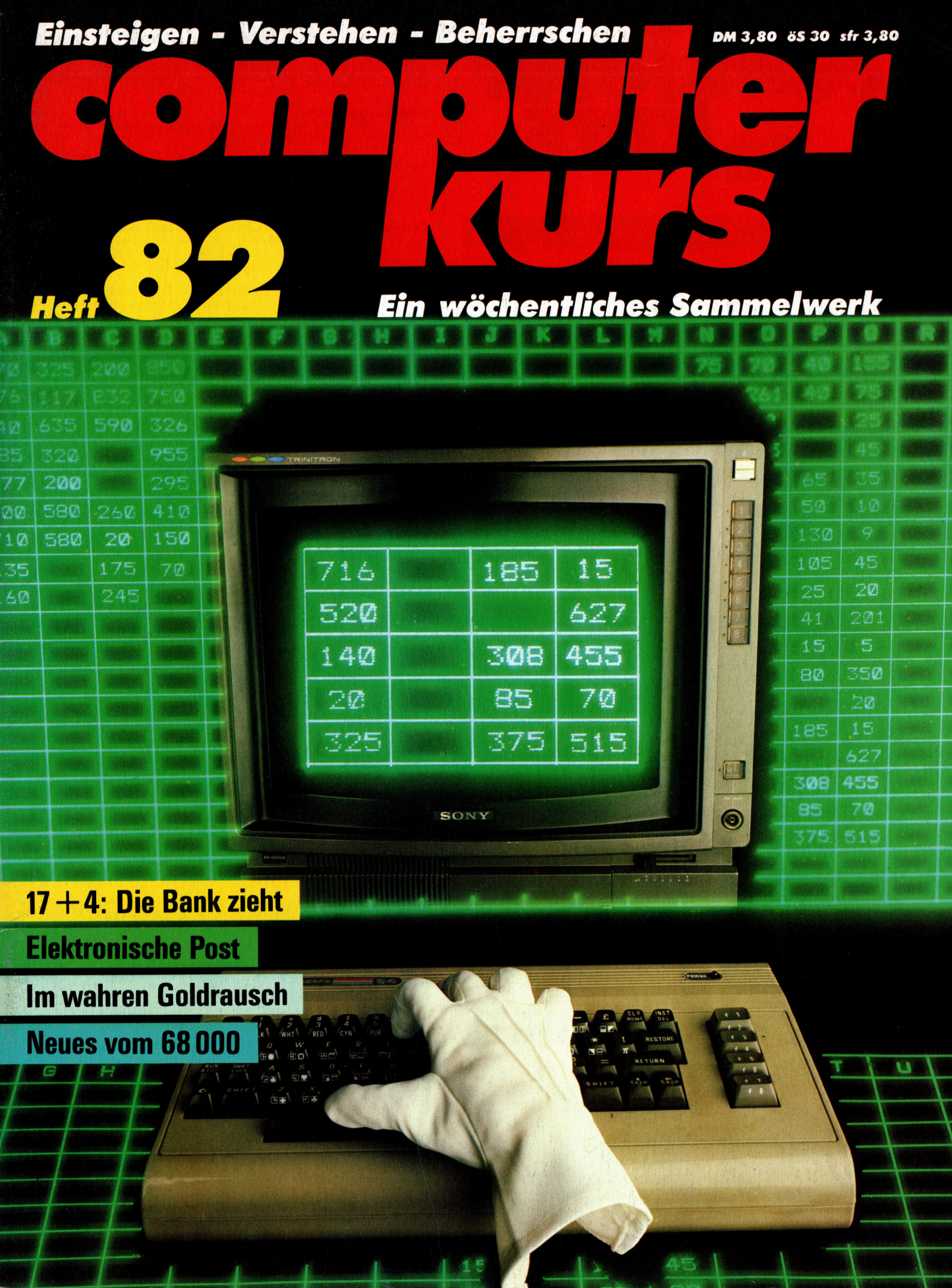
Einsteigen - Verstehen - Beherrschen

DM 3,80 öS 30 sfr 3,80

computer kurs

Heft 82

Ein wöchentliches Sammelwerk



17 + 4: Die Bank zieht

Elektronische Post

Im wahren Goldrausch

Neues vom 68 000

computer kurs Heft 82

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.

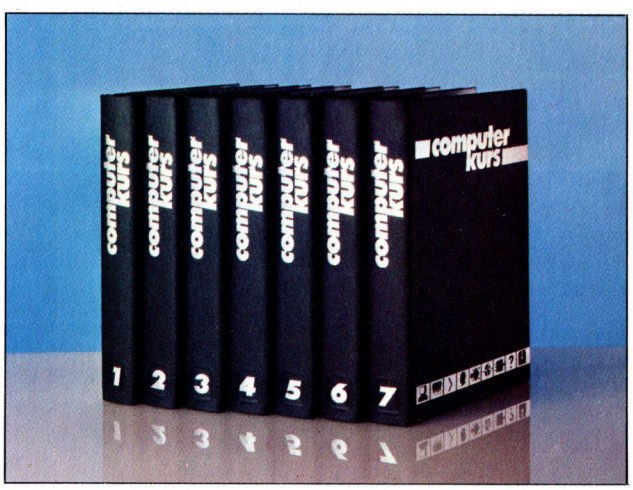
Schweiz: Der Sammelordner kostet sfr 15. Bitte wenden Sie sich an Ihren Kiosk; dort werden Sie jederzeit die gewünschten Exemplare erhalten.

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: Peter Aldick (verantw. f. d. Inhalt), Gudrun Anderson, Joachim Knipp, 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

Die Bank zieht

In unserem 17+4-Spiel erreichen wir nun den Punkt, an dem der Spieler die komplette Hand ausspielen kann. Jetzt betrachten wir die Routinen, die es der Bank ermöglichen, entsprechend zu reagieren.

Hat der Spieler sein Blatt vervollständigt, ist die Bank am Zug. Die Regeln, die wir in dieser Version des 17+4-Spiels angenommen haben, geben der Bank einige Vorteile. Als erstes kennt die Bank die Punktezahl des Gegners und muß somit keine Züge riskieren, die ihr Blatt sprengen würden. Der zweite Vorteil ist, daß die Bank nur den gleichen Punktestand erreichen muß, um die Runde zu gewinnen. Die Bank darf nicht ein Blatt mit 12, 13 oder 14 Punkten ausspielen.

