ROW NOS ****
1870 FOR C=V1 TO V2
1880 PRINT AT 2*(C-V1)+3,0;CHR$
(C+64);"."
1890 NEXT C
1895 RETURN
1900 REM * FORMULA CURRENT CELL
1920 LET D$=F$((Y-1)*15+X,1 TO )
1930 PRINT AT 18,0;"FORMULA:
                "
1940 PRINT AT 18,0;"FORMULA: ";D
$
1945 RETURN
```

Routinen zum Feldaufbau

```
3000 REM ************************
3001 REM * SETUP ARRAYS         *
3002 REM ************************
3010 DIM H(4): DIM V(7): DIM S(2
0): DIM S$(20): DIM E$(20): DIM
G$(20): DIM C(20)
3020 FOR C=0 TO 3
3030 LET H(C+1)=6*C+8: REM CALC
X-POS
3040 NEXT C
3050 FOR C=1 TO 7
3060 LET V(C)=2*C+1: REM CALC Y-
POS
3070 NEXT C
3075 LET X=1: LET Y=1
3080 LET H1=X: LET H2=X+3: LET V
1=1: LET V2=V1+6
3090 REM ************************
3091 REM * VALUE ARRAY          *
3092 REM ************************
3100 DIM M(15,15): DIM N(15,15)
3110 FOR I=1 TO 15
3120 FOR J=1 TO 15
3130 LET M(I,J)=I*J
3140 NEXT J: NEXT I
3150 DIM F$(255,20): DIM G$(20):
 RETURN
```

Acorn B

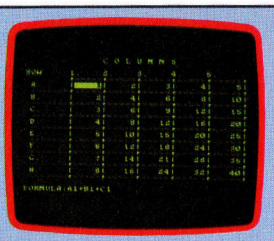


```
   10REM **** BBC SPREADSHEET ****
   40 MODE 4
   50 *FX4,1
   60 LET CO$=CHR$(30):CL$=CHR$(8):CR$=C
HR$(9):CU$=CHR$(11):CD$=CHR$(10)
   70 REM VDU 23,1,0;0;0;0;
  100 GOSUB 3000:REM SETUP ARRAYS & SCRE
EN VARIABLE
  110 GOSUB 1000:REM PRINT SCREEN
  120 GOSUB 1700:REM PRINT DATA WINDOW O
N SCREEN
  130 GOTO 1100:REM MAIN KEYBOARD ROUTIN
E
  999 STOP
 1000 PRINT CHR$(12)
 1005 PRINT "                C O L U M N S
"
 1006 PRINT
 1007 PRINT "ROW      1.     2.     3.     4
.     5."
 1010 PRINT "          +------+------+------+
------+------+"
 1020 FOR C= 1 TO 7
 1030 PRINT " "CHR$(C+64);".      |      |
     |      |      |"
 1040 PRINT " --------+------+------+------+
------+------+"
 1050 NEXT C
 1060 PRINT " "CHR$(C+64);".      |      |
     |      |      |"
 1070 PRINT " --------+------+------+------+
------+------+"
 1080 RETURN
 1100 P$=CHR$(Y+64)+STR$(X):PRINT CO$;CD
$"CELL:";P$;" "
 1110 LET A$=GET$
 1120 IF A$=CHR$(137) THEN 1200:REM MOVE
 CURSOR RIGHT
 1130 IF A$=CHR$(136) THEN 1300:REM MOVE
 CURSOR LEFT
 1140 IF A$=CHR$(138) THEN 1400:REM MOVE
 CURSOR DOWN
 1150 IF A$=CHR$(139) THEN 1500:REM MOVE
 CURSOR UP
 1152 IF A$="H"THEN GOSUB 6000:REM PRINT
 HELP SCREEN
 1154 IF A$="F" THEN GOSUB 2000:REM INPU
T FORMULA
 1156 IF A$="S"THEN GOSUB 5150:REM STORE
 CURRET SHEET
 1158 IF A$="G"THEN GOSUB 5100:REM GET P
REVIOUS SHEET
 1160 IF A$="C" THEN GOSUB 2300:REM CALC
ULATE SHEET
 1165 IF A$=CHR$(13) THEN RETURN
 1170 IF A$>="0" AND A$<="9" THEN GOSUB
2100:REM INPUT DATA ROUTINES
 1180 IF A$="Z" THEN GOSUB 5000:REM CLEA
R SHEET
 1185 IF A$="R" THEN GOSUB 5700:REM REPL
ICATE SHEET
 1187 IF A$="T" THEN GOSUB 5200:REM TAB
TO NEW CELL
 1189 IF (INKEY(-119)) THEN GOSUB 7000:R
EM LOAD SAVE ROUTINES
 1190 GOTO 1100:REM GO BACK TO START
 1200 REM ****** MOVE RIGHT ******
 1210 IF X=15 THEN 1100
 1220 IF X=H2 THEN GOSUB 1600:X=X+1:H1=H
1+1:H2=H2+1:GOTO 1270
 1230 GOSUB 1600:LET X=X+1:GOSUB 1650:GO
TO 1100
 1270 GOSUB 1800:GOSUB 1700:GOTO 1100
 1300 REM ******* MOVE LEFT *****
 1310 IF X=1 THEN 1100
 1320 IF X=H1 THEN GOSUB 1600:X=X-1:H1=H
1-1:H2=H2-1:GOTO 1370
 1330 GOSUB 1600:LET X=X-1:GOSUB 1650:GO
TO 1100
 1370 GOSUB 1800:GOSUB 1700:GOTO 1100
 1400 REM **** MOVE DOWN *****
 1410 IF Y=15 THEN 1100
 1420 IF Y=V2 THEN GOSUB 1600:LET Y=Y+1:
V1=V1+1:V2=V2+1:GOTO 1470
 1430 GOSUB 1600:LET Y=Y+1:GOSUB 1650:GO
TO 1100
 1470 GOSUB 1850:GOSUB 1700:GOTO 1100
 1500 REM ***** MOVE UP *****
 1510 IF Y=1  THEN 1100
 1520 IF Y=V1 THEN GOSUB 1600:LET Y=Y-1:
V1=V1-1:V2=V2-1:GOTO 1570
 1530 GOSUB 1600:LET Y=Y-1:GOSUB 1650:GO
TO 1100
 1570 GOSUB 1850:GOSUB 1700:GOTO 1100
 1600 REM ****** TURN CURSOR OFF ****
 1610 P$=STR$(M(Y-V1+1,X-H1+1))
 1615 IF LEN(P$)<5 THEN P$=" "+P$:GOTO 1
615
 1620 COLOUR 1:COLOUR 128
 1625 PRINT TAB(H(X-H1+1)-1,V(Y-V1+1)-1)
;P$
 1630 COLOUR 1:COLOUR 128
 1640 RETURN
 1650 REM ****** TURN CURSOR ON *****
 1660 P$=STR$(M(Y-V1+1,X-H1+1))
 1665 IF LEN(P$)<5 THEN P$=" "+P$:GOTO 1
665
 1670 COLOUR 2:COLOUR 129
 1675 PRINT TAB(H(X-H1+1)-1,V(Y-V1+1)-1)
;P$
 1680 COLOUR 1:COLOUR 128
 1690 GOSUB 1900:RETURN
 1700 REM ***** PRINT DATA WINDOW FROM S
HEET ON SCREEN ****
 1710 FOR I=0 TO 7
 1720 FOR J=0 TO 4
 1730 P$=STR$(M(I+V1,J+H1))
 1735 IF LEN(P$)<5 THEN P$=" "+P$:GOTO 1
735
 1740 PRINT TAB(H(J+1)-1,V(I+1)-1);"
 "
 1745 PRINT TAB(H(J+1)-1,V(I+1)-1);P$
 1750 NEXT J,I
 1760 GOSUB 1650:RETURN
 1800 REM ***** PRINT COLUMN NO. ****
 1810  FOR I=H1 TO H2:PRINT TAB(8+6*(I-H
1),3);I;"."
 1820 NEXT I:RETURN
 1850 REM ****** PRINT ROW LABELS *****
 1860 FOR C=V1 TO V2
 1870 PRINT TAB(1,V(C-V1+1)-1);CHR$(C+64
);"."
 1880 PRINT:NEXT C:RETURN
 1900 REM **** FORMULA OF CURRENT CELL *
**
 1910 LET D$=F$((Y-1)*15+X)
 1920 PRINT TAB(0,22);"
                      "
 1930 PRINT TAB(0,22);"FORMULA: ";D$
 1940 RETURN
```

Routinen zum Feldaufbau

```
 3000 REM ***** SETUP ARRAYS ******
 3010 DIM H(5),V(8),ST(20),ST$(20),E$(20
),G$(20),C(20)
 3020 FOR C=0 TO 4
 3030 LET H(C+1)=6*C+10
 3040 NEXT C
 3050 FOR C=1 TO 8
 3060 LET V(C)=2*C+4:REM CALC YPOS
 3070 NEXT C
 3075 X=1:Y=1:REM INITIAL CURSOR POSITIO
 3080 H1=X:H2=X+4:V1=Y:V2=Y+7
 3090 REM ***** DIM SHEET ARRAYS ****
 3100 DIM M(15,15):DIM N(15,15)
 3110 FOR I=1 TO 15:FOR J=1 TO 15
 3120 LET M(I,J)=I*J+I
 3130 NEXT J,I
 3140 DIM F$(225)
 3150 RETURN
```

Schneider CPC



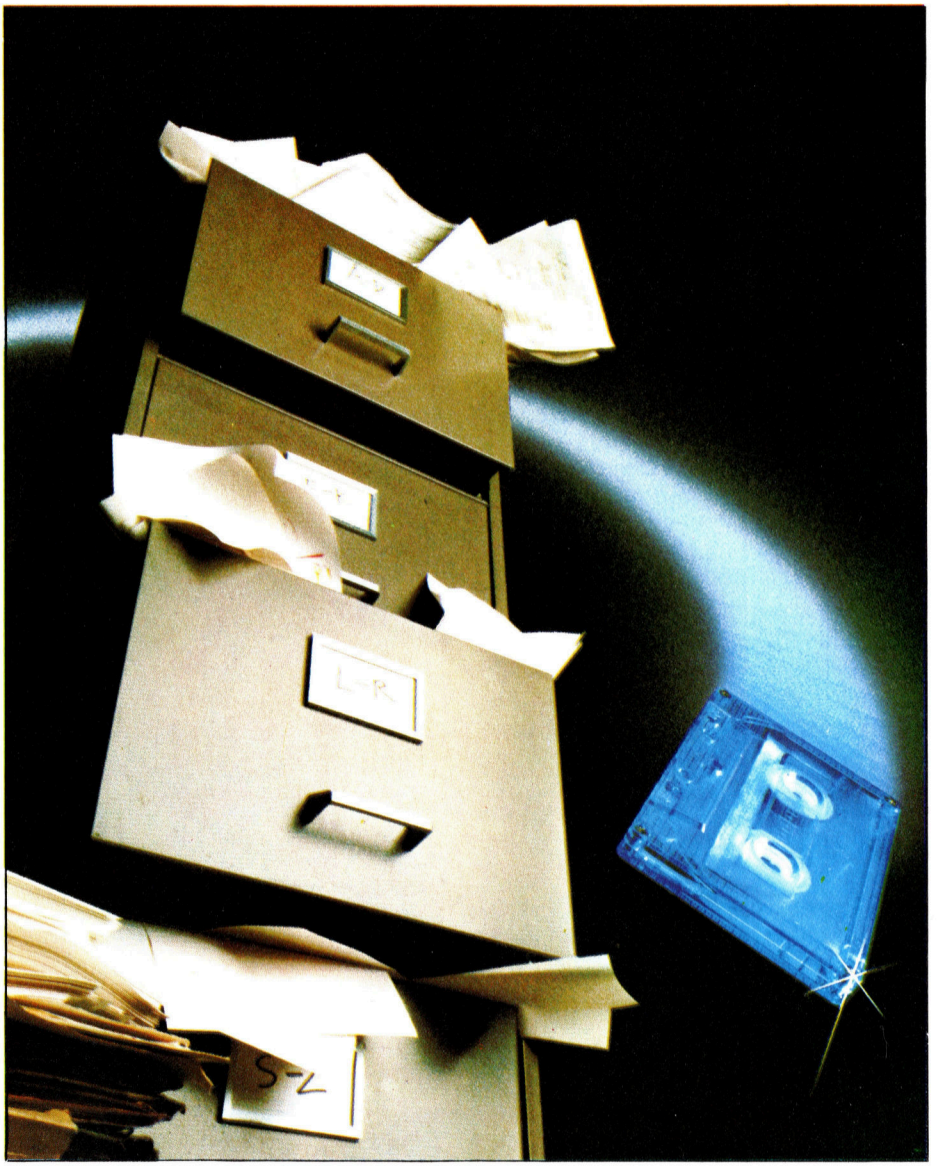
```
100 GOSUB 3000:REM SETUP ARRAYS & VARIA
BLES
110 GOSUB 1000:REM PRINT SCREEN
120 GOSUB 1700:REM PRINT DATA ON SCREEN
130 GOTO 1100
999 STOP
1000 REM PRINT SCREEN DISPLAY
1005 CLS:PRINT "                C O L U M
N S"
1006 PRINT
1007 PRINT "ROW      1.     2.     3.     4.
     5."
1010 PRINT "          +------+------+------+-
-----+------+"
1020 FOR C=1 TO 7
1030 PRINT " "CHR$(C+64);".      |      |
     |      |      |      |"
1040 PRINT " -------+------+------+------+-
-----+------+"
1050 NEXT C
1060 PRINT " "CHR$(C+64);".      |      |
     |      |      |      |"
1070 PRINT " -------+------+------+------+-
-----+------+"
1080 RETURN
1100 LOCATE 1,1:P$=CHR$(Y+64)+MID$(STR$(
X),2,2):PRINT "CELL ";P$;" "
1110 A$=INKEY$:IF A$="" THEN 1110
1120 IF A$=CHR$(243) THEN 1200:REM MOVE
RIGHT
1130 IF A$=CHR$(242) THEN 1300:REM MOVE
LEFT
1140 IF A$=CHR$(241) THEN 1400:REM MOVE
DOWN
1150 IF A$=CHR$(240) THEN 1500:REM MOVE
UP
1155 IF A$="F" THEN GOSUB 2000:REM INPUT
 FORMULA
1160 IF A$="C" THEN GOSUB 2300:REM CALCU
LATE SHEET
1165 IF A$=CHR$(13) THEN RETURN
1170 IF A$>="0" AND A$<"9" THEN GOSUB 21
00:REM ENTER NUMERIC DATA
1180 IF A$="B" THEN GOSUB 5000:REM CLEAR
 SHEET
1185 IF A$="V" THEN GOSUB 5100:REM GET P
REVIPOUS SHEET
1187 IF A$="G" THEN 5200:REM MOVE CURSOR
 TO NEW CELL
1188 IF A$="R" THEN GOSUB 5700
1190 GOTO 1100
1200 REM **** MOVE RIGHT ****
1210 IF X=15 THEN 1100
1220 IF X=H2 THEN GOSUB 1600:X=X+1:H1=H1
+1:H2=H2+1:GOTO 1270
1230 GOSUB 1600:LET X=X+1:GOSUB 1650:GOT
O 1100
1270 GOSUB 1800:GOSUB 1700:GOTO 1100
1300 REM **** MOVE LEFT ***
1310 IF X=1 THEN 1100
1320 IF X=H1 THEN GOSUB 1600:X=X-1:H1=H1
-1:H2=H2-1:GOTO 1370
1330 GOSUB 1600:LET X=X-1:GOSUB 1650:GOT
O 1100
1370 GOSUB 1800:GOSUB 1700:GOTO 1100
1400 REM **** MOVE DOWN ****
1410 IF Y=15 THEN 1100
1420 IF Y=V2 THEN GOSUB 1600:Y=Y+1:V1=V1
+1:V2=V2+1:GOTO 1470
1430 GOSUB 1600:LET Y=Y+1:GOSUB 1650:GOT
O 1100
1470 GOSUB 1850:GOSUB 1700:GOTO 1100
1500 REM *** MOVE DOWN ****
1510 IF Y=1 THEN 1100
1520 IF Y=V1 THEN GOSUB 1600:Y=Y-1:V1=V1
-1:V2=V2-1:GOTO 1570
1530 GOSUB 1600:LET Y=Y-1:GOSUB 1650:GOT
O 1100
1570 GOSUB 1850:GOSUB 1700:GOTO 1100
1600 REM *** TURN CURSOR OFF ****
1610 LET P$=MID$(STR$(M(Y,X)),2)
1615 IF LEN(P$)<5 THEN P$=" "+P$:GOTO 16
15
1620 LOCATE H(X-H1+1)-1,V(Y-V1+1):PRINT
P$
1630 RETURN
1650 REM *** TURN CURSOR ON ****
1660 LET P$=MID$(STR$(M(Y,X)),2)
1665 IF LEN(P$)<5 THEN P$=" "+P$:GOTO 16
65
1670 LOCATE H(X-H1+1)-1,V(Y-V1+1):PRINT
CHR$(24);P$;CHR$(24);
1680 GOSUB 1900:RETURN
1700 REM **** PRINT DATA IN SHEET ****
1710 FOR I=0 TO 7
1720 LOCATE 10,V(I+1)
1730 FOR J=0 TO 4
1735 LET P$=MID$(STR$(M(I+V1,J+H1)),2)
1740 IF LEN(P$)<5 THEN P$=" "+P$:GOTO 17
40
1745 LOCATE H(J+1)-1,V(I+1):PRINT P$;
1750 NEXT J,I
1760 GOSUB 1650:REM TURN CURSOR ON
1770 RETURN
1800 REM ***** PRINT COLUMN NOS ****
1810 LOCATE 1,3:PRINT "ROW      ";;
1820 LOCATE 7,3:FOR I=H1 TO H2:PRINT TAB
(H(I-H1+1)-3)I;CHR$(8);".";
1830 NEXT I
1840 RETURN
1850 REM ***** PRINT ROW LETTERS *****
1860 LOCATE 1,4
1870 FOR I=V1 TO V2
1875 PRINT
1880 PRINT " ";CHR$(I+64);"."
1890 NEXT I
1895 RETURN
1900 REM ***** FORMULA OF CURRENT CELL *
**
1920 LET D$=F$((Y-1)*15+X)
1930 LOCATE 1,22
1940 PRINT "FORMULA:
          "
1950 PRINT  '