In diesem Stadium des Spieles zeigt der Bildschirm Ihre Karten und die zwei Karten, die anfangs an die Bank ausgegeben wurden, an. Die erste dieser Karten wurde mit dem Bild nach unten ausgeteilt, so daß die nächste Aufgabe des Programms im Umdrehen der Karte besteht. Die einfachste Lösung wäre, die Kar-

Der Computer ist dran

Schneider CPC

```
130 REM **** computer's turn ****
140 pl=2
150 ep=20:GOSUB 670:REM erase cards
160 cn=hd(pl,1,1):su=hd(pl,1,2):GOSUB 10
00:REM reprint
170 cn=hd(pl,2,1):su=hd(pl,2,2):GOSUB 10
00:REM reprint
180 GOSUB 2500:REM bank twist etc
190 REM **** win or lose ****
200 IF bw=1 THEN GOSUB 700:PEN black:PRI
NT"sorry you lose":GOTO 300
210 GOSUB 700:PEN black:PRINT"you win "
300 a$="":WHILE a$="":a$=INKEY$:WEND

305 REM ** if shift/s then shuffle **
310 IF a$="S" THEN GOSUB 700:PRINT"shuff
ling....please wait":GOSUB 3000
320 GOTO 50
2500 REM **** bank's turn ****
2520 ON pv+1 GOSUB 2540,2550,2560,2570,2
580
2530 RETURN
2540 bw=1:RETURN:REM punter bust
2550 ts=ps:GOSUB 5000:RETURN:REM twist u
ntil beat punter or bust
2560 ts=21:GOSUB 5000:RETURN:REM twist u
ntil pontoon or bust
2570 GOSUB 5200:RETURN:REM twist until 5
 card trick or bust
2580 GOSUB 800:IF ef=2 THEN bw=1:RETURN:
REM bank's royal pontoon
2585 bw=0:RETURN:REM punter's royal pont
oon
5000 REM **** bank twist until target or
 bust ****
5010 GOSUB 800:REM evaluate
5020 IF ef=4 THEN bw=0:RETURN:REM bust
5025 IF ef=2 OR ef=5 THEN bw=1:RETURN:RE
M bank royal pontoon/five card trick
5040 IF tt(2,1)>=ts OR(tt(2,2)<=21 AND t
t(2,2)>=ts) THEN bw=1:RETURN
5045 IF hp(2)>5 THEN bw=0:RETURN:REM fiv
 cards dealt
5050 GOSUB 1300:GOTO 5000:REM deal and r
e-evaluate
5200 REM **** twist until 5 card trick o
r bust ****
5220 GOSUB 800:REM evaluate
5225 IF ef=4 THEN bw=0:RETURN:REM bust
5227 IF ef=2 OR ef=5 THEN bw=1:RETURN:RE
M bank royal pontoon/five card trick
5228 IF hp(2)>5 THEN bw=0:RETURN:REM fiv
 cards dealt
5230 GOSUB 1300:GOTO 5200:REM deal card
and eval again
```

Sinclair Spectrum

```
 130>REM **** COMPUTER'S TURN ****
 140 LET PL=2
 150 LET EP=14: GO SUB 670: REM ERASE CA
RDS
 160 LET CN=D(PL,1,1): LET SU=D(PL,1,2):
 GO SUB 1000: REM REPRINT
 170 LET CN=D(PL,2,1): LET SU=D(PL,2,2):
 GO SUB 1000: REM REPRINT
 180 GO SUB 2500: REM BANK TWIST ETC
 190 REM **** WIN OR LOSE ****
 200 IF BW=1 THEN  GO SUB 700: PRINT "SO
RRY YOU LOSE": GO TO 300
 210 GO SUB 700: PRINT "YOU WIN #";BT
 300 LET A$=INKEY$: IF A$="" THEN  GO TO
 300
 305 REM ** IF SYM/S THEN SHUFFLE **
 310 IF A$=CHR$ 195 THEN  GO SUB 700: PR
INT "SHUFFLING ... PLEASE WAIT": GO SUB
3000
 320 GO TO 50
2500>REM **** BANK'S TURN ****
2520 GO SUB (PV*10)+2540
2530 RETURN
2540 LET BW=1: RETURN : REM PUNTER BUST
2550 LET TS=PS: GO SUB 5000: RETURN : RE
M TWIST UNTIL BEAT PUNTER OR BUST
2560 LET TS=21: GO SUB 5000: RETURN : RE
M TWIST UNTIL PONTOON OR BUST
2570 GO SUB 5200: RETURN : REM TWIST UNT
ILFIVE CARD TRICK OR BUST
2580 GO SUB 800: IF EF=2 THEN  LET BW=1:
 RETURN : REM ROYAL PONTOON
2585 LET BW=0: RETURN : REM PUNTER'S ROY
AL PONTOON
5000>REM **** BANK TWIST UNTIL TARGET OR
 BUST
5010 GO SUB 800: REM EVALUATE
5020 IF EF=4 THEN  LET BW=0: RETURN : RE
M BUST
5025  IF EF=2  OR EF=5 THEN  LET BW=1: R
ETURN
5040 IF T(2,1)>=TS OR (T(2,2)<=21 AND T(
2,2)>=TS) THEN  LET BW=1: RETURN
5050 GO SUB 1300: GO TO 5000: REM DEAL A
ND RE-EVALUATE
5200 REM **** TWIST UNTIL 5 CARD OR BUST
 ****
5220 GO SUB 800: REM EVALUATE
5225 IF EF=4 THEN  LET BW=0: RETURN : RE
M BUST
5227  IF EF=2  OR EF=5 THEN  LET BW=1: R
ETURN
5228  IF P(2)>5  THEN  LET BW=0: RETURN
5230 GO SUB 1300: GO TO 5200
```

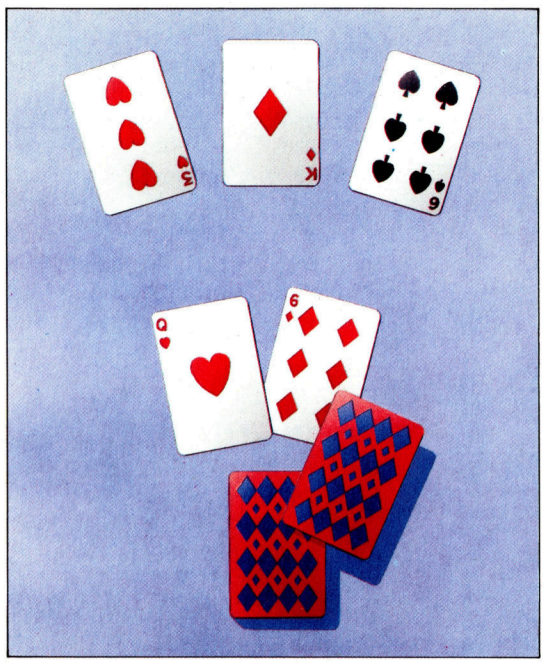
ten der Bank zu löschen, dazu können wir die Löschroutine in Zeile 670 benutzen. Anschließend zeichnen wir die Karten erneut mit dem Bild nach oben. Dafür setzen wir CN und SU (Kartenwert und Farbe) auf den Wert der Bankkarten in der Feldvariablen HD(. . .), und zum Schluß rufen wir die bereits vorgestellte Karten-Darstellungsroutine auf. Dies wird alles in der Hauptprogrammschleife in den Zeilen 150 bis 170 erledigt. Hat der Spieler verloren, so ist das Spiel beendet.

Erreicht der Spieler einen Punktestand von weniger als 21, muß die Bank so oft ziehen, bis sie die gleiche Punktzahl erreicht, übertrifft oder das Blatt sprengt.

Hat der Spieler einen „Five-Card-Trick", muß die Bank ziehen und versuchen, ebenfalls fünf Karten zu erreichen.

Hat der Spieler einen Black-Jack, so muß die Bank ebenfalls einen Black-Jack erreichen.

Die Unterroutine in Zeile 2500 wählt die benötigten Schritte durch ON . . . GOSUB-Befehle, in Verbindung mit der Variablen PV. Wenn die Bank twisten muß, um die Karte des Spielers zu übertreffen, setzt das Programm die Variable TS auf die zu erreichende Punktezahl und verzweigt in eine weitere Unterroutine bei Zeile 5000.



In dieser 17+4-Version hat die Bank den Vorteil, genau zu wissen, welchen Punktestand sie erreichen muß. In unserem Beispiel hat der Spieler 19 Punkte. Die Bank muß nun noch mindestens eine weitere Karte aufnehmen, um den Spieler zu überbieten.

Commodore 64

```
130 REM **** COMPUTER'S TURN ****
140 PL=2
150 EP=20:GOSUB670:REM ERASE CARDS
160 CN=HD(PL,1,1):SU=HD(PL,1,2):GOSUB100
0:REM REPRINT
170 CN=HD(PL,2,1):SU=HD(PL,2,2):GOSUB100
0:REM REPRINT
180 GOSUB2500:REM BANK TWIST ETC
190 REM **** WIN OR LOSE ****
200 IF BW=1 THEN GOSUB700:PRINTCHR$(156)
;"SORRY YOU LOSE":GOTO 300
210 GOSUB700:PRINT CHR$(156);"YOU WIN"
300 GET A$:IF A$=""THEN 300
305 REM ** IF SHIFT/S THEN SHUFFLE **
310 IF A$=CHR$(211)THENGOSUB700:PRINT"SH
UFFLING ... PLEASE WAIT":GOSUB 3000
320 GOTO 50
2500 REM **** BANK'S TURN ****
2520 ON PV+1 GOSUB 2540,2550,2560,2570,2
580
2530 RETURN
2540 BW=1:RETURN:REM PUNTER BUST
2550 TS=PS:GOSUB5000:RETURN:REM TWIST UN
TIL BEAT PUNTER OR BUST
2560 TS=21:GOSUB5000:RETURN:REM TWIST UN
TIL PONTOON OR BUST
2570 GOSUB5200:RETURN:REM TWIST UNTIL 5
CARD TRICK OR BUST
2580 GOSUB800:IF EF=2 THEN BW=1:RETURN:R
EM ROYAL PONTOON
2585 BW=0:RETURN:REM PUNTER'S ROYAL PONT
OON
5000 REM **** BANK TWIST UNTIL TARGET OR
 BUST ****
5010 GOSUB800:REM EVALUATE
5020 IF EF=4 THEN BW=0:RETURN:REM BUST
5025 IF EF=2 OR EF=5 THEN BW=1:RETURN
5040 IF TT(2,1)>=TS OR(TT(2,2)<=21AND TT
(2,2)>=TS)THEN BW=1:RETURN
5045 IF HP(2)>5 THEN BW=0:RETURN:REM FIV
E CARDS DEALT
5050 GOSUB 1300:GOTO 5000:REM DEAL AND R
E-EVALUATE
5200 REM**** TWIST UNTIL 5 CARD OR BUST
****
5220 GOSUB800:REM EVALUATE
5225 IF EF=4 THEN BW=0:RETURN:REM BUST
5227 IF EF=2 OR EF=5 THEN BW=1:RETURN
5228 IF HP(2)>5 THEN BW=0:RETURN
5230 GOSUB 1300:GOTO 5200:REM DEAL CARD
```

Acorn B

```
  130 REM
  140 PL=2
  150 EP=20:GOSUB 670
  160 CN=HD(PL,1,1):SU=HD(PL,1,2):GOSUB
1000
  170 CN=HD(PL,2,1):SU=HD(PL,2,2):GOSUB
1000
  180 GOSUB 2500
  190 REM
  200 IF BW=1 THEN GOSUB 700:PRINT"SORRY
, YOU LOSE!":GOTO 300
  210 GOSUB 700:PRINT"YOU WIN "
  300 A$=GET$
  305 REM
  310 IF A$="S" THEN GOSUB 700:PRINT"SHU
FFLING....PLEASE WAIT":GOSUB 3000
  320 GOTO 50
 2500 REM
 2520 ON PV+1 GOSUB 2540,2550,2560,2570,
2580
 2530 RETURN
 2540 BW=1:RETURN
 2550 TS=PS:GOSUB 5000:RETURN
 2560 TS=21:GOSUB 5000:RETURN
 2570 GOSUB 5200:RETURN
 2580 GOSUB 800:IF EF=2 THEN BW=1
 2585 BW=0:RETURN
 5000 FORDL=1TO100:NEXT
 5010 GOSUB 800
 5020 IF EF=4 THEN BW=0:RETURN
 5025 IF EF=2 THEN BW=1:RETURN
 5040 IF TT(2,1)>=TS OR (TT(2,2)<=21 AND
TT(2,2)>=TS) THEN BW=1:RETURN
 5050 GOSUB 1300:GOTO 5000
 5200 REM
 5210 GOSUB 1300
 5220 GOSUB 800
 5225 IF EF=4 THEN BW=0
 5230 IF EF<>5 THEN 5210
 5240 BW=1:RETURN
```

Hochstapelei

In den vorigen Folgen dieser Serie hatten wir festgestellt, daß sich Subroutinen ausgezeichnet zur Programmstrukturierung eignen. Wir wollen uns nun die Beziehung der Subroutinen zum Stapel ansehen und dann auf die Parameterübergabe eingehen.