FORMULA:";D$
1960 LOCATE 1,1
1970 RETURN
```

Routinen zum Feldaufbau

```
3000 REM ******************************
3001 REM * SETUP ARRAYS & VARAIBLES   *
3002 REM ******************************
3010 DIM H(5),V(8),ST(20),ST$(20),E$(20)
,G$(20),C(20)
3020 FOR C= 0 TO 4
3030 H(C+1)=6*C+11:REM CALC XPOS
3040 NEXT C
3050 FOR C=1 TO 8
3060 LET V(C)=2*C+3:REM CALC YPOS
3070 NEXT C
3075 LET X=1:Y=1
3080 LET H1=X:H2=X+4:V1=Y:V2=Y+7
3090 REM ******************************
3091 REM * VALUE ARRAY                 *
3092 REM ******************************
3100 DIM M(15,15):DIM N(15,15)
3110 FOR I=1 TO 15
3120 FOR J=1 TO 15
3130 LET M(I,J)=I*J
3140 NEXT J,I
3150 REM * FORMULA ARRAY               *
3152  REM ******************************
3160 DIM F$(255)
3170 LET F$(1)="A1+B1+C1"
3180 LET F$(31)="C1+C2+C3"
3190 LET F$(16)="B1+B2+B3"
3200 RETURN
```

Alles in Ordnung!

Ist Ihr Adreßbuch ein einziges Chaos? Sind die Clubkarten durcheinander geraten, oder macht Ihre Cassettenkartei mehr Ärger, als sie nützt? Unser Programm schafft Ordnung.



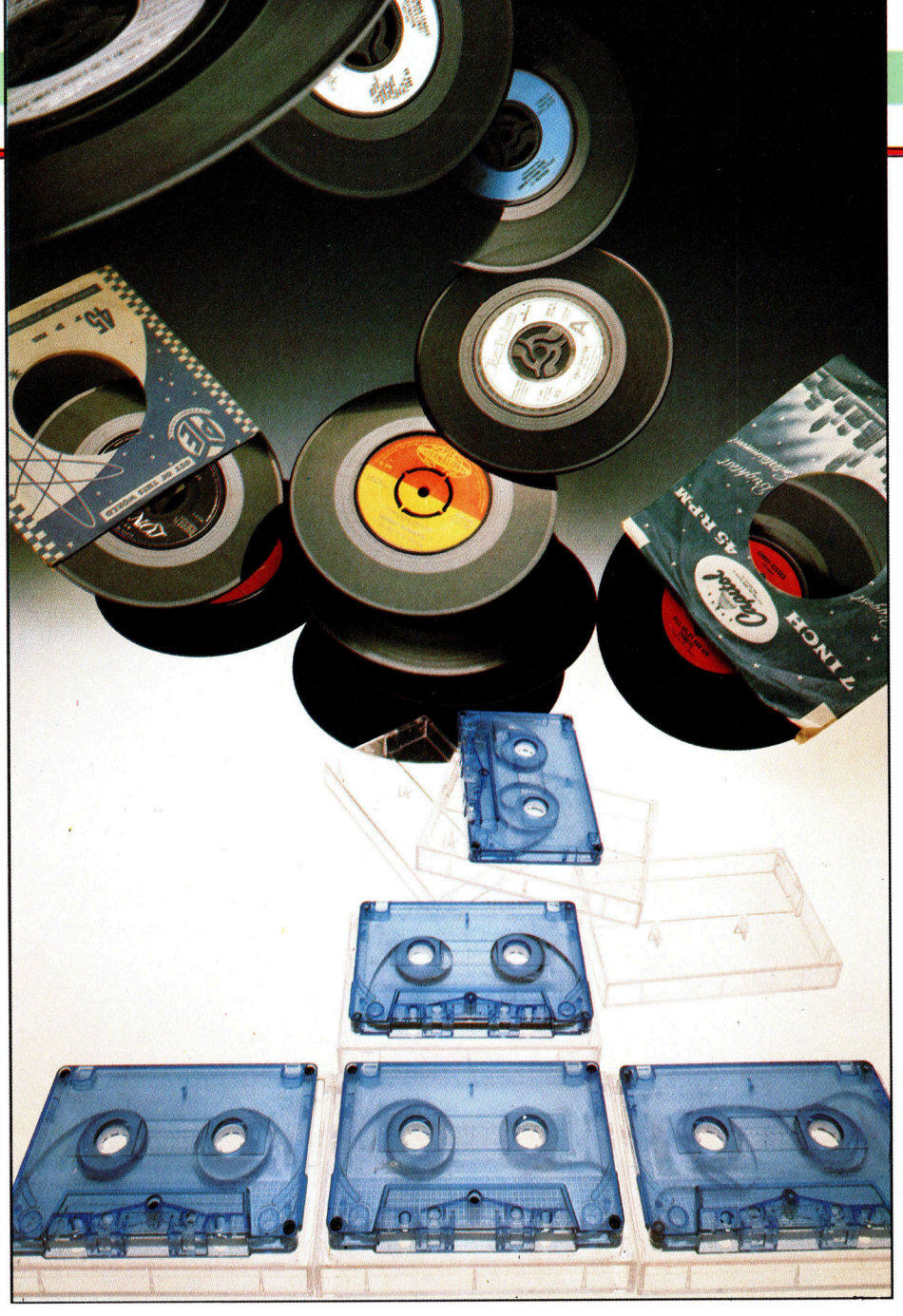
Und was kann man damit anfangen?" ist eine Frage, die man oft von Leuten ohne Computer hört. Manch einer kennt dann keine befriedigende Antwort. Hier ist ein Programm, das den Computer etwas Sinnvolles tun läßt. Es dreht sich um eine Computerdatei, wie sie im täglichen Leben Verwendung finden kann. Etwa wenn man die Adressen von Freunden oder die der Mitglieder eines Clubs speichern, die Geburtstage von Verwandten und Bekannten nicht ewig vergessen, seine Münzen-, Schallplatten- oder Rezeptsammlung katalogisieren oder den Überblick über seine ständig wachsende Computerspielsammlung behalten will. Die einzige Einschränkung für die Dateien liegt in der Speichergröße des Rechners. Deshalb sollten Sie Ihre Einträge möglichst kurz und bündig halten.

Die Datei eröffnen

Nachdem Sie das Programm eingegeben und gestartet haben, erscheint auf dem Bildschirm das Hauptmenü. Es repräsentiert eine Auswahl der Möglichkeiten, die das Programm bietet. Sie können etwa Daten abfragen oder Daten suchen. Aber zuerst müssen Sie eine Datei eröffnen.

Um eine Datei zu eröffnen, muß der Rechner wissen, wieviele Datensätze Sie benötigen und wie lang sie sein sollen. „Datei eröffnen" ist Möglichkeit (Option) 1 des Hauptmenüs. Um sie auszuwählen, müssen Sie die 1 drücken. Dann fragt der Rechner: „Bist du dir sicher?" Das ist ein Schutz gegen unbeabsichtigtes Drücken der Taste 1, weil beim Eröffnen der Datei alle gespeicherten Daten gelöscht werden. Wer sich sehr wohl „sicher ist", drückt j für ja; wenn nicht, irgendeine andere Taste. Sie kommen dann automatisch ins Hauptmenü zurück.

Nachdem man j gedrückt hat, fragt der Rechner nach der Anzahl der benötigten Felder. Felder sind die Einzelinformationen in einem Datensatz. Eisenbahnfans gebrauchen wahrscheinlich vier Felder: Lokomotivklasse, Nummer, wann und wo gesehen. Ein Datensatz darf nicht mehr als acht Felder enthalten, weil nur acht gleichzeitig auf den Bildschirm passen. Nachdem Sie die Anzahl der Felder festgelegt haben, ist die nächste Frage: „Name Feld 1?". Im Beispiel oben wäre die Antwort: „Klasse". Dann werden Sie nach der Länge des ersten Feldes gefragt. Das ist die Anzahl der Zeichen, die dort stehen sollen. Die maximale Länge eines Feldes darf 19 Zeichen betragen (54 auf dem Schneider). Falls Ihre Information länger ist, müssen Sie sie in mehrere Felder aufteilen. Nachdem der



Rechner die Informationen für das erste Feld erhalten hat, verlangt er sie noch für das zweite, dritte und so weiter. Ihre Datei kann also um so mehr Datensätze speichern, je kürzer Name und Zeichenzahl eines Feldes sind. Nachdem diese Übung vollzogen ist, rechnet der Computer sofort aus, für wieviele Datensätze er Platz hat. Der Spectrum fragt zusätzlich, wieviele dieser Datensätze Sie wirklich benötigen. Falls Sie mit weniger auskommen, als Platz vorhanden ist, würde der Spectrum sonst viel Zeit mit dem Abspeichern des leeren Speicherplatzes vergeuden.

Nach dem Eröffnen der Datei befindet man sich automatisch wieder im Hauptmenü. Dort wählt man Option 2, um mit der Eingabe der Daten zu beginnen.

Datensatz verschieben

In der obersten Bildschirmzeile steht stets die Meldung, die die Anzahl der möglichen und der schon eingegebenen Datensätze angibt. Dort steht beispielsweise: „DU HAST 10 VON 100 DATENSÄTZEN BELEGT". Direkt darunter prangen die Namen der Felder. Der Cursor steht ganz unten. Hier geben Sie auch Ihre Daten ein. Denken Sie daran, die Maximallänge nicht zu überschreiten.

Nachdem Sie die RETURN- oder ENTER-Taste gedrückt haben, wird der eingegebene Datensatz nach oben verschoben. Die Eingabezeile wird gelöscht, und Sie können weitere Daten eintippen. Jedesmal, wenn ein Datensatz eingegeben ist, wird er gespeichert. Der Computer geht dann sofort zum nächsten leeren Datensatz über. Wenn Sie die RETURN- oder ENTER-Taste drükken, bevor Sie einen weiteren Datensatz eingegeben haben, bringt Sie der Computer ins Hauptmenü.

Damit Sie sich die Daten wieder ansehen können, müssen Sie Option 3 des Hauptmenüs wählen. Auf dem Bildschirm erscheint dann der erste Datensatz. Das ist nicht unbedingt der erste, den Sie eingegeben haben, sondern der erste, den der Rechner an diese Stelle alphabetisch korrekt sortiert hat.

Die Methoden zum alphabetischen Sortieren variieren zwar etwas, normalerweise werden die Daten aber nach dem ersten Feld sortiert. Der Rechner sieht sich jeweils den ersten Eintrag im ersten Feld an und sortiert die Datensätze alphabetisch. Falls in zwei Datensätzen der erste Buchstabe übereinstimmt, sieht sich der Computer den zweiten an, dann den dritten und so weiter.

Das erste Problem entsteht, wenn Zahlen im ersten Feld stehen. Der Rechner sortiert eine Zahl vor einen Buchstaben. Ansonsten benutzt er dieselbe Sortiermethode wie für Buchstaben. Er betrachtet also nicht die ganze Zahl, sondern sortiert nach den einzelnen Ziffern. Falls also im ersten Feld Ihrer Datensätze die Zahlen von 1 bis 100 stehen, wird der Computer folgende Reihenfolge auswählen: 1, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 100 und dann 2, 20, 21 usw.

Eine Möglichkeit, dieses Problem elegant zu lösen, ist, entweder keine Zahlen im ersten Feld zu benutzen oder die Zahlen so einzugeben: 001, 002, 003 . . . 100.

Ein zweites Problem entsteht, wenn Sie Groß- und Kleinschrift vermischen, weil der Rechner große vor kleinen Buchstaben auswählt. Also kommt ABC GmbH vor Aaron & Co. Wenn möglich, sollten Sie nur Großbuchstaben benutzen.

Wenn Sie sich die Daten ansehen, erscheint am Bildschirmrand:

V(ORWÄRTS) Z(URÜCK) M(ENÜ)

Wenn Sie die V-Taste drücken, er-

scheint der nächste Datensatz. Wenn man V mehrfach drückt, sieht man die gesamte Datei.

Drücken Sie die Z-Taste, erscheint der Datensatz vor dem jetzt auf dem Schirm abgebildeten. Man kann also mit der V- und Z-Taste nach Belieben vor- und rückwärts durch die ganze Datei laufen. Mit der M-Taste gelangt man sofort ins Hauptmenü.

„Daten finden" ist Option 4 des Hauptmenüs. Mit ihr können Sie sämtliche Datensätze durchsuchen.

Wenn Sie Taste 4 betätigen, fragt der Rechner, in welchem Feld er suchen soll. Sie müssen dessen Nummer angeben: 1 für das erste, 2 für das zweite usw., von oben gezählt. Nun sollten Sie eingeben, wonach in diesem Feld geforscht werden soll; beispielsweise „JOJO" oder „WURSTHAUSEN" oder „SCHEIBENKLEISTERWEG". Nach der Eingabe drükken Sie RETURN oder ENTER.

Das Wort muß genau mit dem im Feld gespeicherten übereinstimmen. Falls es im Feld in Großbuchstaben steht, werden Sie es nicht finden, wenn Sie nach Kleinbuchstaben suchen. Auch der Wortabstand muß exakt stimmen. Falls Sie versehentlich die Leertaste vor oder nach dem Eintrag gedrückt haben, wird der Computer vermutlich nichts Gescheites finden. Dieser Fehler tritt sogar dann auf, wenn beide Wörter auf dem Schirm gleich aussehen.

Verwenden Sie deshalb möglichst sowohl bei der Eingabe als auch beim Suchen nur Großbuchstaben, und vermeiden Sie Leerzeichen.

Alphabetisch gelistet

Wenn der Computer die gesuchten Daten nicht findet, teilt er es mit und kehrt ins Hauptmenü zurück. Wenn er sie findet, listet er sie alphabetisch auf. Die zweite der beiden Angebotszeilen am unteren Bildschirmrand lautet:
A(ENDERN) L(OESCHEN) D(RUCKEN)

Wenn Sie den angezeigten Datensatz ändern möchten, drücken Sie die A-Taste. Dann wird nach der Nummer des zu ändernden Feldes gefragt. Diese Feldnummer wird wieder von oben gezählt.

Dann sollten Sie die „Aenderung eingeben". Dazu müssen Sie das gesamte Feld neu eingeben, selbst wenn nur eine Zahl geändert werden soll. Die Korrektur wird mit RETURN oder ENTER beendet. Sie können anschließend ein weiteres Feld ändern, indem Sie A drücken. Wollen Sie einen Eintrag ändern, der sich nicht bei den gerade ausgewählten befindet, müssen Sie ins Hauptmenü zurückkehren (M), neue Daten suchen (4) und diese dann korrigieren (A). Dazu können Sie auch Option 3, „Daten ansehen", benutzen.

Um einen Eintrag zu löschen, müssen Sie ihn mittels Option 3 oder 4 des Hauptmenüs finden. Dann erhalten Sie wieder die bekannte Angebotszeile. Um den Eintrag zu löschen, müssen Sie nur noch die Taste L drücken. Dann wird gefragt: „Bist du sicher?" Das ist ein guter Schutz gegen unbeabsichtigtes Löschen wichtiger Daten.

Wenn Sie aber sicher sind, den Eintrag löschen zu wollen, drücken Sie die Taste „J", ansonsten irgendeine andere. Nun wird der Eintrag gelöscht, und der (entsprechend der im Hauptmenü gewählten Option) nächste findet sich auf dem Bildschirm.

Wenn Sie den Eintrag, der als letzter gefunden wurde, gelöscht haben, bringt Sie der Computer automatisch ins Hauptmenü zurück.

Die letzte Option der beiden Angebotszeilen ist „D(RUCKEN)". Drükken Sie die D-Taste beim Commodore oder Schneider, so erhalten Sie die Meldung: „Teste Drucker". Sie sollten dann prüfen, ob der Drucker eingeschaltet und angeschlossen ist. Ist das nicht der Fall, besteht die Gefahr, daß das Programm abstürzt.

Dieser Test ist beim Atari und beim Spectrum nicht nötig, weil diese nicht reagieren, wenn kein Drucker angeschlossen ist.

Ist die Überprüfung gelaufen, sollten Sie die Taste W für „weiter" drücken. Wenn Sie eine andere Taste nehmen, stehen wieder alle sechs Optionen zur Verfügung. Drücken Sie aber C, und der Drukker arbeitet nicht, weil er vielleicht doch nicht richtig angeschlossen war, müssen Sie das Programm abbrechen: beim Commodore mit RUN/STOP, beim Schneider mit ESC ESC. Um wieder neu zu starten, sollten Sie GOTO 10 beim Schneider eingeben. Beim Commodore müssen sie allerdings GOTO 100 tippen.

Wenn Sie eine Datei abspeichern möchten, drücken Sie die 5. Dann werden Sie aufgefordert, der Datei einen Namen zu geben. Haben Sie das getan und RETURN oder ENTER gedrückt, sagt Ihnen der Rechner: „DATEI WIRD GESPEICHERT".

Jetzt werden, außer beim Spectrum, nur die Daten abgespeichert. Um das Programm zu speichern, müssen Sie Option 7 „PROGRAMM BEENDEN" wählen und SAVEn.

In zwei Stufen laden

Wenn Sie sich eine Datei wieder ansehen wollen, müssen Sie sie in zwei Stufen laden. Zuerst muß das Programm selbst geladen und gestartet werden. Dann wählen Sie Option 6 im Hauptmenü, um die Datei zu laden. Dazu müssen Sie wieder den Dateinamen eingeben, unter dem Ihre Informationen abgespeichert sind. Der Computer antwortet: „DRÜCKE PLAY UND EINE TASTE".

Nun sucht der Rechner auf der Cassette die Datei und lädt sie. Er bestätigt dann: „„KORREKT GELADEN". Findet der Computer die gesuchte Datei nicht, dann listet er alle die auf, die er gefunden hat, und kehrt danach automatisch ins Hauptmenü zurück. Zu diesem Zeitpunkt können Sie, wieder mit Option 6, eine andere Datei laden oder weitersuchen. Auf dem Spectrum werden die Daten und das Programm zusammen geladen. Anschließend können Sie alle weiteren Dateien mit Option 6 des Hauptmenüs laden.

Der Atari fragt beim Laden oder Speichern einer Datei nach „GERAET:DATEINAME". An dieser Stelle können Sie entscheiden, welches Gerät angesprochen werden soll. Falls Sie eine Diskettenstation besitzen, geben Sie hier „D: (Filename)" ein, beim Recorder „C: (Filename)". Der Name kann acht Buchstaben lang sein und wird mitgespeichert.

Sie verfügen jetzt über ein sehr gutes Dateisystem, mit dem Sie Dateien jeder Art erstellen können. Auf einer Cassette lassen sich etliche Dateien unterbringen. Aber es ist si-

cher sinnvoller, verschiedene Dateien auf verschiedenen Cassetten zu eröffnen.

Im Prinzip arbeiten die „großen" Dateien von Rechenzentren in Behörden oder Versicherungsgesellschaften wie die hier abgedruckte. Allerdings sind in diesen Anlagen um einiges größere Mengen an Daten zu verarbeiten. Aber selbst für den Hausbedarf könnte ein Dateiprogramm mal etwas knapp sein.

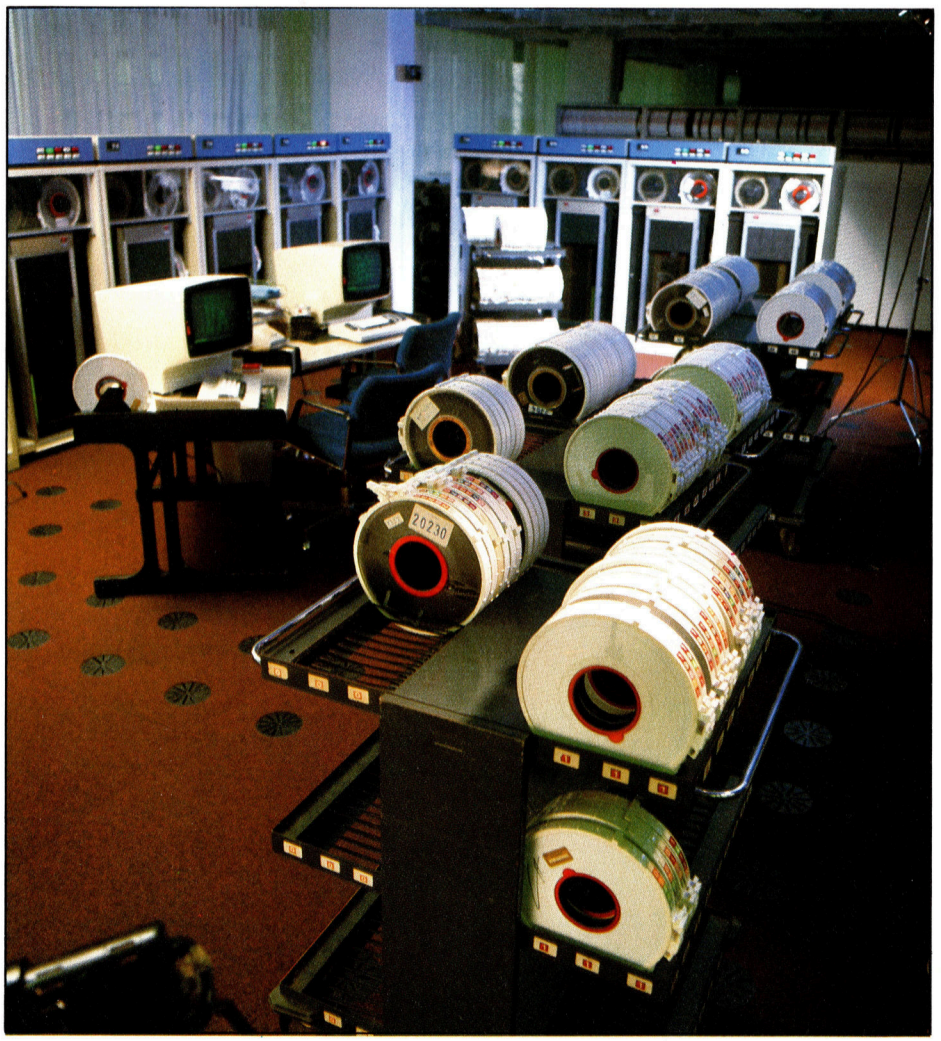
Denken Sie darum schon beim Einrichten Ihrer Dateien daran, daß Sammlungen aller Art die Tendenz haben zu wachsen. Um dem Schicksal zu entgehen, sehr bald schon größere Datenmengen zu jonglieren, unterteilt man etwa eine wirklich große Schallplattensammlung von vornherein in sinnvolle Abschnitte; vielleicht nach den Interpreten, so daß man Schallplatten A–K, L–R und S–Z erhält. Dann ist auf den Cassetten genügend Raum für Nachträge. Und achten Sie von Anfang an darauf, prägnante Abkürzungen zu verwenden. Wenn Sie Ihre Computerprogramme unter Kontrolle bringen wollen, notieren Sie etwa den Datenträger als C, D oder M statt als Cassette, Disk und Modul. So spart man Platz.

Um mit einer Floppy zu arbeiten, müssen sie in den Zeilen 5010 und 6010 „open 1,1" in „open 1,8" ändern!

Atari