Sehen wir uns zunächst ein Programmbeispiel an, das die Subroutine CALC zweimal aufruft:

```
                    (. . . Codezeilen . . .)
2000 4EB8           JSR CALC      * Subroutine
                                    aufrufen
                    (. . . Codezeilen . . .)
2100 4EB8           JSR CALC      * Subroutine
                                    nochmals
                                    aufrufen
2102 4000           (. . . Codezeilen . . .)
4000 3200 CALC      MOVE D0,D7 * Subrouti-
                                 neneinsprung
                    (. . . Codezeilen . . .)
4100 4E75           RTS           * Rücksprung
```

Bei jeder Ausführung von JSR schiebt der 68000 den Inhalt des Programmzählers (die Adresse des auf JSR folgenden Befehls) auf den Stack und lädt die von JSR angegebene Adresse in den PC. Wenn der 68000 am Ende der Subroutine das RTS findet, zieht er die Rücksprungadresse (auch „Linkadresse" genannt) wieder vom Stack und lädt sie in den PC. Die Linkadresse wird im Langwortformat auf den Stack geschoben. In unserem Beispiel befinden sich – nach Ausführung des ersten CALC – die im nebenstehenden Bild gezeigten Werte auf dem Stack und im PC.

Die zweite Hälfte des Bildes zeigt, wie der Stack nach Ausführung von RTS und dem Laden der Linkadresse in dem PC aussieht. Die Linkadresse wird dabei auf dem Stack nicht gelöscht, sondern beim nächsten Stackeinsatz einfach überschrieben.

Die für den Linkvorgang nötigen Abläufe erledigt die Hardware des 68000.

Bei verschachtelten Subroutinen wird zur Speicherung der Rücksprungadressen der Stack erweitert. Wenn sich CALC jedoch selbst aufruft (Rekursion), werden die Rücksprungadressen solange „gestapelt", bis die Rekursion endet oder das Programm wegen des Stacküberlaufs abstürzt.

Sehen wir uns nun an, wie Ein- und Ausgabeparameter übergeben werden. In der letzten Folge hatten wir CALC über D1 mit Daten versorgt und das Ergebnis in D2 erhalten. Da die Registerzahl begrenzt ist, reicht diese Lösung jedoch nur für kleine Programme.

Bei längeren Programmen bietet sich, besonders bei der Arbeit mit Compilern, der Stack als Übergabemedium an. Die Parameterübergabe an CALC würde dann so aussehen:

```
2000 MOVE PARAM1,—(SP) * ersten Parameter auf den
                         Stack schieben
2004 MOVE PARAM2,—(SP) * zweiter Param.
2008 JSR CALC          * CALC aufrufen
200A . . .
```

Nun enthält der Stack beim Einsprung in CALC

```
                PARAM1
                PARAM2
                ls link
SP in CALC —>   ms link
```

Der Zugriff auf PARAM1 geschieht mit einem Distanzwert auf SP, damit die Linkadresse übersprungen wird:

```
ADDQ #6,SP      * Pointer korrigieren
MOVE (SP),D4    * Parameter holen
```

oder einfacher mit

```
MOVE 6(SP),D4
```

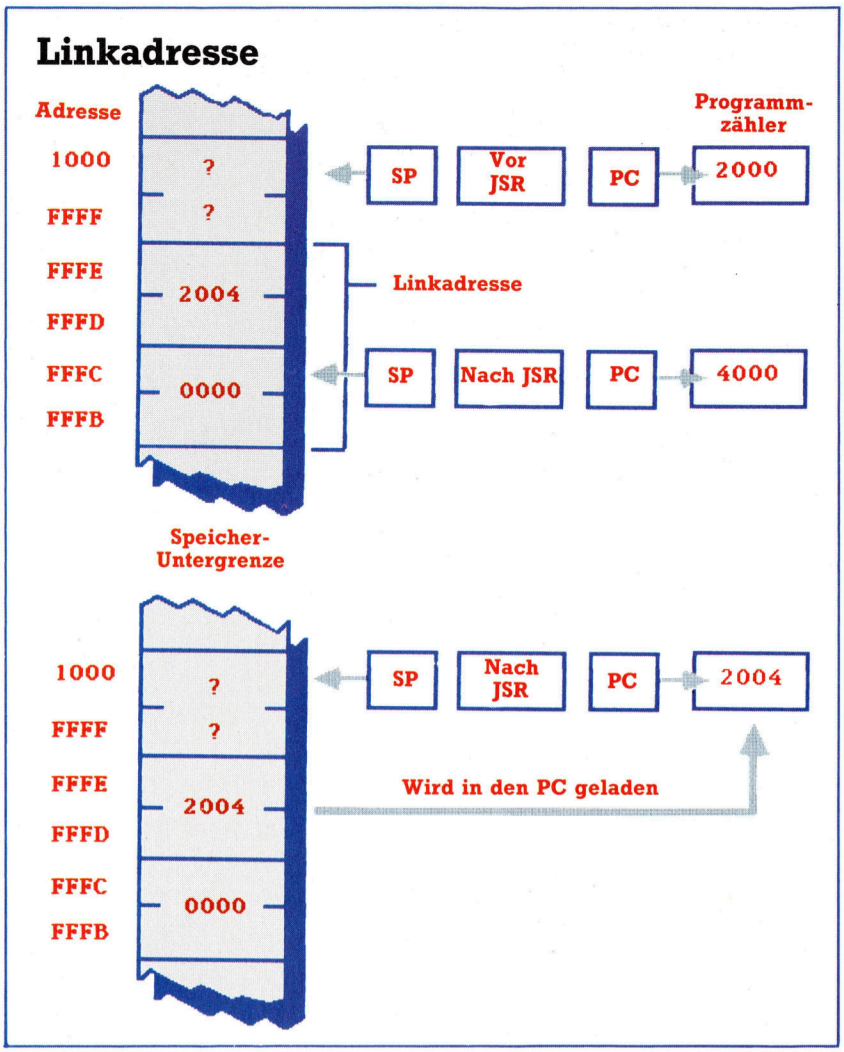
Bei einer Korrektur des Stackpointers ist sicherzustellen, daß SP vor Ausführung der Rücksprungadresse auf die Linkadresse zeigt, da Sie sonst unvorhersehbare Ergebnisse erhalten. Dies ist auch bei der Rückgabe der Ergebnisse per Stack wichtig, es sei denn, die Subroutine überschreibt einen oder beide Eingabeparameter.