```
1 REM DATEI TEIL EINS
10 OPEN #5,4,0,"K":POKE 82,1
20 GOSUB 900
100 PRINT CHR$(125): PRINT
    L$;"Hauptmenue": POKE 710,210
110 PRINT L$;"1...Datei eroeffnen"
120 PRINT L$;"2...Daten eingeben"
130 PRINT L$;"3...Daten abfragen"
140 PRINT L$;"4...Daten suchen"
150 PRINT L$;"5...Speichern"
160 PRINT L$;"6...Laden"
170 PRINT L$;"7...Programm beenden"
180 PRINT :PRINT "Waehle eine
    Zahl":S=1:GOSUB 930
190 GET #5,T:T=T-48: ON T<1 OR T>7
    GOTO 190: GOTO T*1000
900 DIM W$(100),ML$(57): RESTORE 911:
    FOR I=1 TO 57:READ P:
    ML$(I)=CHR$(P):NEXT I
910 DIM L$(10):L$="□":
    L$(10)="□":L$(2)=L$: RETURN
911 DATA 104,104,104,170,104,104,157,66,3,
    104,157,69,3,104,157,68,3,104,157,73,3,
    104,157,72,3,32,86,228,48,11
912 DATA 189,72,3,133,212,189,73,3,133,213,
    96,169,0,133,212,133,213,189,67,3,201,
    136,240,232,133,195,96
920 FOR L=1 TO 2:GOSUB 930: NEXT L
930 FOR T=15 TO 0 STEP -0.6: SOUND
    0,0+S*10,10,T: SOUND 1,1+S*10,10,T:
    NEXT T:RETURN
940 POSITION 2,20: FOR T=1 TO 4: PRINT
    CHR$(156);:NEXT T: RETURN
950 PRINT "Weiter mit RETURN";:GET #5,T:
    GOTO 100
960 POSITION 4,8: PRINT "Eingeben:": PRINT
    "Geraet:Dateiname": INPUT W$: IF
    LEN(W$) THEN RETURN
970 POP :GOTO 100
1000 GRAPHICS 17:POSITION 2,4: PRINT
    #6;"datei eroeffnen": POSITION 2,7:
    PRINT #6;"BIST DU SICHER?": POKE
    764,255: S=24
1010 T=T+1:ON PEEK(764)<>255 GOTO
    1020: ON T<20 GOTO 1010: POKE
    708,40*(PEEK(708)=0): T=0: GOTO
    1010
1020 GET #5,T:GRAPHICS 0: ON T<>74
    GOTO 100: POKE 710,16
1030 CLR :GOSUB 900:PRINT : PRINT
    "Anzahl der Felder (1-8)";:TRAP 1030:
    INPUT F: IF F<1 OR F>8 THEN 1030
1040 DIM N$(F*10),L(F),P(F):
    N$="□":N$(F*10)="□":
    N$(2)=N$:P=1
1050 FOR I=1 TO F:PRINT : PRINT "Name
    Feld ";I;: INPUT W$: IF LEN(W$)>10
    THEN W$(11)=" "
1060 N$((I-1)*10+1,I*10)=W$
1070 PRINT "Laenge Feld ";I;: TRAP
    1070:INPUT L:L(I)=L: TRAP 40000: IF
    L<1 OR L>27 THEN 1070
1080 P(I)=P:P=P+L:TL=TL+L: NEXT I:
    DIM R$(TL)
1090 SP=INT((FRE(0)-100)/TL):PRINT:
    PRINT :PRINT "Max. Anzahl
    Datensaetze= ";SP: DIM DAT$(SP*TL)
1100 GOTO 950
2000 IF NOT SP OR LR=SP THEN PRINT :
    PRINT "Keine Datei im Speicher":S=25:
    GOSUB 920:GOTO 100
2010 PRINT CHR$(125): PRINT "Du
    hast ";LR;" von ";SP; " Datensaetzen":
    PRINT "belegt": R$="□":
    R$(TL)="□": R$(2)=R$
2020 FOR I=1 TO F: POSITION 2,1+I*2:
    PRINT N$((I-1)*10+,I*10);":"
2030 POSITION 2,18: FOR T=1 TO 4: PRINT
    CHR$(156);:NEXT T: PRINT "Deine
    Eingabe (max. ";L(I);" Zeichen)":INPUT
    W$
2040 IF I=1 AND W$="" THEN 100
2050 IF LEN(W$)>L(I) THEN W$(L(I)+
    1)=""
2060 POSITION 13,1+I*2: PRINT W$
2070 R$(P(I), P(I)+LEN(W$)-1)=W$:
    NEXT I
2080 IF LR=0 THEN DAT$=R$: GOTO
    2200
2090 FOR I=LR TO 1 STEP -1
2100 IF DAT$((I-1)*TL+1,I*TL)<R$
    THEN POP: GOTO 2120
2110 DAT$(I*TL+1,I*TL+TL)=DAT$
    ((I-1)*TL+1,I*TL):NEXT I
2120 DAT$(I*TL+1,I*TL+TL)=R$
2200 LR=LR+1:GOTO 2000
3000 P=1:IF NOT LR THEN PRINT :PRINT
    "Keine Daten vorhanden!":S=25: GOSUB
    920:GOTO 100
3010 GOSUB 8500:GOSUB 8510
3020 GOSUB 940: PRINT "V(orwaerts)
    Z(urueck) M(enue)": PRINT " A(endern)
    L(oeschen) D(rucken)"
3030 GET #5,T:IF T=77 THEN 100
3040 IF T=86 THEN
    P=P+1-(P>LR-1): GOTO 3010
3050 IF T=90 THEN P=P-1+(P=1):
    GOTO 3010
3060 T=(T=68)+2*(T=65)+3*(T=76):
    ON T GOTO 8540,3200,3100:GOTO 3030
3100 GOSUB 940:PRINT "Daten loeschen";:
    S=25:GOSUB 920:PRINT " - BIST DU
    SICHER (J/N)":GET #5,L:IF L<>74
    THEN 3010
3110 DAT$((P-1)*TL+1)=DAT$(P*TL+1):
    LR=LR-1
3120 PRINT CHR$(125):PRINT "Datensatz
    geloescht!":GOTO 950
3200 GOSUB 940:GOSUB 930:PRINT
    "Welches Feld aendern?"
3210 GET #5,SU:SU=SU-48:IF SU<1 OR
    SU>F THEN 100
3220 PRINT "Aenderung (max. ";L(SU);
    " Zeichen":PRINT
    N$((SU-1)*10+1,SU*10);":";
3230 W$="□":W$(100)=W$:W$(2)=W$:
    INPUT W$
3240 W$(L(SU)+1)="":R$(P(SU),P(SU)+
    LEN(W$)-1)=W$
3250 DAT$(SU*TL+1,SU*TL+TL)=R$
3260 GOTO 3010
4000 PRINT CHR$(125):POSITION 10,1:
    PRINT "Daten suchen"
4010 IF LR<2 THEN 3000
4020 FOR I=1 TO F:POSITION 2,1+I*2:
    PRINT I;" - ";N$((I-1)*10+1,I*10):
    NEXT I
4030 PRINT :PRINT "In welchem Feld willst Du
    suchen ";
4040 GET #5,SU:SU=SU-48:IF SU<1 OR
    SU>F THEN 100
4050 PRINT CHR$(156);"Suchwort in
```

```
    Feld ";SU;" ";:INPUT W$
4060 FOR L=1 TO LR:P=(L-1)*TL+
    P(SU)
4070 IF DAT$(P,P+LEN(W$)-1)=W$
    THEN P=L:POP :GOTO3010
4080 NEXT L:GOTO 100
5000 POKE 710,16: IF NOT LR THEN 3000
5010 PRINT CHR$(125): PRINT "Datei
    sichern": PRINT
5020 GOSUB 960
5030 TRAP 5100:CLOSE #1: OPEN
    #1,8,0,W$:T=1
5040 T=T+1:ON W$(T,T)<>":" GOTO
    5040: IF LEN(W$)>T THEN
    W$=W$(T+1)
5050 PRINT #1;W$: PRINT "Schreibe Datei";
    CHR$(34);W$;CHR$(34)
5060 PRINT #1;F:PRINT #1;TL: FOR I=1
    TO F: PRINT #1;L(I): PRINT #1;P(I):
    NEXT I: PRINT #1;LR
5070 PRINT #1;N$: PRINT #1;DAT$:
    CLOSE #1
5080 TRAP 40000:GOTO 100
5100 PRINT :PRINT :GOSUB 920: CLOSE
    #1: PRINT "ERROR- ";
    PEEK(195);" beim Speichern ";W$: GOTO
    950
6000 POKE 710,16: PRINT CHR$(125):
    PRINT "Datei laden":PRINT :S=20:
    GOSUB 930
6010 GOSUB 960:TRAP 610:CLOSE #1:
    OPEN #1,4,0,W$:CLR :GOSUB 900
6020 INPUT #1,W$,F,TL:PRINT :PRINT
    "Lade ";W$: POKE 710,210
6030 DIM L(F),P(F),N$(F*10),R$(TL)
6040 SP=INT((FRE(0)-100)/TL): DIM
    DAT$(SP*TL)
6050 FOR I=1 TO F: INPUT #1,L,P:L(I)=
    L:P(I)=P:NEXT I
6060 INPUT #1,LR: PRINT LR;" von ";SP;
    "Datensaetzen belegt"
6070 INPUT #1,N$: L=USR(ADR(ML$),16,
    7,ADR(DAT$),SP*TL)
6080 IF NOT L THEN 6100
6090 DAT$(L+1)="":S=9: GOSUB 930:
    GOTO 100
6100 POSITION 2,20: PRINT "ERROR- ";
    PEEK(195);" beim Laden ";W$:S=25:
    GOSUB 920:GOTO 950
7000 PRINT CHR$(125): POKE 710,16:
    PRINT " M(enue) / B(ASIC)?";:
    GET #5,T: IF T<>66 THEN 100
7010 POKE 82,2:GRAPHICS 0:END
8500 R$=DAT$((P-1)*TL+1,P*TL):
    RETURN
8510 PRINT CHR$(125): PRINT
    #D;"Dateisatz Nummer ";P
8520 PRINT #D: FOR I=1 TO F
8530 PRINT #D;N$((I-1)*10+1,I*10);":";
    R$(P(I),P(I)+L(I)-1):NEXT I:RETURN
8540 TRAP 8560:CLOSE #4:OPEN #4,8,0,
    "P":D=4
8550 GOSUB 8510:D=0:GOTO 3010
8560 TRAP 40000:GOSUB 920: GOTO 3020
8700 GOSUB 920:GOTO 3030
```



Blick in ein Rechenzentrum: Daten über Daten auf Magnetbändern.

Commodore

```
5 print chr$(14)+chr$(8): poke 53280,6: poke
    53281,6: poke 808,234
10 gosub 20: goto 200
19 rem****** stringbelegung **********
20 cd$=chr$(17): ch$=chr$(19):
    cr$=chr$(29): cu$=chr$(145):
    cl$=chr$(157)
30 cs$=chr$(147): de$=chr$(20):
    rn$=chr$(18): rf$=chr$(146):
    rt$=chr$(13)
40 we$=chr$(5): ro$=chr$(150):
    gr$=chr$(30): hb$=chr$(154): rem*****
    farben
50 ul$="- - - - - - - - - - - - -
    - - - - - - - - - - - - - - - -
    - - - - - - - - - - - - - - - -"
60 fr$=ro$+"□□□□□"+rn$+"********
    □bist□du□sicher□?□********"+rf$
70 dim m$(7): for i=1 to 7: read m$(i): next i
80 data "Datei□eroeffnen", "Daten
    eingeben□", "Daten□ansehen□□"
90 data "Daten□suchen",
    "Speichern□□□",
    "Laden□□□□□□□",
    "Programm□beenden"
100 dim rv$(2): rv$(0)=rn$: rv$(2)=rf$
110 fl=0: dl$=chr$(20): qu$=chr$(34)
120 return
199 rem****** hauptmenue **********
200 print cs$+cd$+gr$+ul$
210 print "□*□□□□*□□*□□*□□*□*□□
    *□□□*□□□*□***□*□□*□*□
    □*"
220 print "□**□**□*□*□*□**□*□□
    □**□**□*□□□□**□*□*□□*"
230 print "□*□*□*□***□*□*□**□□
    □*□*□*□**□□*□**□*□□*"
240 print "□*□□□*□*□*□*□*□□*
    □□□*□□□*□*□□□*□□*□*
```