Weit einfacher ist der Einsatz eines separaten Parameterstacks (beispielsweise A6), da dabei der Stackpointer nicht beeinflußt wird und für Hardwarefunktionen reserviert bleiben kann. In diesem Fall sieht der Aufruf so aus:

```
MOVE PARAM1,—(A6) * ersten Parameter
                    auf den Stack A6
                    schieben
MOVE PARAM2,—(A6) * zweiten Parameter
JSR  CALC         * SP für die Link-
                    adresse freihalten
```

In der Subroutine lassen sich die Parameter dann leicht mit

```
MOVE (A6)+,D2 * zweiten Parameter holen
MOVE (A6)+,D1 * ersten Parameter
                holen
```



Der 68000 legt eine Rücksprungadresse (die „Linkadresse") auf dem Stack ab, bevor er die Programmsteuerung an eine Subroutine übergibt. Diese Adresse wird im Langwortformat auf dem Stack abgelegt und beim Auftreten eines RTS-Befehls wieder in den Programmzähler eingesetzt.

abrufen, da keine Linkadressen übersprungen werden müssen. Beachten Sie, daß die Parameter in umgekehrter Reihenfolge eingelesen werden.

Stacks werden viel in modernen Hochsprachen mit Blockstruktur eingesetzt. Dabei ordnet die Sprache dem Parameter und den lokalen Variablen von Prozeduren Platz in einem fest definierten Speicherbereich zu. Hier ein Beispiel:

```
Procedure Berechnen (xval, yval : int);
  var
    store : int;
  begin
    store := 2*xval + yval^2;
```

Dieses PASCAL-Modul arbeitet mit den Parametern „xval" und „yval" und der lokalen Variablen „store", denen im Stack Platz zugeteilt wird. Alle weiteren, im Programm nicht aufgeführten Zwischenvariablen des Compilers (z. B. die yval^2-Komponente von „store" während der Bewertung von 2*xval) werden nach der „Push"- und „Pop"-Methode auf dem Stack abgelegt bzw. von dort abgerufen.

Der 68000 bietet für die Verwaltung der Stackdaten die beiden Befehle LINK (Adreßregister auf den Stapel schieben) und UNLK (Adreßverbindung lösen). Die Befehle werden gemeinsam eingesetzt und reservieren im Stack einen Speicherblock für den Subroutineneinsatz. Nach Einsprung in eine Subroutine richtet LINK das Adreßregister RZ („Rahmenzeiger") ein, das auf einen Datenbereich des Stacks zeigt, und bewegt den SP um einen bestimmten Distanzwert nach unten. Wenn ein Stack nach Ausführung von JSR so aussieht:

```
                PARAM1
                PARAM2
                ls link
        SP →    ms link
```

dann hat er nach LINK folgenden Inhalt:

```
                PARAM1
                PARAM2
                ls link
                ms link
        RZ →    alter RZ
Distanzwert . . . lokale Variable
        SP
```

Falls die Subroutine weiteren Speicher benötigt, wächst der Stack nach „unten".

Kurz vor Ende der Subroutine (d. h. vor RTS) versetzt UNLK die Pointer wieder in den Zustand vor dem Subroutinenaufruf.

Die beiden bisher erwähnten Fälle sind die zwei Pole der Parameterübergabe des 68000. Auf der einen Seite werden die Parameter einfach in Datenregister (bei der indirekten Parameterübergabe auch in Adreßregister) geladen, während andererseits der speziell strukturierte Stack die Parameterübergabe von Hochsprachen ermöglicht. Für den normalen Gebrauch reicht die Übergabe per Register jedoch oft nicht aus, während die Stacksteuerung zu kompliziert ist.

Hier schaffen andere Methoden Abhilfe. So wird zum Beispiel ein Datenbereich festgelegt, der die Ein- und Ausgabeparameter einer Gruppe von Subroutinen (d. h. eines Moduls) enthält. Dieser Datenbereich ist nicht global, sondern wird nur von dieser Subroutinengruppe für die Parameterübergabe eingesetzt. Dabei holt sich jede Subroutine ihre Variablen aus einem festen Speicherbereich und speichert dort auch die Ergebnisse. Hier ein Beispiel:

```
CALC  LEA    INPUTS,A4   * auf Eingabe-
                         parameter zei-
                         gen
      MOVE   (A4),D0     * ersten Para-
                         meter holen
      MOVE   (A4)+,D1    * zweiten Para-
                         meter holen
      . . . Programmcode
      MOVE   D5,OUTPUT   *Ausgabepara-
                         meter spei-
                         chern
      RTS
```

Wenn Sie den Pointer auf einen Parameterbereich in einem definierten Adreßregister über-

geben, wird die erste Zeile des Beispiels überflüssig. Diese Methode arbeitet ebenfalls mit separaten Stackbereichen für die Parameter INPUTS und OUTPUT.

Eine zweite Methode verwendet zwar Register, setzt damit aber den Befehl MOVEM ein. Sehen wir uns den Befehl einmal genauer an.

```
MOVEM Registerliste, SAVEAREA
```

legt die in der Registerliste angegebenen Register in einen Speicherbereich ab, der bei der absoluten Adresse SAVEAREA anfängt.

```
MOVE D3/D5/A2,SAVEAREA
```

lädt D3 in SAVEAREA, D5 in SAVEAREA+2 und A2 in SAVEAREA+4. MOVEM vereinfacht diesen Vorgang:

```
MOVEM D3/D5/A2,—(SP)
```

Der Befehl stapelt die Register in aufeinanderfolgenden, absteigenden Adressen. Beim Sichern einer Reihe von Registern erlaubt der 68000 auch folgende Kurzversion:

```
MOVEM D1—D5/A4,—(SP)
```

Hier werden D1 bis D5 und A4 gespeichert.