```
□□*"
250 print "□*□□□*□*□*□*□*□□*
    □□□*□□□*□***□*□□*□□**"
260 print rt$+ul$+rt$+we$
270 print rn$+cr$+"1□"+m$(1);: if fl=0
    then 350
280 print ,"4□"+m$(4)+rt$+rt$+cr$+
    rn$+"2□"+m$(2);
290 print ,"5□"+m$(5)+rt$+rt$+cr$+
    rn$+"3□"+m$(3);
350 print ,"6□"+m$(6)+rt$
360 print tab(10) rn$+"7□"+m$(7)+
    rt$+rt$+rt$+gr$: c=0: j=1
370 rem***** eingabewarteschleife
380 print tab(10) cu$+rv$(1+j)+
    "□Waehle eine Zahl□"
390 get a$:i=i+1: if i>15 then i=0: j=-j
400 if a$="" then 380
410 a=val(a$): if a< 1 or a > 7 then gosub
    12000: goto 380
420 if a=7 then 480
430 if fl=0 then 460
440 if a=1 or a=6 then 480
445 if a>2 and a<6 and be=0 then gosub
    12000: goto 380
450 goto 470
460 if a>1 and a<6 then gosub 12000: goto
    380
470 on a goto 1000,2000,3000,4000,5000,
    6000,7000
480 print cu$+fr$
490 get a$: if a$="" then 490
500 if a$="j" or a$="J" then 470
510 goto 200
999 rem****** (1) datei eroeffnen *****
1000 clr: gosub 20: print cs$+cd$+
    "Anzahl der Felder (1-8) ?□";
1010 b$="12345678": gosub 13010: fe=i:
    print str$(fe)+rt$
1015 dim fe$(fe), fl(fe)
1020 for j=1 to fe
1030 print "□□Name Feld"j"□:?□";
1040 x=20: l=16: gosub 1100
1050 if len(b$)=0 then gosub 12000: goto
    1040
1060 fe$(j)="□"+we$+rn$+b$+
    ":"+rf$
1070 print rt$+cu$+"□□Laenge Feld
    □"+b$+"□:?□";: gosub 14000
1080 x=35: l=2: gosub 11000: if b<1 or
    b>19 then gosub 12000: goto 1080
1090 fl(j)=b: z=z+b: print rt$
1100 next j
1110 a=fre(x)-65536*(fre(x)<0)
1120 da=int(a/(z+5+3*fe))
1130 print " Max. Anzahl Datensaetze ="da
1140 for i=1 to 1000: next
1150 dim da$(da,fe-1), z(da)
1160 fl=1: be=0
1170 za$=left$("12345678",fe)
```

```
1180 goto 2000
1999 rem****** (2) daten eingeben *****
2000 for j=be to da
2010 print cs$+rn$+we$+"Von"da"▮▮
    Datensaetzen sind"j"▮▮ belegt!"
2020 for k=1 to fe
2030 x=0: y=21: gosub 100000: print
    cr$+we$+rn$+"Datensatz
    Nummer:"j+1
2040 print rt$+fe$(k);: gosub 14000
2050 x=20- l=fl(k): gosub 11000: if
    b$="" and k=1 then 2130
2060 da$(j,k-1)=b$
2070 x=1: y=3+(k-1)*2: gosub 10000:
    print fe$(k);
2080 x=20: gosub 10000: print b$
2090 next k: if j=0 then z(j)=0: goto2120
2100 for k=0 to j-1: if da$(j,0)><da$
    (z(k),0) then 2140
2110 next k: z(j)=j
2120 next j
2130 be=j: goto 200
2139 rem***** einfuegen
2140 for i=j to k+1 step -1: z(i)=z(i-1):
    next
2150 z(k)=j: goto 2120
2999 rem****** (3) daten ansehen ******
3000 j=0
3010 if c>0 then if da$(z(j),c-1)<>c$
    then 3110
3015 print cs$+we$+rn$+"□Datensatz
    Nummer:"j+1
3020 for k=1 to fe
3030 print rt$+we$+str$(k)+fe$(k);
3040 x=20: y=peek(214): gosub 10000:
    print hb$+da$(z(j),k-1)
3050 next k
3060 x=0: y=19: gosub 10000
```

```
3070 print gr$+rn$+"■■■V(orwaerts)
   ■■A(endern) ■■M(enue) "
3080 print rt$+rn$+"■■■
   Z(urueck) ■■L(oeschen)■■
   D(rucken)"
3090 b$="vzalmd": gosub 13010
3100 on i goto 3110,3200,3300,3400,200,
   3600
3109 rem***** v(orwaerts)
3110 j=j+1: if j>=be then 3130
3120 goto 3010
3130 if c=0 then 3000
3140 print we$+rt$+"□□□□***** keine
   weiteren Daten *****"
3150 for i=0 to 1500: next: goto 200
3199 rem***** z(urueck)
3200 j=j-1: if j<0 then j=be-1
3210 goto 3010
3299 rem***** a(endern)
3300 if fe=1 then k=1: print rt$: goto 3330
3310 print rt$+we$+"□Bitte Feldnr.
   eingeben:□"
3320 b$=za$: gosub 13010: k=i
3330 print cu$+"□□"+fe$(k);: gosub
   14000
3340 x=20: l=fl(k): gosub 11000
3350 da$(z(j),k-1)=b$
3360 goto 3010
3399 rem***** l(oeschen)
3400 print rt$+fr$
3410 get a$: if a$="" then 3410
3420 if a$<>"j" and a$<>"J" then
   3010
3430 be=be-1: if be=0 then 200
3440 for i=0 to fe-1: da$(z(j),i)=da$
   (be,i): next
3450 if j=be and z(j)=be then 3010
3460 for i=0 to be: if z(i)=be then z(i)=z(j):
   goto 3480
3470 next i
3480 for i=j to be: z(i)=z(i)+1: next
3490 goto 3010
3599 rem***** d(rucken)
3600 print rt$+we$+"□□"+rn$+
   "****** Drucker fertig (j/n) ? ******"+rf$
3610 gosub 13000: if i=2 then 3010
3620 open 4,4,7: print#4, "□Datensatz
   Nummer:" j+1
3630 for i=1 to fe
3640 print#4, fe$(i) spc(20-fl(i))
   da$(z(j),i-1)
3650 next i: print#4, ul$+rt$: close 4: goto
   3010
3999 rem****** (4) daten suchen *******
4000 print cs$+we$
4010 if fe=1 then c=1: goto 4050
4020 for i=1 to fe: print rt$+"□□"+
   str$(i)+cr$+fe$(i): next
4030 print gr$+rt$+rt$+"□***□Welches
   Feld soll ich suchen□?□***□"
4040 b$=za$: gosub 13010: c=i
4050 print rt$+"□□□"+fe$(c);
4060 x=20: l=fl(c): gosub 11000
4070 c$=b$: goto 3000
4999 rem********** (5) speichern **********
5000 gosub 5500: open 1,1,1,b$: print
   rt$+"□□Ich speichere□!!!"
5010 print#1, fe;rt$;da;rt$;be;rt$
5020 for i=1 to fe: print#1, qu$+fe$(i)+
   qu$+rt$;fl(i);rt$;: next i
5030 for j=0 to be-1: for i=0 to fe-1:
   print#1, da$(z(j),i): next i,j
5040 close 1: goto 200
5499 rem***** dateinamen eingeben
5500 print cs$+we$+rt$+rt$+
   "□□□"+rn$+"□Bitte gib den
   Dateinamen ein□"+rt$+hb$
5510 x=10: l=16: gosub 11000
5520 if b$="" then gosub 12000: goto 5500
5540 print we$+rt$+rt$+"□□Recorder
   fuer Aufnahme vorbereiten,"
5550 print rt$+"□□und dann 'RETURN'
   druecken."
5560 get a$: if a$<>rt$ then 5560
5570 return
5999 rem****** (6) laden **************
6000 clr: gosub 20: gosub 5500
6010 open 1,1,0,b$: print rt$+"□□Ich lade
   !!!"
6020 input#1, fe, da, be
6030 dim fe$(fe), fl(fe), da$(da,fe-1), z(da)
6040 for i=1 to fe: input#1, fe$(i), fl(i): next
   i
6050 for i=0 to be-1
6060 for j=0 to fe-1
6070 input#1, da$(i,j): next j: z(i)=i: next i:
   closei
6080 fl=1:za$=left$("12345678",fe):
   goto 200
6999 rem****** (7) prg. beenden *******
7000 print cs$: end
9999 rem****** cursor setzen **********
10000 print ch$;
10010 if x>0 then for i=1 to x: print    r$;:
   next
10020 if y>0 then for i=1 to y: print cd$;:
   next
10030 return
10999 rem****** stringeingabe *********
11000 y=peek(214): gosub 10000: gosub
   14000
11010 p0=1024+40*y+x: p1=p0:
   i=128: b$=""
11020 poke p1,(peek(p1) and 127) or (i and
   128)
11030 i=(i+12) and 255: get a$: if a$=""
   then 11020
11040 if a$=dl$ then 11090
11050 if a$=rt$ then poke p1,peek(p1)and
   127: b=val(b$): return
11060 if asc(a$)and127<32 or a$=cm$
   then 1120
11070 if p1>=p0+1 then 11120
11080 poke p1,peek(p1)and127: p1=p1+1:
   print a$;: b$=b$+a$: goto 11020
11090 if p1=p0 then 11120
11100 poke p1,peek(p1)and127: p1=p1-1:
   print dl$;
11110 b$=left$(b$,len(b$)-1): goto 11020
11120 gosub 12000: goto 11020
11999 rem****** brummton bei fehler ***
12000 poke 54277,32: poke 54278,255: poke
   54296,15
12010 poke 54273,6: poke 54276,33: for i=0
   to 100: next
12020 poke 54276,0: poke 54296,0: return
12999 rem****** eingabe abfrage *******
13000 b$="jn"
13010 get a$: if a$="" then 13010
13020 for i=1 to len(b$): if a$=mid$(b$,i,1)
   then return
13030 next i: gosub 12000: goto 13010
13999 rem****** zeilenrest loeschen ***
14000 a=39-peek(211)
14010 for i=1 to a: print "□";: next
14020 for i=1 to a: print cl$;: next
14030 return
```

Um mit einer Floppy zu arbeiten, müßt ihr in den Zeilen 5000 und 6010 „open 1,1" in „open 1,8" ändern!

Schneider

```
10 INK 0,9: INK 1,24: PEN 1: BORDER 9:
   MODE 2
15 ON ERROR GOTO 10000
20 n%=1
30 WINDOW #1,2,40,2,2: WINDOW
   #2,42,79,2,2
40 WINDOW #3,1,80,4,23: WINDOW
   #4,1,80,25,25
50 PRINT CHR$(150);STRING$(78,
   CHR$(154));CHR$(156);
60 PRINT CHR$(149);TAB(41);CHR$(149);
   TAB(80);CHR$(149);
70 PRINT CHR$(147);STRING$(78,
   CHR$(154)); CHR$(153);
80 LOCATE 1,24: PRINT STRING$(80,
   CHR$(154));
90 CLS #1: PRINT #1,TAB(10);
   "H□A□U□P□T□ -
   □M□E□N□U□E"
100 CLS #3: LOCATE #3,30,4
110 PRINT #3,"1□:□DATEI EROEFFNEN":
   PRINT #3
120 PRINT #3,TAB(30);"2□:□DATEN
   EINGEBEN": PRINT #3
130 PRINT #3,TAB(30);"3□:□DATEN
   ABFRAGEN": PRINT #3
140 PRINT #3,TAB(30);"4□:□DATEN
```

```
   SUCHEN": PRINT #3
150 PRINT #3,TAB(30);"5□:□DATEI
   ABSPEICHERN": PRINT #3
160 PRINT #3,TAB(30);"6□:□DATEI
   LADEN": PRINT #3
170 PRINT #3,TAB(30);"7□:□
   PROGRAMM BEENDEN": PRINT #3
180 CLS #4: PRINT #4,"Deine Wahl ?"
190 a$=INKEY$: IF a$="" THEN 190 ELSE
   a=VAL(a$)
200 IF a<1 OR a>7 THEN 190
210 IF ms%=0 AND (a>1 AND a<6) THEN
   190
220 IF a<>7 THEN 250
230 CLS #4: PRINT #4,"Bist Du sicher ?
   (j/n)"
240 a$=INKEY$: IF a$="" THEN 240 ELSE
   IF LOWER$(a$)="j" THEN CLS: END:
   ELSE GOTO 90
250 ON a GOSUB 1000,2000,2200,4000,
   5000,6000
260 SOUND 1,100,30,13: GOTO 90
1000 CLS #4: PRINT #4,"Bist Du sicher ?
   (j/n) "
1010 a$=INKEY$: IF a$="" THEN 1010
   ELSE IF LOWER$(a$)<>"j" THEN 90
1020 IF ms%>0 THEN ERASE daten$:
   r%=0: n%=1
1030 CLS #3: CLS #1: CLS #2: PRINT
   #1,TAB(7);"D□A□T□E□I□□
   □E□R□O□E□F□F□N□E□N"
1040 CLS #4: PRINT #4,"Wieviele Felder
   soll ein Datensatz haben ? (max. 8 Felder)"
1050 a$=INKEY$: IF a$="" THEN 1050
   ELSE a=VAL(a$)
1060 IF a<1 OR a>8 THEN 1050 ELSE
   a%=a
1070 LOCATE #3,20,4
1080 FOR n=1 TO a%: PRINT #4,"Name
   von Feld";n;":□";: x=15: y=4: GOSUB
   6400
1090 n$(n)=s$
1100 PRINT #3,TAB(20);n;CHR$(8);
   ". Feld :□";n$(n): PRINT #3
1110 CLS #4: PRINT #4,"Laenge von
   Feld";n;":□";: x=2: y=4: GOSUB 6400
1120 IF s$="" THEN 1110 ELSE
   a(n)=VAL(s$)
1130 IF a(n)>54 OR a(n)<2 THEN 1110
1140 r%=r%+a(n): PRINT #4: NEXT
1145 IF r%<10 THEN r%=10
1150 ms%=(FRE("")/r%)-
   (FRE("")/r%)*0.25
1160 CLS #4: PRINT #4,"Du kannst bis
   zu";ms%"Datensaetze benutzen !"
1170 DIM daten$(ms%,a%)
1180 CALL &BB18: RETURN
2000 CLS #1: CLS #4: PRINT #1,
   TAB(10);"D□A□T□E□N□□□
   E□I□N□G□E□B□E□N"
2010 CLS #3: CLS #2
2020 IF n%-2=ms% THEN PRINT #4,"DIE
   DATEI IST VOLL !";: CALL &BB18:
   RETURN
2040 PRINT #2,DEC$((n%,"# # # #");
   "□von□";DEC$((ms%,"# # # #");
   "□Datensaetzen verbraucht";
2060 PRINT #4,"Mit < ENTER > kehrst Du
   zurueck zum Menue";
2070 LOCATE #3,5,4: FOR n=1 TO a%:
   PRINT #3,TAB(5);n$(n);TAB(20);":□";
2080 x=a(n): y=3: GOSUB 6400
2090 IF s$="" AND n=1 THEN n=a%:
   q=1: GOTO 2110
2100 daten$(n%,n)=s$: PRINT #3
2110 NEXT
2120 IF q=1 THEN RETURN
2130 n%=n%+1
2140 IF n%=2 THEN 2010
2150 g=n%-1
2160 IF daten$(g,1)>daten$(g-1,1) THEN
   2010
2170 FOR n=1 TO a%: c$=daten$(g,n):
   daten$(g,n)=daten$(g-1,n):
   daten$(g-1,n)=c$: NEXT
2180 g=g-1: IF g=1 THEN 2010
2190 GOTO 2160
2200 IF n%=1 THEN RETURN
3000 CLS#1: PRINT #1,TAB(7);
   "D□A□T□E□N□□□
   A□B□F□R□A□G□E□N"
3010 d%=0: c=1: q=0
3020 WHILE q<>1
3030 d%=d%+c
3040 IF d%>n%-1 THEN d%=1
3050 IF d%<1 THEN d%=n%-1
```

Schneider-Tip

Im Listing des Dateiprogramms für den Schneider taucht der Befehl DEC$ auf. Wahrscheinlich haben Sie sich schon über die zweimal geöffnete, aber nur einmal wieder geschlossene Klammer danach gewundert. Behalten Sie diese Schreibweise unbedingt bei, sonst erhalten Sie auf dem 464 einen SYNTAX-ERROR. Hier liegt nämlich ein Fehler im BASIC-Interpreter des Computers vor; deshalb erscheint dieses Kommando auch in keiner offiziellen Befehlsauflistung für den 464. Mit dem DEC$ können Sie Zahlenvariablen zu Strings machen und dabei gleich formatieren. Schauen Sie zur Verwendungsweise des Befehls unter PRINT USING in Ihrem Handbuch nach. Wenn Sie die angegebene Schreibweise benutzen, funktioniert der Befehl einwandfrei.