MOVEM kann die Register (als Zieloperand) aber auch zurückladen:

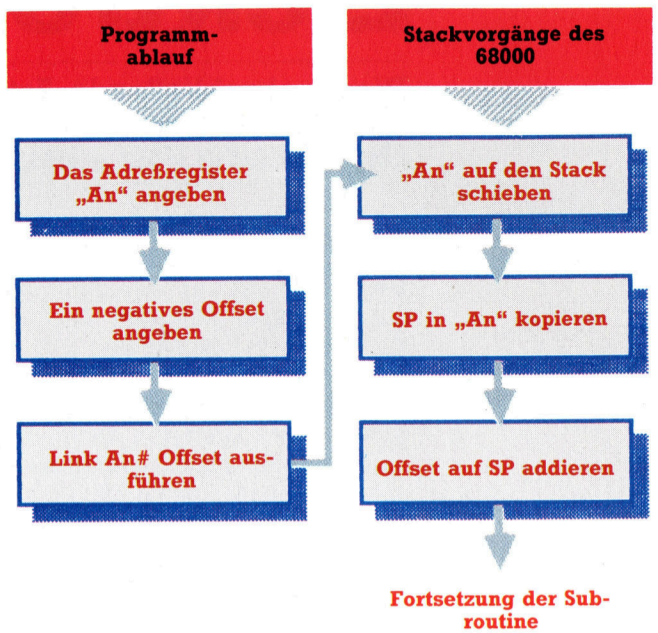
```
MOVEM—(SP),D1—D5/A4
```

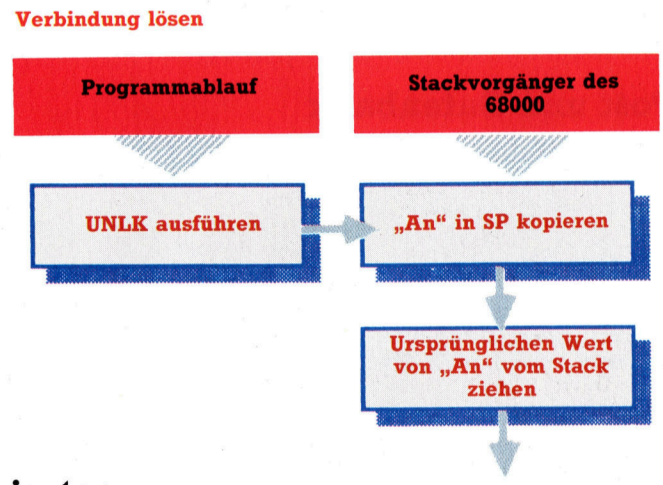
setzt die Register wieder auf ihre ursprünglichen Inhalte.

Bei der Parameterübergabe lassen sich mit dieser Methode leicht Register freisetzen, die sonst für spezielle Zwecke gebraucht werden (z. B. zur Systemsteuerung). Wenn beispielsweise D0 und D1 als Parameterregister definiert sind und (die sonst für Systemzwecke reservierten Register) D2 bis D7 für die Subroutine eingesetzt werden sollen, kann MOVEM sie für die Dauer der Subroutine freistellen. Z. B.:

```
CALC MOVEM  D2—D7,—(SP) * alten Inhalt
                          von D2—D7
                          sichern
     MOVE   D0,D2       * ersten Pa-
                          rameter la-
                          den
     . . .              * Code, der
                          D2—D7 ein-
                          setzt
     MOVEM  —(SP),D2—D7 * alten Inhalt
                          zurückladen
     RTS
```

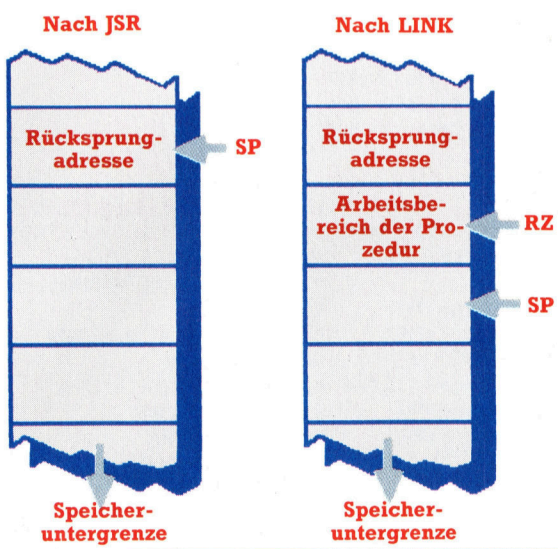
Sie sehen, der 68000 hat mehrere Möglichkeiten, Parameter an Subroutinen zu übergeben, und verfügt für diese Aufgabe auch über entsprechende Befehle.





Reservierte Plätze

Mit LINK und UNLK können Programme sehr schnell einen Stackbereich für Subroutinen reservieren. Dies ist besonders bei Mehrplatzanwendungen wichtig, da Subroutinen hier von anderen Programmen unterbrochen werden können. LINK bewegt den Stackpointer um einen Distanzwert abwärts. Das Register „An“ dient als „Rahmenzähler“, mit dem die Subroutine auf die Daten des reservierten Speicherblocks zugreift. UNLK stellt schließlich die ursprünglichen Werte des SP und von „An“ wieder her.



Elektronische Post

Dieser abschließende Artikel unserer Unix-Reihe befaßt sich mit weiteren Annehmlichkeiten, die das System bietet, z. B. die elektronische Post – bei Unix ein Systembestandteil.

Elektronische Briefkästen stehen zwar seit einiger Zeit zur Verfügung, das Verfahren hat sich aber aufgrund der umständlichen Handhabung, Langsamkeit und geringen Zuverlässigkeit nur wenig durchsetzen können. Bei Unix ist die elektronische Post schon eingebaut. Sie erlaubt den Nachrichtenaustausch zwischen Benutzern derselben Anlage und über geeignete Kanäle auch die Kommunikation zwischen zwei beliebigen Rechnern mit Unix-System. Beide Verfahren laufen nach dem gleichen Schema ab.

Das elektronische Postsystem wird durch den Befehl „mail" aktiviert; damit ist Ihre persönliche Briefkastendatei eröffnet und der Zugriff auf alle an Sie gerichteten Briefe möglich. Die eintreffenden Botschaften tragen eine Nummer und sind mit der Benutzerkennung des Absenders versehen, auch die Weitergabe von Kopien an andere Teilnehmer ist vermerkt. Neue und ungelesene Post wird beim Einloggen angezeigt.

Ingres

Das Postsystem meldet sich mit dem Promptzeichen „&". Es gibt eine Vielzahl von Befehlen für die Handhabung der Mitteilungen und für diverse Systemfunktionen wie das Wechseln der Directory. Zum Ausstieg aus dem Postmodus tippen Sie entweder „x" oder „q" ein, je nachdem ob die Briefe (soweit ungelöscht) im

Post und Datenbank

```
Berkeley 4.2 Vax/Unix (infc3)
Type (Ctrl-D) to disconnect

login: com-mcc
Password:
You are a normal user (class 3)
Jobs: 6 Superiors: 2 Maximum: 21
Last login: Thu Nov 14 13:47:42 on ttyn04

You have mail.                    (Hinweis, daß Post vorliegt)

%mail
Mail version 2.18 5/19/83. Type ? for help.
"/usr/spool/mail/com-mcc": 1 message 1 new
>N 1 com-vjp Fri Nov 15 09:52 11/271 "Specimen mailing"
&                 (Mail-Prompt; nach Drücken von Return Anzeige
                                        von Neueingängen)
Message 1:
From com-vjp Fri Nov 15 09:52:35 1985
From: com-vjp (Vicki)
To: com-daf, com-mcc
Subject: Specimen mailing         (Betreff: Musterbrief)
Hello                             (Inhalt der Nachricht)

& ?                               (Auflisten der verfügbaren Befehle)
  Mail      Commands
t <message list>        types messages
n                       goto and type next message
e <message list>        edit messages
f <message list>        give head lines of messages
d <message list>        delete messages
s <message list> file   appends messages to files
u <message list>        undelete messages
r <message list>        reply to messages
pre <message list>      make messages go back to /usr/mail
m <user list>           mail to specific users
q                       quit, saving unresolved messages in mbox
x                       quit, do not remove system mailbox
h                       print out active message headers
!                       shell escape
c [directory]           chdir to directory or home if none given
A <message list>        consists of integers, ranges of same, or user names
                        separated by spaces. If omitted, Mail uses the last
                        message typed.
A <user list>           consists of user names or distribution names
                        separated by spaces .
Distribution names are defined in .sendrc in your home directory.

% ingres demo      (Aufruf der Ingres-Datenbank mit Vorführdaten)

INGRES version 7.10 (10/27/81) login
Fri Nov 15 10:28:42 1985
go
* print parts      (Bereitstellung von Kommandos im Arbeits-
                                        bereich; Aktivierung durch /g)
*\g
Executing...

parts relation
| pnum | pname              | colour    | weight | qoh   |
|------|--------------------|-----------|--------|-------|
|   10 | byte-soap          | clear     |      0 |   143 |
|    1 | central processor  | pink      |     10 |     1 |
|   11 | card reader        | grey      |    327 |     0 |
|    2 | memory             | grey      |     20 |    32 |
|   12 | card punch         | grey      |    427 |     0 |
|    3 | disk drive         | black     |    685 |     2 |
|   13 | paper tape reader  | black     |    107 |     0 |
|    4 | tape drive         | black     |    450 |     4 |
|   14 | paper tape punch   | black     |    147 |     0 |
|    5 | tapes              | grey      |      1 |   250 |
|    6 | line printer       | yellow    |    578 |     3 |
|------|--------------------|-----------|--------|-------|
continue
* \p                      (Aktueller Inhalt des Arbeitsbereichs)
print parts
```

Briefkasten bleiben und beim nächsten Aufruf wieder avisiert werden sollen, oder ob sie zur Aufbewahrung in eine „mbox"-Datei kommen sollen. Was darin abgelegt ist, läßt sich zwar wie jede Textdatei bearbeiten, aber nicht mehr direkt als „Post" erreichen.

Ein Postversand innerhalb des eigenen Systems kann über den Befehl „m" erfolgen, nach außen über „mail", jeweils mit dem Adressatennamen dahinter, und zwar in Form der normalen Login-Benutzerkennung. Die Liste kann beliebig lang sein, alle angeführten Teilnehmer erhalten eine eigene Briefkopie.

Wenn Sie regelmäßig Post an denselben Empfängerkreis verschicken, können Sie sich mit einer „alias"-Vereinbarung das wiederholte Eintippen sämtlicher Namen ersparen. Für die Festlegung solcher Parameter ist eine spezielle Datei vorgesehen. Dort können Sie auch einen Vermerk (.mailrc oder .sendrc) hinterlassen, wenn Sie über alle Posteingänge sofort informiert werden möchten.

„Ingres" ist in den meisten Unix-Fassungen bereits enthalten, andernfalls aber nachträglich zu implementieren. Die Benutzerschnittstelle ist zwar weniger komfortabel als bei anderen Datenbanken, aber dafür ist dies relationale System auch sehr viel leistungsfähiger. Wie fast alles bei Unix kann auch Ingres den individuellen Erfordernissen des Benutzers angepaßt werden. Der unten wiedergegebene Bildschirmdialog bietet dazu ein Beispiel.

Texteditor

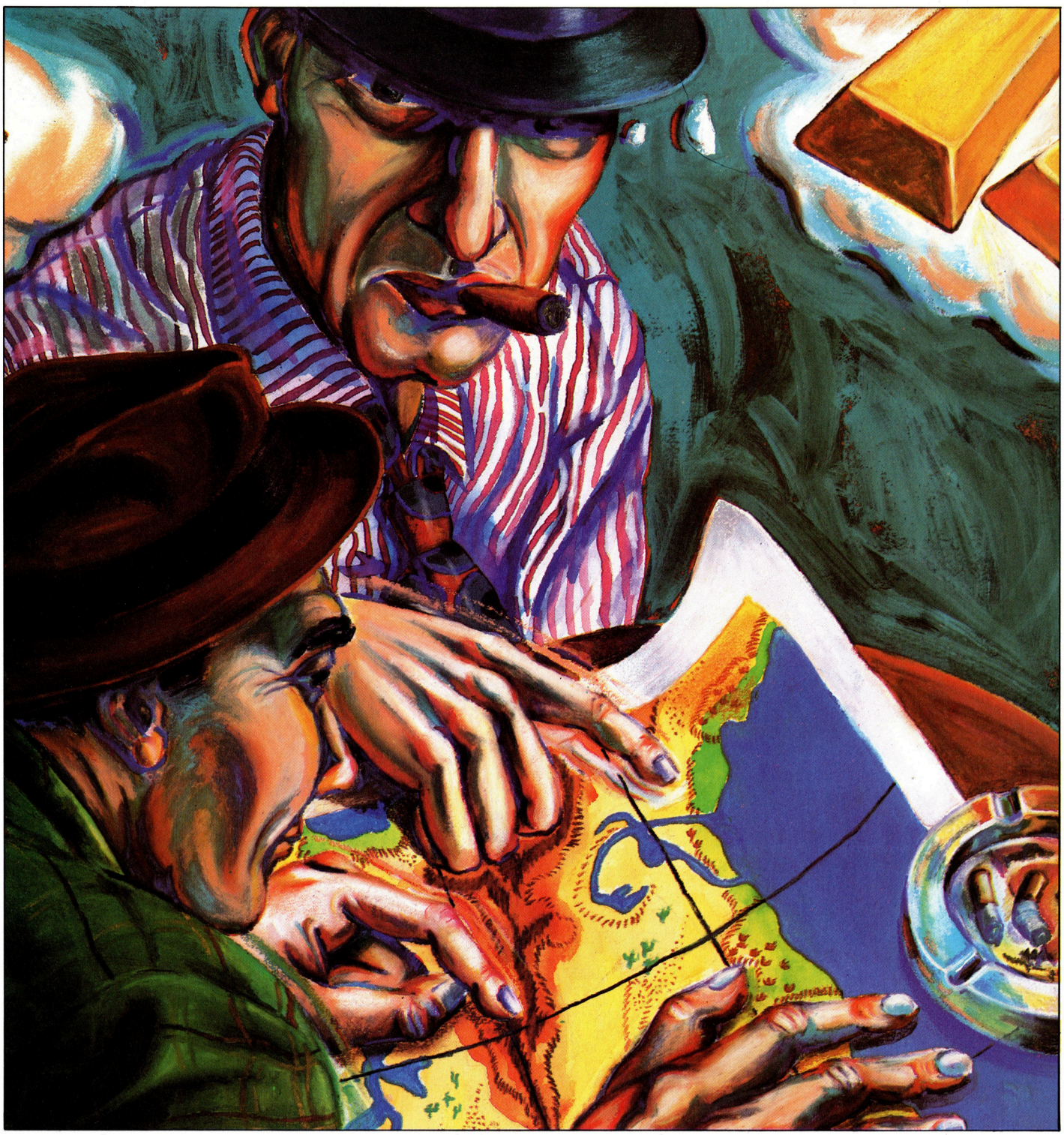
Ohne Texteditor wäre kein Betriebssystem komplett. Bei Unix gibt es davon standardmäßig gleich drei, und bei Bedarf lassen sich noch weitere einbauen. Der einfachste Editor ist „ed". Er ist ziemlich unhandlich, hat aber den Vorteil, daß sämtliche Editierfunktionen als Befehle verfügbar sind. Sie können deshalb Kommandodateien für die automatische Bearbeitung von Dokumenten zusammenstellen. Die beiden anderen Editoren heißen „ded" und „vi"; sie sind wahlweise zeilen- oder bildschirmorientiert zu verwenden.