```
3060 GOSUB 6800 'SCREEN
3070 GOSUB 7000 'KEY
3080 IF n%<2 THEN q=1
3090 WEND
3100 RETURN
4000 CES #1: PRINT #1,TAB(11);
   "D□A□T□E□N□□□
   S□U□C□H□E□N"
4010 IF n%=1 THEN RETURN
4020 CLS #3: LOCATE #3,30,4: FOR j=1
   TO a%: PRINT #3,TAB(30);j;
   CHR$(8);".";n$(j): PRINT #3: NEXT
4030 d%=1: c=1: q=0
4040 CLS #4: PRINT #4,"In welchem Feld
   willst Du suchen ?"
4050 a$=INKEY$: IF a$="" THEN 4050
   ELSE v=VAL(a$)
4060 IF v<1 OR v>a% THEN 4050
4070 CLS #4: PRINT #4,"Suchwort in
   Feld";v;":□";
4080 x=a(v): y=4: GOSUB 6400: sw$=s$
4090 WHILE q<>1 AND c<>0
4100 q=-1
4110 GOSUB 5500
4120 IF c<>0 THEN GOSUB 6800: GOSUB
   7000
4125 WEND
4130 IF q=1 THEN RETURN
4140 CLS #4: PRINT #4,"Keinen Datensatz
   gefunden": CALL &BB18: RETURN
4500 CLS #4: PRINT #4,"Willst Du diesen
   Datensatz wirklich loeschen ? (j/n)"
4510 a$=INKEY$: IF a$="" THEN 4510
   ELSE IF LOWER$(a$)<>"j" THEN
   RETURN
4520 IF d%=n%-1 THEN 4560
4530 FOR i=d%+1 TO n%: FOR j=1 TO a%
4540 daten$(i-1,j)=daten$(i,j)
4550 NEXT j: NEXT i
4560 n%=n%-1: d%=d%-c: RETURN
5000 CLS #1: CLS #3: PRINT #1,TAB(4);
   "D□A□T□E□I□□□□A□B□S□
   P□E□I□C□H□E□R□N"
5010 CLS #4: PRINT #4,"Bitte Dateinamen
   angeben :□";: x=16: y=4: GOSUB 6400
5020 IF s$="" THEN 5010
5030 LOCATE #3,17,10: PRINT #3,"D□A
   □T□E□I□□□W□I□R□D□□□
   A□B□G□E□S□P□E□I□C□H□E
   □R□T□ !"
5040 WINDOW SWAP 0,4
5050 OPENOUT s$
5060 PRINT #9,ms%,n%,a%
5070 FOR i=1 TO a%: WRITE #9,n$(i);a(i):
   NEXT
5080 FOR i=1 TO n%-1: FOR j=1 TO a%
5090 PRINT #9,daten$(i,j)
5100 NEXT j: NEXT i
5110 CLOSEOUT
5120 WINDOW SWAP 4,0
```

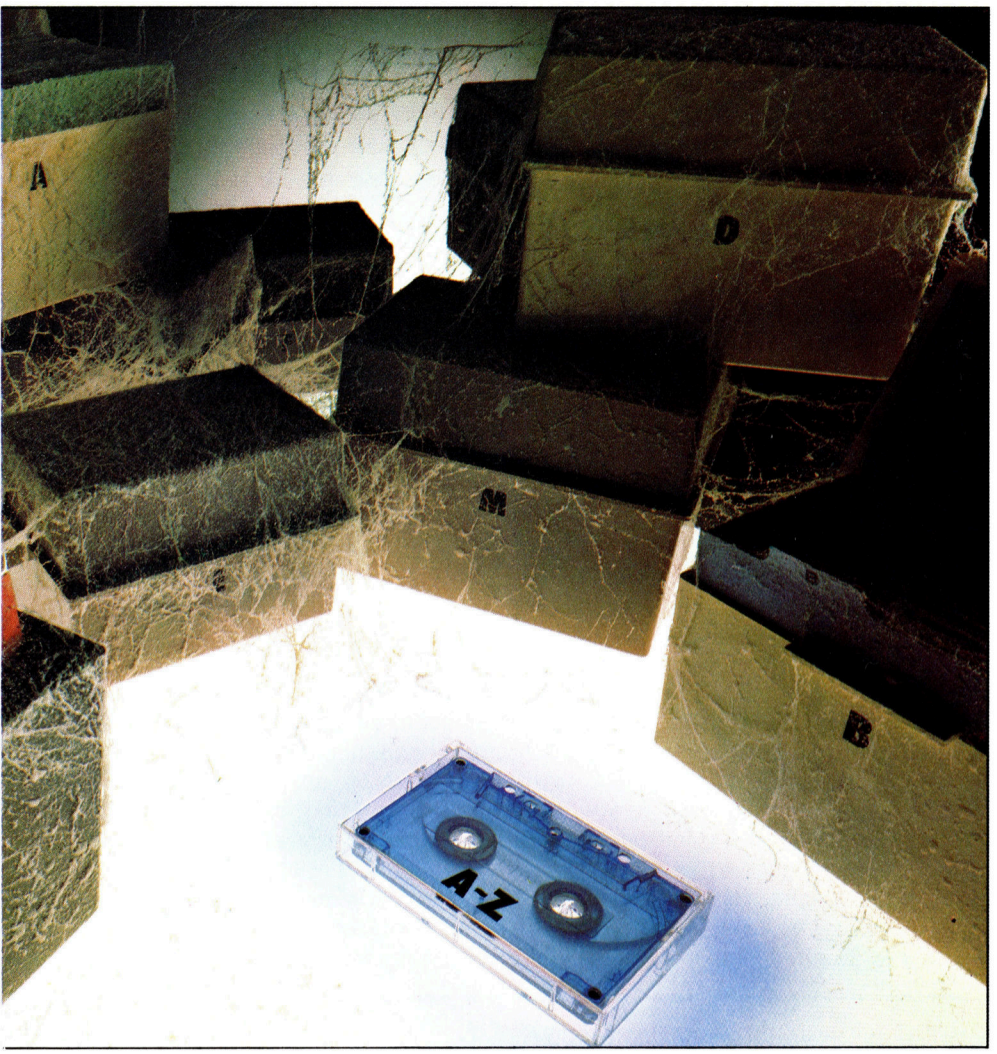
```
5130 RETURN
5500 t=0
5505 d%=d%+c
5510 IF d%<1 OR d%>n%-1 THEN
   c=-c: t=t+1: d%=d%+c
5520 IF sw$=daten$(d%,v) THEN RETURN
5530 IF t<2 THEN 5505
5540 c=0
5550 RETURN
6000 CLS #1: PRINT #1,TAB(10);
   "D□A□T□E□I□□□
   L□A□D□E□N";
6010 CLS #3: CLS #4: PRINT #4,"Bitte
   Dateinamen angeben :□";: x=16: y=4:
   GOSUB 6400
6020 LOCATE #3,20,10: PRINT #3,
   "D□A□T□E□E□I□□□
   W□I□R□D□□□□
   G□E□L□A□D□E□N□ !"
6030 IF ms%>0 THEN ERASE daten$
6040 WINDOW SWAP 0,4
6050 OPENIN s$
6060 INPUT #9,ms%,n%,a%
6070 FOR i=1 TO a%: INPUT #9,n$(i),a(i):
   NEXT
6080 DIM daten$(ms%,a%)
6090 FOR i=1 TO n%-1: FOR j=1 TO a%
6100 LINE INPUT #9,daten$(i,j)
6110 NEXT j: NEXT i
6120 CLOSEIN
6130 WINDOW SWAP 4,0
6140 RETURN
6200 CLS #4: PRINT #4,"Ist der Drucker
   angeschlossen ? (j\n)"
6210 a$=INKEY$: IF a$="" THEN 6210
   ELSE IF LOWER$(a$)<>"j" THEN
   RETURN
6220 WIDTH 80
6230 PRINT #8,CHR$(10);CHR$(13)
6240 FOR i=1 TO n%-1: FOR j=1 TO a%
6250 PRINT #8,n$(j);TAB(20);" :□";
   daten$(i,j)
6260 NEXT j
6270 PRINT #8,CHR$(10): NEXT i
6280 RETURN
6400 PRINT #y,CHR$(143)+CHR$(8);
6410 s$="":FOR t=1 TO x+1
6420 k$=INKEY$: IF k$="" THEN 6420
   ELSE k=ASC(k$)
6430 IF k=127 AND t>1 THEN t=t-1:
   PRINT #y,CHR$(16)+CHR$(8)+
   CHR$(143)+CHR$(8);:
   s$=LEFT$(s$,t-1): GOTO 6420
6440 IF k=13 THEN t=x+1: PRINT
   #y,CHR$(16);: GOTO 6490
6450 IF k<>13 AND t=x+1 THEN 6420
6460 IF k<32 OR k>126 THEN 6420
6470 s$=s$+CHR$(k)
6480 PRINT #y,CHR$(k);CHR$(143)+
   CHR$(8);
6490 NEXT: RETURN
6600 CLS #4: PRINT #4,"Welches Feld ?"
6610 a$=INKEY$: IF a$="" THEN 6610
   ELSE IF VAL(a$)>a% AND
   VAL(a$)<1 THEN 6610
6620 r=VAL(a$)
6630 CLS #4: PRINT #4,"Aenderung :□";:
   x=a(r): y=4: GOSUB 6400
6640 IF s$="" THEN 6630 ELSE
   daten$(d%,r)=s$
6650 IF r<>1 THEN RETURN
6660 IF d%=1 THEN 6710
6670 IF daten$(d%,1)>daten$(d%-1,1)
   THEN 6700
6680 FOR t=1 TO a%: c$=daten$(d%,t):
   daten$(d%,t)=date$(d%-1,t):
   daten$(d%-1,t)=c$: NEXT
6690 d%=d%-1: IF d%=1 THEN 6740
   ELSE 6670
6700 IF d%=n%-1 THEN 6740
6710 IF daten$(d%,1)<daten$(d%+1,1)
   THEN 6740
6720 FOR t=1 TO a%: c$=daten$(d%,t):
   daten$(d%,t)=daten$(d%+1,t):
   daten$(d%+1,t)=c$: NEXT
6730 d%=d%+1: GOTO 6700
6740 d%=d%-c: RETURN
6800 CLS #3: CLS #2
6810 PRINT #2,TAB(10);" Datensatz
   Nummer";d%
6820 LOCATE #3,5,4
6830 FOR s=1 TO a%: PRINT #3,TAB(5);
   n$(s);TAB(20);":□";daten$(d%,s):
   PRINT #3
```

```
6840 NEXT
6850 PRINT #4,"V(orwaerts)□□□
     Z(urueck)□□□M(enue)□□□
     L(oeschen)□□□A(endern)□□□
     D(rucken)";
6860 RETURN
7000 a$=INKEY$: IF a$="" THEN 7000
     ELSE k=ASC(a$) AND &5F
7010 IF k=86 OR k=13 THEN c=1:
     RETURN
7020 IF k=77 THEN q=1: RETURN
7030 IF k=68 THEN GOSUB 6200: RETURN
7040 IF k=65 THEN GOSUB 6600: RETURN
7050 IF k=90 THEN c=-1: RETURN
7060 IF k=76 THEN GOSUB 4500: RETURN
7070 GOTO 7000
10000 IF ERR<>7 THEN PRINT ERR,ERL:
     STOP
10010 CLS #4: PRINT #4,
     "Dateidimensionierung nicht moeglich!"
10020 ms%=0: r%=0: n%=1: CALL
     &BB18: GOTO 90
```

Sinclair

```
5 LET R=0: LET U=0: LET V=1
10 BORDER V: PAPER V: INK 7: POKE
   23609, 20: POKE 23658,8
100 CLS : PRINT INVERSE V;AT V,6;"□H□
    A□U□P□T□M□E□N□U□E"
110 PRINT AT 5,6;"1 :— Datei eroeffnen
    ""TAB 6;"2 :— Daten eingeben"" TAB 6;
    "3 :— Daten abfragen""TAB 6;"4 :— Daten
    suchen""TAB 6;"5 :— Speichern""TAB 6;
    "6 :— Laden""TAB 6;"7 :—Programm
    beenden"; #V;TAB 6;"— WAEHLE EINE
    ZAHL —"
500 LET I$=INKEY$: IF I$="" THEN GOTO
    500
510 IF I$<"1" OR I$>"7" THEN GOTO
    500
520 IF R=U AND I$<>"1" AND I$<>
    "6" AND I$<>"7" THEN GOTO 500
530 BEEP .1,10: CLS : GOSUB (CODE
    I$-48)*1000: GOTO 100
1000 PRINT AT 7,9;"BIST DU SICHER ?":
     PAUSE U: IF INKEY$="" THEN
     GOTO 1000
1010 IF INKEY$<>"J" AND INKEY$
     <>"j" THEN RETURN
1020 PRINT INVERSE V;AT 10,6;"□DATEI
     EROEFFNEN□"
1030 INPUT AT 0,0;"Anzahl der Felder
     (1-8)?□";A: IF A<1 OR A>8 THEN
     GOTO 1030
1040 DIM A(A): DIM B(A+V): DIM
     N$(A,10):LET T=U: FOR N=V TO A
1050 INPUT AT 0,0;"Name
     Feld□";(N);"□?□"; LINE N$(N)
1060 INPUT AT V,0;"Laenge
     Feld□";(N);"□?□";A(N): IF
     A(N)>50 THEN GOTO 1060
1070 LET B(N)=T: LET T=T+A(N): NEXT
     N:LET B(N)=T
1080 PRINT AT 16,2;"MAX. Anzahl Datenfelder
     □=□";INT(((PEEK 23730+256*PEEK
     23731)-29500)/T)
1090 INPUT "Wieviele brauchst Du ?□";
     R: DIMA$(R,T): RETURN
2000 LET C=V
2010 IF A$(C,V)="□" THEN GOTO 2100
2020 IF C=R THEN GOTO 2500
2030 LET C=C+V: GOTO 2010
2100 PRINT AT 0,0;"Du hast□";C-V;
     "□von□";R;"□Datensaetzen ":PRINT
     "Belegt"
2110 FOR N=V TO A: PRINT INVERSE
     V;AT V+N*2,U;N$(N); INVERSE 0;AT
     V+N*2,12; FLASH V;"?": INPUT "(Max.
     □";(A(N));"□Zeichen)", LINE
     A$(C,B(N)+V TO B(N+V)): PRINT AT
     V+2*N,12;A$(C,B(N)+V TO B(N+V)):
     NEXT N
2120 FOR F=V TO 150: NEXT F: IF C=V
     THEN RETURN
2130 LET N=C
2140 IF A$(C)>=A$(C-V) THEN RETURN
2150 LET X$=A$(C): LET A$(C)=A$
     (C-V): LET A$(C-V)=X$: LET
     C=C-V: IF C=V THEN RETURN
2160 GOTO 2140
2500 CLS : PRINT FLASH 1;AT 10,6;"□D□
     A□T□E□I□□□V□O□L□L□":
     FOR F=V TO 400: NEXT F: RETURN
3000 LET D=V: IF A$(V,V)="□" THEN
     RETURN
3010 IF D=U THEN LET D=V
3015 IF D-V=R THEN LET D=D-V
3020 IF A$(D,V)="□" THEN LET D=D-V
3030 GOSUB 9500
3040 IF OP=V THEN LET D=D+V: GOTO
     3010
3050 IF OP=2 THEN LET D=D-V: GOTO
     3010
3060 IF OP=3 THEN RETURN
3070 IF OP=4 THEN GOSUB 8000
3080 IF OP=5 THEN LET MD=V: GOSUB
     9000:IF D=U THEN RETURN
3090 GOTO 3030
4000 FOR N=V TO A: PRINT INVERSE V;AT
     N*2,9;N; INVERSE U; TAB 11;":—□";
     N$(N): NEXT N
4010 PRINT ""In welchem Feld□
     (1 bis□":A:") suchen?"
4020 IF INKEY$="" THEN GOTO 4020
4030 LET Y$=INKEY$: IF CODE Y$<49 OR
     CODE Y$>48+A THEN GOTO 4030
4040 BEEP .1,10: LET Z=VAL Y$: PRINT
     ""In Feld□";Z;"□welches Suchwort?":
     DIM Z$(V, A(Z)): INPUT LINE Z$(V)
4044 CLS : LET K=V
```

```
4045 IF A$(K,B(Z)+V TO B(Z+V))=Z$(V)
     THEN GOTO 4050
4046 IF K=R OR A$(K,V)="□" THEN
     CLS: PRINT AT 9,3;"□NICHTS
     MIT□";Z$(V),'TAB 10;"IN FELD□";
     Z;"□ZU FINDEN"
4047 LET K=K+V: GOTO 4045
4050 LET D=V: LET PM=V: LET MO=V
4060 IF D>R THEN LET D=PM
4070 IF D=U THEN LET D=PM
4080 IF A$(D,V)="□" THEN LET D=PM
4090 IF A$(D,B(Z)+V TO B(Z+V))
     <>Z$(V) THEN LET D=D+MO:
     GOTO 4060
4100 GOSUB 9500
4110 LET PM=D
4120 IF OP=V THEN LET MO=V: LET
     D=D+MO: GOTO 4060
4130 IF OP=2 THEN LET MO=-V: LET
     D=D+MO: GOTO 4060
4140 IF OP=3 THEN RETURN
4150 IF OP=4 THEN GOSUB 8000
4160 IF OP=5 THEN LET DF=U: LET
```




```
    MD=2: GOSUB 9000: IF DF=V OR
    A$(V,V)="□" THEN RETURN
4170 GOTO 4100
5000 INPUT "Dateiname?□□"; LINE
    Q$:IF LEN Q$<V OR LEN Q$>10
    THEN GOTO 5000
5010 SAVE Q$ LINE 10: RETURN
6000 PRINT AT 8,U;"Welche Datei laden? (Nur
    'ENTER' laedt die erste Datei vom Band)"
6010 INPUT LINE X$: IF LEN X$>10
    THEN GOTO 6010
6020 PRINT AT 13,U;" 'PLAY' AM
    RECORDER DRUECKEN": LOAD X$
7000 PRINT AT 10,8;"Bist Du sicher?":IF
    INKEY$="" THEN GOTO 7000
7010 IF INKEY$<>"J" AND INKEY$
    <>"j" THEN RETURN
7020 RANDOMIZE USR U
8000 INPUT AT U,U;"AENDERN Feld Nr.
    (1 bis□";(A);") ? ";J: IF J>A THEN
    GOTO 8000
8010 PRINT FLASH V;AT V+2*J,12;A$
    (D,B(J)+V TO B(J+V)): INPUT AT
    U,U;"Bitte neues Feld eingeben", LINE
    A$(D,B(J)+V TO B(J+V)): PRINT AT
    V+2*J,12;A$(D,B(J)+V TO B(J+V))
8020 IF D=R THEN LET J=-V:
    GOTO 8070
8030 IF D=V THEN LET J=V: GOTO 8060
8040 IF A$(D)>A$(D+V) THEN LET J=V
8050 IF A$(D)<A$(D-V) THEN LET
    J=-V
8060 IF A$(D+V,V)="□" AND J=V THEN
    GOTO 8500
8070 IF J=V THEN GOTO 8100
8080 IF A$(D)>=A$(D-V) THEN GOTO
    8500
8090 LET X$=A$(D): LET A$(D)=
    A$(D-V): LET A$(D-V)=X$: LET
    D=D-V: GOTO 8020
8100 IF A$(D)<=A$(D+V) THEN
    GOTO 8500
8110 LET X$=A$(D): LET A$(D)=
    A$(D+V): LET A$(D+V)=X$: LET
    D=D+V: GOTO 8020
8500 BEEP .1,10: PRINT AT U,U;"Eintrag
    Nr.□";D;"□□□": RETURN
9000 PRINT AT 19,U;"LOESCHEN? BIST
    DU SICHER?": PAUSE U
9010 IF INKEY$="" THEN GOTO 9010
9020 IF INKEY$<>"J" AND INKEY$<>
    "j" THEN PRINT AT 19,U;"□□□□□
    □□□□□□□□□□□□□□□
    □□□□□□□□□□□□□": BEEP
    .1,10: RETURN
9030 PRINT AT 19,U;"□ICH LOESCHE
    □□□□□□□□□□□□□□□
    □□□□□"
9035 LET DD=D
9040 IF D=R THEN LET DD=DD-V:
    GOTO 9060
9050 IF A$(D+V,V)<>"□" THEN LET
    A$(D)=A$(D+V): LET D=D+V:
    GOTO 9040
9060 LET A$(D)="": FOR F=V TO 100:
    NEXT F: PRINT AT 19,U;"□□□
    □□□□□□□"
9070 LET D=DD: IF A$(V,V)="□" THEN
    LET D=U: RETURN
9072 IF D=U THEN LET D=V
9075 IF A$(D,V)="□" THEN LET D=D-V
9080 IF MD=V THEN RETURN
9090 LET K=V
9100 IF A$(K,B(Z)+V TO B(Z+V))=Z$(V)
    THEN GOTO 9130
9110 IF K=R OR A$(K,V)="□" THEN LET
    DF=V: GOTO 4046
9120 LET K=K+V: GOTO 9100
9130 LET DD=D: LET PA=V
9140 IF A$(DD,V)="□" OR DD=U THEN
    LET PA=2: LET DD=D: LET MO=
    MO-V
9150 IF A$(DD,B(Z)+V TO B(Z+V))=Z$(V)
    THEN LET D=DD: RETURN
9160 LET DD=DD+MO: GOTO 9140
9500 PRINT AT U,U;"Datei Nummer□";D;
    "□□": FOR N=V TO A: PRINT
    INVERSE V;AT V+2*N,U;N$(N); INVERSE
    U;TAB 12;A$(D,B(N)+V TO
    B(N+V)): NEXT N
9510 PRINT INVERSE V;AT 20,U;"V(orwaerts)
    □Z(urueck)□□M(enue)□□A(endern)
    □□□L(oeschen)□D(rucken)"
9520 IF INKEY$="" THEN GOTO 9520
9530 LET V$=INKEY$: IF V$="D" THEN
    COPY :LPRINT : LPRINT : LPRINT :
    GOTO 9520
9540 LET OP=U: IF V$="V" THEN LET
    OP=V:LET MO=V
9550 IF V$="Z" THEN LET OP=2: LET
    MO=-V
9560 IF V$="M" THEN LET OP=3
9570 IF V$="A" THEN LET OP=4
9580 IF V$="P" THEN LET OP=5
9590 IF OP=U THEN GOTO 9520
9600 BEEP .1,10: RETURN
```

Die lange Leitung

Obwohl sie bereits viele Jahre eine gewichtige Rolle in der Telekommunikationsindustrie bekleiden, erreichten die Modems erst jetzt ihren großen Vorstoß sowohl im geschäftlichen als auch im privaten Bereich.

Modems sind in den USA das große Geschäft. In Deutschland und Großbritannien galten sie lange als Luxusartikel am Computer. Die Gründe hierfür lassen sich in zwei Aussagen zusammenfassen: Zunächst bringen die Verbindungsgebühren des Telefonnetzes, speziell während der Hauptgeschäftszeit, erhebliche Kosten mit sich. Jemand, der sein Modem wöchentlich mehrere Stunden einsetzt, staunt oft über die folgende Telefonrechnung, die plötzlich einen wundersam angewachsenen Betrag ausweist.

Der zweite Grund liegt im schleppenden Ausbau spezieller Kommunikationsnetze. In England beispielsweise reagierte die „British Telecom" zu langsam und führte ihr System nur stückchenweise ein. Wegen des fehlenden Standards entwickelten sich private Dienste, die alle mit verschiedenen Protokollen arbeiteten. Ein weiteres Hindernis für Modemanbieter und Benutzer ist die, in Deutschland wie in Großbritannien vorgeschriebene, technische Abnahme durch die entsprechenden Behörden, deren langwierige Genehmigungsverfahren und damit verbundene Kosten die Verbreitung des Modems bremsten. Technische Neuentwicklungen brauchten zum Teil Jahre, um durch diesen Verordnungsberg hindurch endlich den Anwender zu erreichen.

In den USA sind Ortsgespräche kostenlos und fast alle Datenbanken und Kommunikationssysteme arbeiten nach dem „Hayes-Standard". Der Hayes-Standard ist ein System von Protokollen, das von vielen Herstellern anderer Modems übernommen wurde. Dieser Standard ist weder in Deutschland noch in England behördlich zugelassen und der Gebrauch eines Hayes-Modems somit strafbar. Eigentlich aber sollte ein Modem diese und eine Anzahl anderer Leistungen aufweisen, um dem Anwender hierzulande maximale Flexibilität in diesem verwirrenden Kommunikations-Dschungel zu bieten. Ungeachtet aller Probleme steigen die Verkaufszahlen von Modems weiter an. Lassen sie uns nun einen Blick auf die Leistungsmerkmale einiger angebotener Modems werfen.

Pace nightingale

Für den Gebrauch mit
Acorn B, Commodore 64 und Schneider CPC

Übertragungsrate
Zum Senden und Empfangen stehen je sieben Übertragungsraten von 75 bis 9800 Baud zur Auswahl

Nightingale

Das „Pace Nightingale" kann, obwohl speziell für den Acorn B konzipiert, an jeden Computer mit RS232-Schnittstelle angeschlossen werden. Wie auch beim Modem-1000-Paket ist im Lieferumfang ein EPROM enthalten, das, in einem der freien ROM-Sockel des Acorn B eingesetzt, die erforderliche Software bereithält. Obgleich viele der Modemfunktionen über das menügesteuerte Programm anzuwählen sind, muß die Übertragungsrate mittels Schalter an der Vorderseite des Gerätes eingestellt werden.

Das Programm ermöglicht den Betrieb des Computers in Prestel- oder Terminalemulations-Modus. In beiden Betriebsarten können Daten übertragen und im Speicher des Computers bearbeitet werden.

Zum Nightingale gehört das umfangreichste Handbuch aller hier beschriebenen Modems. Angesichts der großen Zahl von zu Lehr- und Lernzwecken eingesetzten Acorn-Computer produzierte der Hersteller ein Handbuch, das genau für diesen Markt zugeschnitten ist. Es bietet dem Leser mehr Informationen, als den Modems für Heimcomputer beiliegen.

VTX 5000

Ungeachtet seiner wechselvollen Geschichte ist das VTX 5000, hergestellt von OE Ltd., zu dem Standard-Modem für den Spectrum geworden. Es wurde ursprünglich für Prism entwickelt, mit dem Ziel, die Anwenderzahlen von Micronet zu erhöhen, von dem die Firma große Anteile besaß. Auch nach dem Zusammenbruch von Prism Anfang 1985 und der Übernahme des Modems durch Modem House ging der Verkauf nicht zurück.

Auf der Vorderseite des Modems sind eine Netzlampe, eine Bereitschaftslampe (die eine bestehende Verbindung anzeigt) und der dazugehörende Schalter installiert. Ein anderer Schalter dient zur Auswahl einer der drei möglichen Betriebsarten: Micronet, TX (senden) oder RX (Empfangen). An der Rückseite befindet sich eine Telefonsteckdose und ein Anschlußkabel mit BT-Stecker, der direkt mit der Telefonwandsteckdose verbunden werden kann.

Das VTX 5000 enthält die erforderliche Software in einem ROM. Im Gerät sind zwei Platinen untergebracht, eine übernimmt die serielle Übertragung und Decodierung, auf der anderen ist die Elektronik zur Verbindung mit dem Spectrum aufgebaut, wie der ACIA-Chip und das ROM mit dem Betriebssystem. Diese Methode der Modemansteuerung konfrontiert

VTX 5000

Für den Gebrauch mit
Sinclair Spectrum

Übertragungsrate
75/1200 Band

den Spectrum-Besitzer mit einem Problem: Das Modem-ROM blendet das BASIC-ROM des Computers aus und belegt dessen Speicherbereich. Fehler treten bei Betrieb des Modems zusammen mit dem Interface 1 auf, etwa zum Sichern der empfangenen Daten. Das Interface 1 schaltet ebenfalls das BASIC-ROM aus und belegt diesen Speicherplatz, wodurch sich die beiden Geräte nicht gemeinsam einsetzen lassen. Die Modem-Software ist menügesteuert.

Modem 1000

Aus dem gleichen Hause wie das VTX 5000 kommend, bietet das Modem 1000 viele Vorzüge in Verbindung mit dem Spectrum. Doch ist das Modem 1000 auch zum Betrieb an anderen Rechnern gedacht und besitzt kein ROM mit rechnerspezifischer Software.

Die externen Kontrollen sind mit denen des VTX 5000 identisch: Eine Bereitschaftsanzeige mit Schalter und ein dreistufiger Schalter für die Betriebsart. An der Rückseite sind die Telefonanschlüsse und Daten-Ein/Ausgänge zum Computer untergebracht. Da das Modem 1000 nicht über den Computer mit Strom versorgt wird, sind Netztransformator und Netzkabel in das Gerät eingebaut.

Die Elektronik zur seriellen Übertragung und der ACIA-Chip sind weitgehend identisch mit den im VTX 5000 verwendeten Schaltungen. Zum Betrieb des Modem ist natürlich Software erforderlich. Für den Acorn B gibt es ein ROM, das in einen der freien Steckplätze im Computer einzusetzen ist. Das darin enthaltene Programm ähnelt der Spectrum-Version. Menügesteuert unterstützt es die „Log on"-Prozedur zum Micronet und das Übertragen von Programmen und Nachrichten.

Aufgrund der internen Struktur des Commodore 64 ist die zum Betrieb des Modem 1000

Modem 1000

Für den Gebrauch mit
allen bekannten Heimcomputern

Übertragungsrate
75/1200 Baud

benötigte Schnittstelle komplexer als die des VTX 5000. Sie besteht aus einer Platine, die in den Modulschacht des C64 eingesteckt wird. Darauf befinden sich außer dem Programm, das in den Bereich des BASIC-ROMs eingeblendet wird, auch spezielle Anpassungsschaltungen, die den Commodore-Zeichensatz in den allgemein gebräuchlichen ASCII-Kommunikations-Standard umwandeln. Ein anderer Teil der Schaltungen erzeugt die normalen RS232-Signale zum Steuern des Modems.

KDS Communicator 104

Für den Gebrauch mit Schneider CPC

Übertragungsraten 75/1200, 300/300, 1200/75, 1200 Halbduplex

Communicator 104

Dieses Modem, hergestellt von KDS Electronics, wurde speziell für den Betrieb an Schneider-Computern entwickelt. Wie beim VTX 5000 ist der Erweiterungsanschluß an der Rückseite des Rechners der Adressat des Modems, jedoch übernimmt ein externer Transformator die Stromversorgung. Wie die übrigen hier vorgestellten Modems wird auch dieses zwischen Wanddose und Telefon geschaltet. Um die Verbindung zum Computer am anderen Ende der Leitung herzustellen, wählt man am Telefon die entsprechende Nummer und schaltet dann das Modem ein.

Der Communicator unterscheidet sich von den anderen Modems durch zwei Digitalanzeigen an der Vorderseite, die zusätzliche Informationen präsentiert, etwa die gerade gewählte Telefonnummer, oder ob der Computer gerade auf das Modem-ROM zugreift. Die menügesteuerten Programme sind mit sehr vielen Optionen ausgestattet und in einem ROM im Modem gespeichert. Aus dem Modusmenü wählt man die Betriebsart – zum Beispiel Prestel oder Bulletin Board – wonach sich das Modem automatisch auf die richtige Übertragungsrate einstellt, die man nach Belieben wieder verändern kann. Außerdem berücksichtigt die Software Funktionen wie automatische Wahl, Autoantwort und auch die Trägertonerkennung.

Im Communicator ist eine große Anzahl zusätzlicher Systemkommandos implementiert. Dazu gehören, neben anderen, Kommandos zum Einschalten des Modem-ROMs, Einstellen der Baudrate und der Länge des Datenworts und für die Konfiguration des Druckers zum Erstellen von Hardcopies während der Verbindung. Das Hauptproblem des Communicators liegt darin, daß ein Telefon nicht zum normalen Fernsprechen angeschlossen sein kann, während das Modem aktiv ist. Es ist daher notwendig, entweder das Telefon oder das Modem mit der Telefonleitung zu verbinden.

WS3000

Für den Gebrauch mit jedem Computer mit RS232-Schnittstelle

Übertragungsrate 75/1200, 300/300, 1200/1200 Halb- oder Vollduplex

WS3000

Im Angebot der Miracle Technology steht diese Modem-Serie an der Spitze des Spektrums an Microcomputer-Zubehör – sie deckt damit fast alle vom Anwender benötigten Funktionen ab.

Die Serie WS3000 umfaßt drei Geräte, die insgesamt einen weiten Bereich unterschiedlicher Übertragungsraten und Protokolle beinhalten, von 75/1200 Baud Halbduplex (für den Zugriff auf Prestel) bis 1200/1200 Baud Vollduplex Übertragung für den direkten Datenaustausch zwischen den Benutzern. An der Vorderseite gibt eine Reihe von Anzeigelampen Informationen über den Betriebszustand des Gerätes. Das sind beispielsweise die Operationen auf den RS232-Leitungen.

Jedes der drei Geräte besitzt Hilfsroutinen, die automatische Wahl, Autoantwort und interne Geschwindigkeitsanpassung der RS232 unterstützen. Dazu arbeitet die WS3000-Serie auch nach dem Hayes-Standard, was unter Umständen sehr nützlich für den professionellen Einsatz sein kann.

Kurze Unterbrechung

Die Zeitsteuerung ist eine der wichtigsten Betriebssystemkomponenten. Wie die meisten anderen Heimcomputer arbeitet auch der Schneider CPC mit Interrupts und Ereignissen.

Die Zeitsteuerung des Betriebssystems ist für Abläufe zuständig, die zu bestimmten Zeitpunkten ausgelöst werden müssen. Auf den Schneider-Geräten arbeitet diese Zeitsteuerung nach zwei verschiedenen Methoden: mit Hardware-„Interrupts" und softwaregesteuerten „Ereignissen".

Interruptsignale werden außerhalb des Z80-Microprozessors erzeugt. Sie unterbrechen die laufende Programmausführung und lösen bestimmte Abläufe aus.

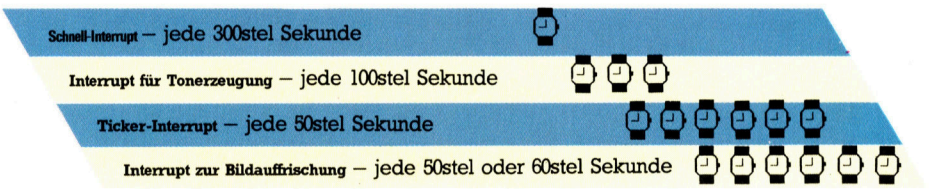
Der Z80-Prozessor des CPC wurde für Interrupts des Modus 1 programmiert. Dabei wird die Programmsteuerung bei Empfang eines Interrupts an die Interrupt-Bearbeitungsroutine (bei $0038) übergeben. Theoretisch kann jedes Gerät jederzeit Interrupts erzeugen und so die Aufmerksamkeit des Prozessors auf sich lenken. Auf Schneider-Geräten ohne Erweiterung gibt es jedoch nur den Typ des „Timer-Interrupts".

Der Timer-Interrupt wird von der ULA jede Dreihundertstelsekunde direkt mit der Systemuhr (4 MHz Taktfrequenz) erzeugt. Die interne Zeitsteuerung (die beispielsweise von den BASIC-Befehlen AFTER und EVERY benötigt wird) ist eine Funktion dieser Interruptart.

Timer-Interrupts werden direkt von den „Kernroutinen" der Firmware bearbeitet, die je nach Interruptebene (siehe Bild) weitere Firmwareroutinen aktiviert. Hier die vier Signale im einzelnen:

- Der „Schnell-Interrupt" tritt auf, wenn Kernroutinen einen Timer-Interrupt empfangen.
- Der „Interrupt für Tonerzeugung" tritt nach jedem dritten Timer-Interrupt ein und wird von der Firmware zur Synchronisierung der Datenübertragung des Klang-Chips eingesetzt. Der Anwender kann nur indirekt (durch Zählen der Schnell-Interrupts) feststellen, wann diese Art von Interrupt eintritt.
- Der „Ticker-Interrupt" tritt nach jedem sechsten Timer-Interrupt ein. Dabei löst die Firmware eine Tastaturprüfung aus.
- Der „Interrupt zur Bildauffrischung" tritt nach jedem fünften oder sechsten Timer-Interrupt auf (je nach angeschlossenem Bildschirm) und wird für alle zeitabhängigen Anzeigevorgänge eingesetzt.

Die Firmware unterhält einen Zähler für Schnell-Interrupts, der auch dem Anwender zur Verfügung steht. Damit lassen sich zeitabhängige Schleifen anlegen, die nicht von einem Programm unterbrochen werden müssen. Der Zähler kann durch den Aufruf von KL_TIME_SET gesetzt werden (der für den Zeitgeber bestimmte Wert wird im Registerpaar HL übergeben). Der Aufruf von KL_TIME_PLEASE liefert jederzeit in HL den aktuellen Zählerwert. Unser Programmbeispiel zeigt den Einsatz dieser beiden Routinen. Das Modul fragt während des angegebenen Zeitraumes die Tastatur ab und löst dann einen Rücksprung aus.



Der Prozessor der Schneider-Geräte kann das Betriebssystem auf vier Arten unterbrechen. Die „Timer-Interrupts" treten in regelmäßigen Abständen auf und synchronisieren die normalen OS-Funktionen. Sie steuern die Tastaturabfrage, die Bildschirmdarstellung und die Datenübertragung des Tonerzeugungschips.

Sie können aber auch direkt mit Timer-Interrupts arbeiten, wenn Sie den Einsprungspunkt RST 7 bei $0048 patchen. Der Patchcode sollte zuerst den bestehenden Drei-Byte-Eintrag kopieren und dort den neuen Eintrag (normalerweise ein Sprung auf eine Speicheradresse) einsetzen. Dieser neue Eintrag muß verschiebbar sein, damit das Modul auch dann funktioniert, wenn eine zweite Routine ihn wiederum an eine andere Stelle kopiert.

Die neue Routine kann einen beliebigen Ablauf ausführen (z. B. eine Uhr aktualisieren) und sollte danach aber die ursprüngliche Routine mit einem Sprung auf deren neue Adresse aktivieren. Dieser Ablauf stellt sicher, daß zeitabhängige Interruptfunktionen, wie etwa Tastaturabfragen, auch weiterhin ausgeführt werden. In früheren Folgen hatten wir schon beschrieben, wie sich „Code-Keile" in Interruptroutinen einsetzen lassen.

Bisher gingen wir davon aus, daß die ULA die Interrupts veranlaßt. Interrupts werden aber auch von externen Peripheriegeräten erzeugt. So könnte beispielsweise eine serielle Schnittstelle nach jedem empfangenen Zeichen einen Interrupt auslösen, dessen Bearbeitungsroutine das Zeichen dort „abholt".

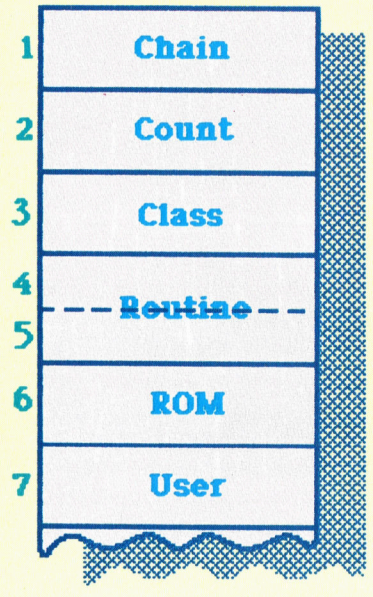
Ereignisse sind „Software-Interrupts". Mit Hilfe des Ereignisblocks behält das Betriebssystem die Übersicht über die aktivierten Ereignisse. Dabei liefern sieben aufeinanderfolgende Bytes die Adresse der Ereignisroutine, auf die sich der Block bezieht, die Zahl der Ereignisse, die noch ausgeführt werden sollen und die Ereignisklasse.

Die Kernroutine muß daher zwischen Timer-Interrupts und externen Interrupts unterscheiden können. Normalerweise ist der Z80 nach Empfang eines Unterbrechungssignals vor weiteren Interrupts abgeschirmt, damit zeitkritische Interruptroutinen nicht gestört werden. Die Kernroutinen zur Interrupt-Bearbeitung schalten den Interruptempfang jedoch wieder an. Wenn dabei das Interruptsignal zum Z80 immer noch auf Null steht (Interruptanforderung), wird ein zweiter Interrupt erzeugt. Da ein zweiter Timer-Interrupt in diesem Stadium aber nicht möglich ist, muß eine externe Interruptanforderung vorliegen – vorausgesetzt, das Interruptsignal bleibt bis zum Ende auf Null. Nach Identifizierung eines externen Interrupts wird die Steuerung an $003B übergeben. Diese Speicherstelle sollte daher mit einem Sprung auf die Interrupt-Bearbeitungsroutine gepatcht werden, die das anfragende Gerät „bedient".

Das System unterstützt „Nicht-Maskierbare Interrupts" (NMI) und Standard-Interrupts. NMIs lassen sich im Gegensatz zu Standard-Interrupts nicht abschalten. NMIs können Systemabläufe mit festem Zeitrahmen empfindlich stören und sollten daher nicht verwandt werden. So schalten beispielsweise die Firmwareroutinen zur Steuerung von Cassetten- und Diskettenlaufwerken sowie ein Teil der Klangmodule die Interrupts aus. Die Verwendung von NMIs könnte hier Chaos hervorrufen.

„Ereignisse" werden am einfachsten als „Software-Interrupts" beschrieben. Dabei führt die Firmware bestimmte Aufgaben unabhängig vom gerade ablaufenden Programm aus. Die Firmware „weiß" auch, zu welchem Zeitpunkt Ereignisse aktiviert werden sollen.

Der Ereignisblock



Systemzeiger (nicht ändern)

Ereigniszähler: Unter Null = Abgeschaltet
Null = Keine Ereignisse übrig
Über Null = Zahl der Ereignisse, die noch ausgeführt werden

Ereignisklasse: Asynchron, Synchron, Express oder Normal

Die Adresse der Ereignisroutine

Byte zur Anwahl des ROMs, in dem sich die Routine befindet

Felder, die vom Anwender definiert werden können

```
; Example of using the Kernel timer routines
;
;
SET.TIME:       EQU   £BD10      ; read the fast tick time count
GET.TIME:       EQU   £BD0D      ; set the fast tick time count
;
READ.CHAR:      EQU   £BB09      ; scan keyboard for char
PAUSE:          EQU   300        ; Pause for 300 ticks
;
                LD    HL,0       ; set time to zero
                LD    DE,0
                CALL  SET.TIME
                LD    BC, PAUSE  ; set wait period
;
; wait for a character to become available from
; the keyboard
;
WAIT:           CALL  READ.CHAR  ; character ready?
                RET   C          ; if so go back
;                                  with carry set
                CALL  GET.TIME   ; find the time
                SBC   HL,BC      ; one second?
;                                  (or greater)
                JR    C,WAIT     ; if not then retry
;
; if we get here, then routine has timed out
; go back with carry false to flag a timeout
;
                RET
;
```

Besondere Ereignisse

Zwar können Ereignisse auch externe Interrupts bearbeiten, doch eignen sie sich am besten für komplizierte Simulationsprogramme mit Vorgängen, die parallel zum „Vordergrundprogramm" ablaufen müssen. So arbeitet beispielsweise der SSA-1 Sprachsynthesizer mit Ereignissen, um Sprache von einem BASIC-Programm erzeugen zu lassen.

Ereignisse werden dem Betriebssystem als „Ereignisblock" übergeben, der mit sieben zusammenhängenden Bytes in den mittleren 32 KBytes des Arbeitsspeichers liegt. Das nebenstehende Bild zeigt den Aufbau eines Ereignisblocks. Drei Felder des Blocks, die Auskunft über den Ereignistyp und die Verarbeitungsadresse geben, gleichen sich bei allen Ereignissen.

Der „Ereigniszähler" hält fest, wie oft das Ereignis eingetreten ist. Für das Auslösen eines Ereignisses wird der Zähler auf eine positive Zahl gesetzt, die die Anzahl der Ereignisse angibt. Nach Ablauf der Ereignisroutine wird der Zähler um Eins dekrementiert. Ein auf eine negative Zahl gesetzter Zähler schaltet dann das Ereignis ab.

Die „Ereignisklasse" zeigt an, ob das Ereignis synchron oder asynchron zum Hauptprogramm laufen soll. Asynchrone Ereignisse werden unabhängig vom Hauptprogramm ausgeführt und sind für Anwendungen gedacht, die unmittelbare Reaktionen erfordern.

Synchrone Ereignisse werden bei ihrem Eintritt in eine Warteschlange gesetzt, die das Hauptprogramm je nach Bedarf in der Reihenfolge ihres Auftretens bearbeitet. Ein Ereignis läßt sich weiterhin auch auf „express" oder „normal" setzen.

In der nächsten Folge gehen wir auf die Unterschiede der Ereignisklassen genauer ein.

Hoffnung durch HOPE-Programme

Funktionelle Programmierung heißt das Zauberwort. Diese Technik verspricht einen Ausweg aus dem Software-Dilemma der heutigen Großrechner und eignet sich insbesondere für die Parallelverarbeitung.

Viele Computer-Wissenschaftler experimentieren mit „funktionellen" Programmiersprachen, die aus der Softwarekrise heraushelfen sollen. In funktionellen Sprachen erstellte Programme haben den Vorteil, daß Programmsegmente unabhängig von ihrer Software-„Umgebung" arbeiten.

Diese Eigenschaft wird durch den Verzicht auf Variablen erreicht, die in funktionellen Sprachen oft ganz fehlen und durch „Funktionen" ersetzt sind. Zwei BASIC-Programme verdeutlichen diese Technik:

```
10 X = 57
20 A = SIN(X)
30 B = LOG(A)
40 C = SQRT(B)
50 PRINT C

10 PRINT SQRT(LOG(SIN(57)))
```

Das zweite Programm führt ohne Variablen zu dem gleichen Ergebnis. Es werden ausschließlich Funktionen auf die Resultate anderer Funktionen angewandt.

Zu größerer Verbreitung gelangte bisher hauptsächlich die reine Form der funktionalen Sprache LISP. Die meisten modernen Dialekte bieten aber zusätzliche nicht-funktionale Möglichkeiten. Auch sind Schleifen möglich, die die Ausführungszeiten auf den Computern verkürzen.

HOPE ist eine rein funktionale Sprache, die an der Universität von Edinburgh entwickelt wurde. HOPE-Programme arbeiten ebenso wie LISP mit der Definition von Funktionen. Jede Funktion besteht aus einer Reihe von Gleichungen, die angeben, welcher Wert für ein beliebiges Argument ausgegeben werden soll. Variablen sind zwar möglich, ihr Wert läßt sich aber nicht durch Befehle beliebig ändern.

Eine Funktion zur Berechnung des Quadrates einer Zahl würde dabei so aussehen:

```
dec sqare: num -> num;
  --- square (x)<=x*x;
```

Dabei steht dec für „declare" und signalisiert den Anfang der Definition; num -> num heißt, daß die Funktion square eine Zahl als Argument hat, und eine Zahl als Wert ausgibt (ähnlich wie in PASCAL gehören Werte in HOPE einer bestimmten Klasse an). Die mit --- beginnende Gleichung bedeutet, daß das Quadrat einer Zahl gleich dieser mit sich selbst multiplizierten Zahl ist. Das Symbol <= bedeutet „wird definiert als" bzw. „kann ersetzt werden durch". x auf der linken Seite bezieht sich auf jede als Argument gegebene Zahl.

Die funktionale Computersprache „HOPE" wurde in der Informatik-Abteilung der Universität Edinburgh entwickelt. Neben R. M Burstall und D. B. MacQueen war auch der amerikanische Wissenschaftler D. T. Sannella von den Bell Laboratories daran beteiligt. Ihren Namen erhielt die Sprache nach der Straße, an der die Forschungseinrichtung liegt: dem Hope Square in Edinburgh.

Wenn wir bei der Anwendung der Funktion square(4) eingeben, antwortet HOPE mit 16:num, also Ergebnis und Werteklasse. Der Ausdruck square darf auch innerhalb anderer Funktionen aufgerufen werden.

Etwas komplizierter ist es, mit HOPE die Fakultät einer Zahl zu berechnen:

```
dec fact: num ->num;
  --- fact (0) <=1;
  --- fact (succ(n))<=(succ(n)*fact(n));
```

Die beiden Gleichungen legen den Funktionswert für alle möglichen Fälle fest (num beinhaltet nur positive Zahlwerte; negative sind ausgeschlossen). Wenn das Argument 0 ist, wird 1 als Ergebnis ausgegeben. In jedem anderen Fall ist die Fakultät von „eins-mehr-als-n" n-mal Fakultät „eins-mehr-als-n". Das ist eine rekursive Definition, weil fact über sich selbst definiert wird; funktionelle Sprachen nutzen eben solche Rekursion anstelle von

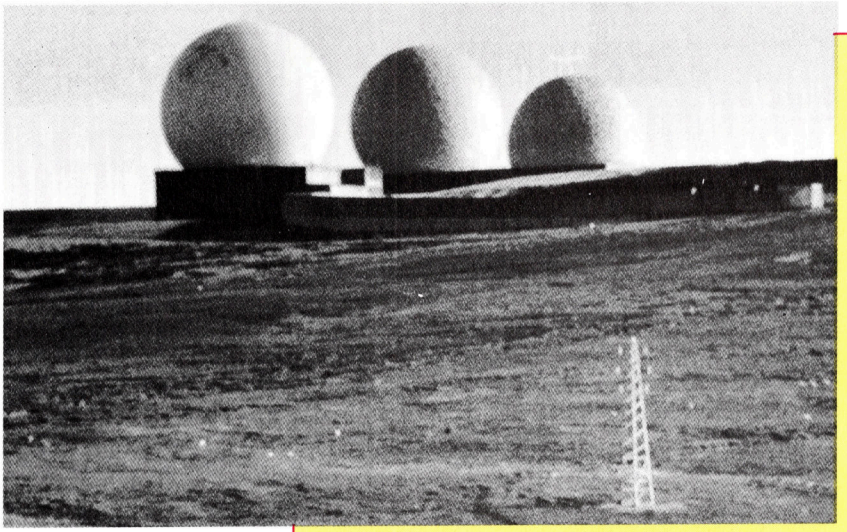
Programmschleifen.

Die Reihenfolge der Gleichungen ist beliebig, es würde auch so gehen:

```
dec fact:num->num;
--- fact(succ(n))<=(succ(n)*fact(n));
--- fact (0) < =1;
```

Die Funktion succ (successor=Nachfolger), die eine um 1 größere Zahl als das Argument ergibt, gehört zu den in HOPE „eingebauten" constructor-Funktionen. Jede HOPE-Datenklasse wird von einer speziellen constructor-Funktion gebildet. Wenn etwa die Konstante 3 eingegeben wird, berechnen wir eine Funktion mit dem Namen und Wert 3, tatsächlich aber die Kurzform des Ausdrucks succ(succ(succ(0))).

Um verschiedene Objekte einer Klasse zu speichern, verwendet HOPE anstelle von Arrays sog. Listen. Listen stehen in Klammern: [1,2,3,4] ist also eine Liste mit vier Zahlen, die über bestimmte Ausdrücke weiterverarbeitet werden können. Auf das Beispiel angewandt,



Der Entwurf umfangreicher computergesteuerter Systeme, wie beispielsweise für diese Frühwarnstation in North Yorkshire, ist ein sehr schwieriges Unterfangen. Bei herkömmlichen Verfahren muß die Arbeit vieler Programmierer zu einem riesigen Gesamtprogramm zusammengefaßt werden. Änderungen und Erweiterungen können unendliche Probleme aufwerfen. Es wird erwartet, daß die Fortschritte in der funktionalen Programmierung hier zu Erleichterungen beitragen.

Die Software-Krise

Software-Krise ist eine etwas dramatische Beschreibung der heutigen Situation: Den immer leistungsfähigeren und preisgünstigeren Rechnern steht die vergleichsweise teure Software gegenüber. Ungleich stark steigen die Kosten für die Programmpflege. Und die Zuverlässigkeit von Programmen bleibt nach wie vor ein Problem.

Ursache für diese Schwierigkeit ist die Struktur herkömmlicher Programmiersprachen. Zwar führten neue Sprachen wie PASCAL zu einigen Verbesserungen; es bleibt aber weiterhin schwierig, aus dem Listing eines Programms abzulesen, was es eigentlich tut. Das gilt speziell für Assembler und nichtstrukturierte Sprachen wie BASIC oder FORTRAN. Selbst bei selbstgeschriebenen Programmen ist es oft schwer, ihren Ablauf später nachzuvollziehen. So etwas erschwert die Programmpflege. Große Software-Systeme, wie sie in der Raumfahrt oder beim Militär eingesetzt werden, sind kaum noch für den einzelnen überschaubar.

Eine der Schwierigkeiten beim Verstehen von Programmen liegt darin, daß sie von der Entwicklungsgeschichte abhängen, also „historisch" betrachtet werden müssen. Anders als in einer mathematischen Formel stellt ein Computerprogramm nicht alle Informationen dar, mit denen es zu einem bestimmten Zeitpunkt arbeitet; so hängen etwa die Variablen eines konventionellen Programms vom bisherigen Programmablauf ab. Nehmen Sie dieses

BASIC-Programm für den Acorn als Beispiel:

```
 10 FLAG%=0
 15 REM
 20 DEF FNx (A%)
 30 FLAG%=1
 40 =2*A%
 45 REM
 50 DEF FNy(A%)
 60 LOCAL B%
 70 IF FLAG%THENB%=3 ELSE B%=4
 80 =B%*A%
 85 REM
 90 PRINT FNy(2)+FNx(1)
100 PRINT FNy(2)+FNx(1)
```

Die beiden Print-Befehle zeigen den Wert 8 bzw. 10. Der Wert von PRINT FNy(2)+FNx(1) ist jeweils verschieden. Eine Änderung der Reihenfolge würde ebenfalls das Ergebnis beeinflussen: FNy(2)+FNx(1) ist nicht das gleiche wie FNx(1)+FNy(2). Um ein Programm ganz zu verstehen, muß man es meist im Kopf oder schriftlich ausführen.

Im Gegensatz hierzu sind mathematische Ausdrücke einfach.

```
(3+4)*(6-2)=(4+3)*(6-2)=7*4=28
```

hat immer den gleichen Wert, weil das mathematische Gesetz besagt, daß 3 + 4 das gleiche ist wie 4 + 3. Der Wert eines Ausdruckes ergibt sich immer aus dem Wert seiner Komponenten. In konventionellen Programmen ist das leider anders.

Wenn es gelingt, Programmen diese mathematischen Eigenschaften zu verleihen, ergeben sich viele Vorteile – die „Software-Krise" hätte ein Ende:

- Programme wären lesbar. Das Programmlisting würde eindeutig und ohne Abhängigkeit von der Vorgeschichte die Funktion darstellen.
- Wie bei Theorien könnte bewiesen werden, daß Programme für alle denkbaren Eingaben richtig arbeiten und nicht nur bei einigen wenigen Prüfwerten. Solche Tests könnten vom Computer durchgeführt werden.
- Richtige, aber uneffektive Programme könnten mit ähnlichen Verfahren wie den in der Mathematik gebräuchlichen umgewandelt werden. Auch dieser Vorgang ließe sich automatisieren.

würde etwa das x im Ausdruck x::y das erste Element der Liste (1) bedeuten, y den Rest [2,3,4]. Das Symbol :: heißt „cons" und ist die constructor-Funktion für Listen.

Textstrings werden als Listen von Buchstaben dargestellt und können zwei Formen haben: „fred" wäre das gleiche wie ‚f',,r',,e',,d'. Eine Funktion, die die Buchstaben eines Wortes zählt, könnte so aussehen:

```
dec  lettercount: list char −>num;
   −−− lettercount (nil) <=0;
   −−− lettercount (x::y)<=lettercount
(y)+1;
```

nil bedeutet hier eine leere Liste. Die Funktion arbeitet so:

```
letter count („aardvaark");
9:num
```

Die Datenklassen sind bei HOPE flexibler als bei PASCAL, man kann auch Funktionen erzeugen, die unterschiedliche Klassen verarbeiten:

```
typevar:alpha

dec listcount:list (alpha) −>num;
   −−− listcount (nil)   <=0;
   −−− listcount (x::y) <=listcount (y) +1;
```

Damit ließe sich die Anzahl der Elemente jeder Liste zählen – man könnte demnach einfach auf lettercount verzichten.

Programmierer können ihre eigenen Datenklassen definieren, die entweder als Argumente oder als Funktionswerte verwendet werden. Selbst Funktionen können als Argument verwendet oder als Resultat ausgegeben werden. Damit ist der Weg zu sehr leistungsfähigen Programmen offen.

Parallelverarbeitung

Ein großer Vorzug funktionaler Programmierung liegt darin, das diese Technik sich sehr gut für die Parallelverarbeitung eignet. Jeder Programmteil funktioniert unabhängig von den anderen. In herkömmlichen Sprachen war die Nutzung gemeinsamer Variablen durch mehrere Programmteile das Nadelöhr für die Parallelverarbeitung. Stellen Sie sich diese beiden BASIC-Programme bei simultaner Abarbeitung vor:

```
10 A=0                    10 B=0
20 FOR X=1 TO K           .
.                         .
.                         .
100 NEXT X                .
2000 A=B + 56             2000 PRINT A
.                         .
.                         .
```

Der für A ausgegebene Wert hängt davon ab, ob das erste Programm bereits Zeile 2000 erreicht hat. Falls nicht, ist A noch 0. Die Verarbeitungszeit für das erste Programm hängt aber von K ab ... damit wird das Ergebnis unkalkulierbar.

In einer Programmiersprache vom HOPE-Typ, bei der es keine Variablen gibt, kann so etwas nicht vorkommen. Als Beispiel soll ein Programm dienen, das die Summe der Fakultäten einer Zahlenliste erzeugt. Die Fakultätsberechnung kennen wir schon:

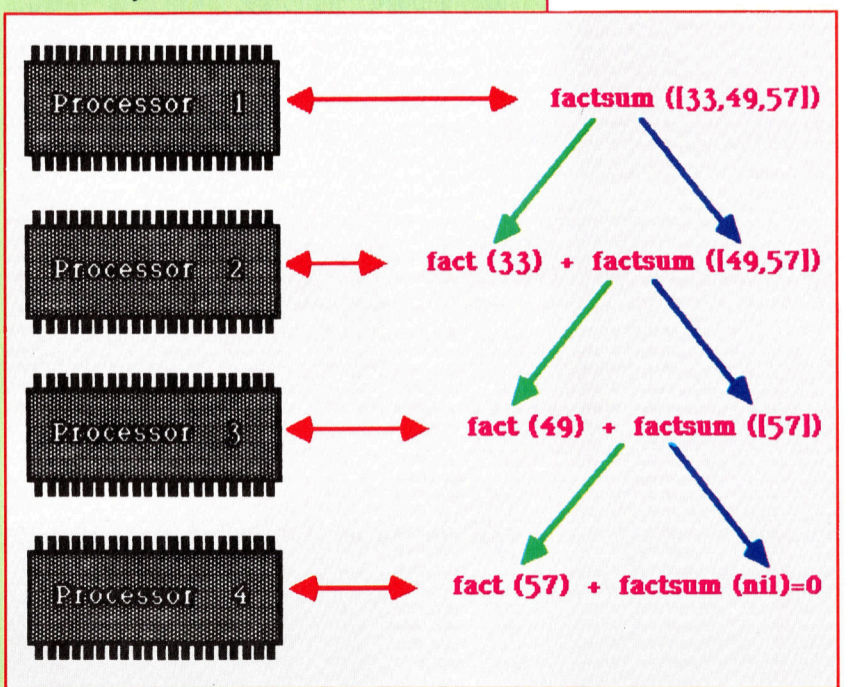
```
dec fact:num −>num;
−−− fact(0) <=1;
−−− fact(succ(n))<=(succ(n)*fact(n));
```

Jetzt können wir unter Verwendung von fact summieren:

```
dec factsum:list(num) −>num;
−−− factsum(nil) <=0;
−−− factsum(x::y)<=fact(x) + factsum(y);
```

Das bedeutet, daß die Fakultätssumme einer Leerliste 0 ist, andernfalls wird die Fakultät des ersten Elementes zur Fakultätssumme des Restes hinzugezählt. Die zwei Ausdrücke auf der rechten Seite der zweiten Gleichung sind völlig unabhängig voneinander und lassen sich daher parallel berechnen.

Es kann jederzeit ein zusätzlicher Prozessor zur Berechnung herangezogen werden, wobei der erste Prozessor die Teilresultate beim Erreichen der 0 addiert. So arbeitet auch der am Imperial College entwickelte Parallelrechner ALICE, der über eine Ringkonfiguration von Inmos-Transputern verfügt. Jeder freie Prozessor im Ring kann eine der Teilaufgaben übernehmen, die über den Ring zur Bearbeitung angeboten werden.



Auf vollen Touren

Ein Rockmusical für den Commodore 64 ist der „Ghetto Blaster". Bei diesem aktionsreichen Spiel sorgen heiße Rhythmen statt Blaue Bohnen für die nötige Spannung.

Der „Ghetto Blaster" hat wie alle guten Spiele ein relativ simples Konzept. Funk-Fan „Rocking Rodney" ist gerade als Bote bei der Plattenfirma „Interdisc" angestellt worden und soll bis zum Abend zehn Vorführbänder zur Zentrale in die Funky Street bringen. Unterm Arm trägt er seinen „Ghetto Blaster" – und läßt unbekümmert die Cassetten laufen. Wenn Sie Rodney mit seinem Turbosound auf arglose Fußgänger zielen lassen und den Joystick drücken, fährt das den Leuten ganz schön in die Knochen – aber Rodneys Welt besteht nicht nur aus solchen Freuden.

Der Auftrag muß nämlich erledigt sein, bevor das Zählwerk des Ghetto Blaster auf 999 steht, und außerdem muß Rodney aufpassen, daß die Batterien nicht leer werden. Gleichzeitig versucht die berüchtigte Bande „Gangsters of the Groove" ihm die Cassetten wegzuschnappen, der Psycho-Killer liegt auf der Lauer, und mit dem langen Arm des Gesetzes ist auch nicht zu spaßen. Die Musikmaschine kann bei einem jähen Zusammenprall mit dem „disco-tauben" Passanten demoliert werden, und das kostet wieder wertvolle Zeit in der Werkstatt. Erschwerend kommt hinzu, daß die zehn Cassetten überhaupt erst aufgespürt werden müssen, denn sie sind in ganz Funky Town versteckt, und zwar in bestimmten Hauseingängen. Jeder einzelne dieser Einfälle ist vielleicht gar nicht übermäßig aufregend, aber in der Summe machen sie das Spiel doch höchst verwickelt und attraktiv.

Der Bildschirm ist zweigeteilt: Die obere Hälfte zeigt die Straße, die Rodney gerade mit seinem Beat beschallt – laut Beiblatt sogar in 3 D-Wiedergabe, was leicht übertrieben erscheint. Im übrigen sind die Szenen aber recht hintersinnig. Die Straßenzüge „Strawberry Fields" und „Blueberry Hill" liegen durchaus in einer feinen Gegend, während „Desolation Row" und „Stoney End" deprimierend verwahrlost sind. Die untere Schirmhälfte nimmt Rodneys Stereokoffer ein, bei der Sie vor allem das Zählwerk mit der Zeitanzeige und die Batteriekontrolle im Auge behalten müssen. Den Lautstärkeregler dürfen Sie nicht zu weit aufreißen, damit die Batterien länger halten.

Wer mit dem Anspruch wirbt, das „erste Computer-Rockmusical" mit einem „beim Commodore 64 unerreichten Funky Sound" auf den Markt zu bringen, muß schon ein recht gutes Spiel geschrieben haben – sonst folgt der Frust den hochgesteckten Erwartungen auf dem Fuße.

Ghetto Blaster

Der Ghetto Blaster holt das letzte an Funky Sound aus dem Commodore 64 heraus. Rodneys erste Cassette mit dem beziehungsreichen Titel „Last Year's T-Shirt Mix" setzt den Maßstab für die folgenden Nummern – kein Wunder, daß die Passanten bei dem fetzigen Disco-Beat so schnell in Fahrt kommen. Wenn Sie davon trotzdem genug haben, sollten Sie zwischendurch ruhig mal irgendwo einkaufen gehen (nicht ohne sich dem Vorwurf auszusetzen, ein Materialist zu sein). Sie können auch in eins der Häuser hineinschlüpfen (sofern Sie nicht hören „I hear you knocking, but you can't come in"), oder vielleicht juckt Sie ein Trip durch den „Itchycoo-Park". Wenn Sie sich verlaufen haben, lassen Sie am besten ein Stoßgebet los, daß Ihnen ein Treffen mit „Jumping Jack Flash" beschert. Dessen Richtungssinn ist nämlich Spitze. Und wenn der Ofen aus ist, drücken Sie die Leertaste und machen Pause.

Der Ghetto Blaster ist im Prinzip zwar auch ein Baller-Spiel, aber statt Kanonen dröhnen hier erfreulicherweise nur heiße Rhythmen. Das Spiel ergreift wie eine heiße Popnummer unbewußt von Ihnen Besitz und läßt Sie süchtig werden. Es ist gekonnt gemacht, sogar mit Humor, der den meisten Spielen ja total abgeht.

Am Ghetto Blaster haben deshalb alle ihren Spaß, vor allem natürlich Teenager, selbst wenn sie den Anspielungen auf die Musik der ausgehenden sechziger Jahre nicht immer folgen können. Und eine Straße namens „Hold Me Close" beeindruckt wahrscheinlich nur die David-Essex-Fans. Aber auch ohne daß jeder die feinen Spitzen voll würdigen kann, hat der Ghetto Blaster alle Chancen, ein Millionenhit zu werden.

Fachwörter von A bis Z

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.

Read/Write Head = Schreib/Lese-Kopf
Dieses Bauteil im Cassetten- und Diskettenlaufwerk besorgt beim Schreiben die oberflächliche Magnetisierung des Datenträgers und fragt sie beim Lesen wieder ab. Bei der Aufzeichnung werden Ströme durch eine kleine Spule mit Eisenkern im Magnetkopf geschickt, um entlang von „Spuren" in der ferromagnetischen Beschichtung die Bits „einzumagnetisieren". Das Lesen setzt voraus, daß sich das Speichermedium relativ zum Kopf bewegt: Dann wirkt auf die Kopfspule ein veränderliches Magnetfeld, das in der Wicklung dem Bitmuster entsprechend elektrische Spannungsimpulse induziert, die verstärkt und an die CPU übertragen werden.

Real Time = Echtzeit
Bei Echtzeit-Betrieb muß der Rechner auf jede Eingabe ohne Verzögerung reagieren. Bei einer Prozeßsteuerung ist das ebenso wichtig wie bei einem Flugsimulator, wo die Situation so realistisch wie möglich wirken soll.

Es kostet erhebliche Mühe, die Reaktionszeit bei solchen Systemen auf ein Minimum zu reduzieren. Welche Verzögerung noch zulässig ist, hängt weitgehend vom konkreten Anwendungsfall ab. Weder bei der Bestätigung einer Buchung am Flugbüro noch bei einem Kassenautomaten kommt es auf Millisekunden an, anders als etwa bei der radargestützten Flugüberwachung. Ein Echtzeit-System im eigentlichen Sinn verlangt extrem schnelle und effiziente Soft- und Hardware.

Record = Datensatz
Wie vieles andere hat auch „Record" in der Datenverarbeitung mehrere (allerdings verwandte) Bedeutungen. Meist ist ein „Record" eine Informationseinheit innerhalb eines „Files", die selbst wieder aus inhaltlich zusammengehörigen Datenfeldern aufgebaut ist. Dafür ist auch die Bezeichnung „logischer Datensatz" gebräuchlich.

Zu unterscheiden davon sind Records oder „Blöcke", aus denen die Information auf dem Datenträger hardwaremäßig zusammengesetzt ist. Diese „physischen" Datensätze stellen die kleinste Datenmenge dar, die der Magnetspeicher akzeptiert oder selbst an die CPU liefert. Bei CP/M etwa entspricht die Blocklänge mit 128 Byte gerade einem Diskettensektor. Ein physischer Datensatz bildet demnach logischerweise eine durch die Formatierung des Datenträgers definierte Einheit.

Recoverable Error = Regenerierbarer Fehler
Wenn der Rechner Daten vom Band einliest, stellt die Fehlererkennungsroutine (meist anhand einer Paritäts- oder Prüfsummenabfrage) hin und wieder Unstimmigkeiten fest. Oft handelt es sich dabei nur um ein einziges falsches Bit in einer langen Datei. Statt nun alles erneut einlesen zu müssen, gibt es eine Methode, bei der nur die fehlerhaften Daten neu geladen werden müssen. Gelingt das ohne abermaligen Protest, handelte es sich um einen „regenerierbaren" Fehler.

Großrechner und Minicomputer, bei denen die CPU direkt den Bandlauf überwacht, gehen meist von sich aus an die Behebung derartiger Fehler: Das Band wird nach dem Erkennen sofort gestoppt.

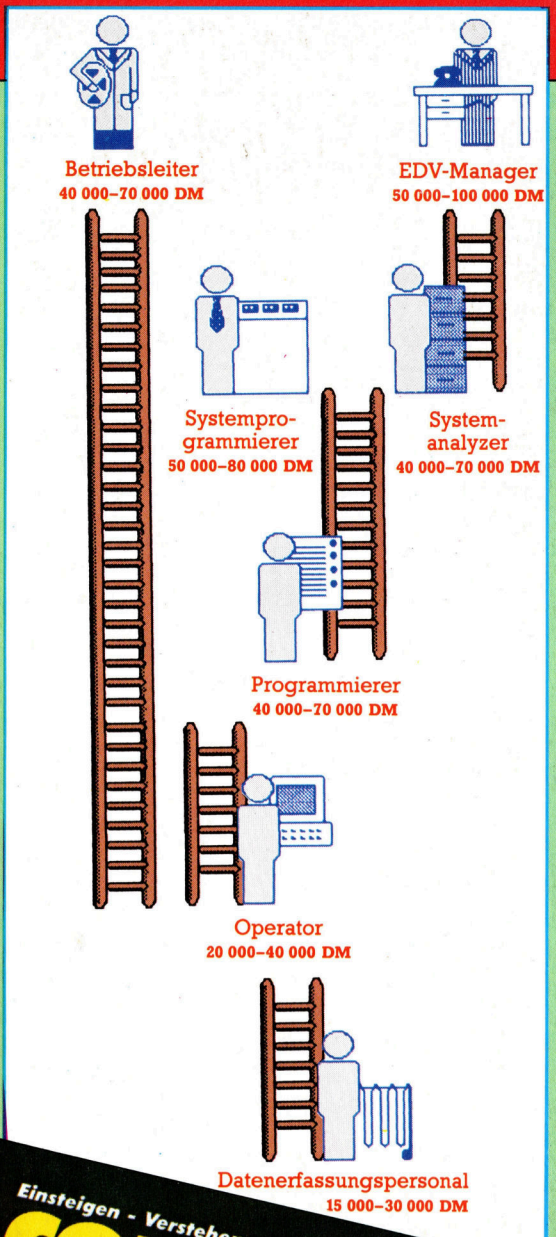
Zur Gewährleistung der nötigen Wirklichkeitstreue müssen bei einem Flugsimulator für das Pilotentraining alle Steuerungsreaktionen und Instrumenten-Rückmeldungen genauso spontan wie im realen Cockpit erfolgen. Um die erforderliche Ansprechgeschwindigkeit des Rechnersystems bei solchen Echtzeit-Problemen zu erzielen, kommt nur die Programmierung in einer Compilersprache in Frage.

Bildnachweise
2157,2158,2159: Mike Clowes
2176,2177,2178: Crispin Thomas
2161, 2180: Caroline Clayton
2182: Don McPhee

+ Vorschau +++ Vorschau +++ Vorschau +

computer kurs

Heft 79



Schreiberlinge

Mit Spielprogrammen Karriere zu machen, ist der Traum vieler Computer-Freunde. Sie muß nur richtig eingeschätzt werden.

Wortwechsel mit Unix

Unix bietet eine Vielzahl von hilfreichen Dienstprogrammen.

Unter der Haube

In diesem Artikel wird die Architektur von drei gängigen Heimcomputern betrachtet.

Ereignisreich

Wir zeigen Ihnen in diesem Artikel, wie sich „Ereignisse“ in der Maschinencodeprogrammierung einsetzen lassen.