Abschließend weisen wir noch einmal darauf hin, daß hier aus Platzgründen nur ein kleiner Teil der Möglichkeiten von Unix gezeigt werden konnte. Unix stellt zweifellos eins der wichtigsten modernen Betriebssysteme dar und hat viele Gestaltungsimpulse gegeben. Sein Einfluß ist unverkennbar.

```
continue
* \r                          (Arbeitsbereich räumen)
go
* range of p is parts         (Spezifizierung eines bestimmten
                               Zugriffsbereichs, Kurzbezeichnung „p")
* retrieve (p.pname)          (Feld „pname" aufsuchen und anzeigen)
* \g
Executing...

| pname                  |
|------------------------|
| byte-soap              |
| central processor      |
| card reader            |
| memory                 |
| card punch             |
| disk drive             |
| paper tape reader      |
| tape drive             |
| paper tape punch       |
| tapes                  |
| line printer           |
|------------------------|
(11 tuples)

continue
* retrieve (p.pname,p.coclour)  (Ansprechen mehrerer Felder)
* where p.colour = "grey"       (Angabe des Suchkriteriums)
* \g
Executing...

2100: line 1, Attribute 'coclour' not in relation parts
(Fehlermeldung: Das falsch getippte Feld „colour" ist nicht
auffindbar)
* retrieve (p.pname,p.colour)
* where p.colour = "grey"
* \g
Executing...

| pname              | colour   |
|--------------------|----------|
| card reader        | grey     |
| memory             | grey     |
| card punch         | grey     |
| tapes              | grey     |
|--------------------|----------|
(4 tuples)

continue
* retrieve (p.pname,p.pnum,total=p.qoh * p.weight)
(Aus den Datenbankfeldern lassen sich neue Werte berechnen
                                        und mit einem Titel versehen)
* \g
Executing...

| pname              | pnum |  total |
|--------------------|------|--------|
| byte-soap          |   10 |      0 |
| central processor  |    1 |     10 |
| card reader        |   11 |      0 |
| memory             |    2 |    640 |
| card punch         |   12 |      0 |
| disk drive         |    3 |   1370 |
| paper tape reader  |   13 |      0 |
| tape drive         |    4 |   1800 |
| paper tape punch   |   14 |      0 |
| tapes              |    5 |    250 |
| line printer       |    6 |   1734 |
|--------------------|------|--------|
(11 tuples)

continue
* \q                          (Rückkehr zur Shell-Ebene)
INGRES version 7.10 (10/27/81) logout
Fri Nov 15 10:41:03 1985
goodbye com-mcc -- come again
% logout
```

Im Goldrausch

Mit unserem neuen Goldgräber-Spiel können Sie ohne finanzielles Risiko ins Goldgeschäft einsteigen. Es geht in diesem neuen Strategiespiel um hohe Einsätze – sind Sie gerissen genug, die Konkurrenz auszustechen?



Goldmine ist ein Strategiespiel, in dem Sie die Inhaberrolle einer Minengesellschaft übernehmen. Ihre Aufgabe ist es, möglichst hohe Gewinne zu erwirtschaften.

Strategiespiele wie auch Adventures sind meist in BASIC geschrieben, weil die Geschwindigkeit der Assembler-Programmierung unnötig ist. Wir haben Goldmine mit Grafik versehen, die den Fortschritt des Abbaus zeigt. Damit ist die Anforderung an die Speicherkapazität zwar größer, das Programm paßt aber trotzdem in einen 16K-Spectrum und sogar in den VC 20.

Weil das Programm lang ist, haben wir es in zwei Teile zerlegt. Diese Teilung wird auch in der Programmbeschreibung beibehalten. Nach Eingabe von Teil 1 sollten Sie diesen mit SAVE abspeichern. Bei manchen Rechnern können Sie jetzt bereits starten und einige Statusmeldungen und Optionen sehen. Weil der zweite Teil fehlt, werden Sie allerdings Fehlermeldungen erhalten.

Zu Beginn des Spieles besitzen Sie bereits die Bergwerksgesellschaft und zwei Millionen Dollar in bar. Dieses Geld soll in die Goldsuche investiert werden. Es sind dann in 30 Runden so viel Gewinn wie möglich zu erzielen. Sie können allein oder gegen jemand spielen, der ein Konkurrenzunternehmen hat.

Neues Glück

Bevor Sie eine Mine ausbeuten können, müssen Sie erst ein passendes Gelände gefunden haben. Dafür muß ein Gutachten angefordert werden, welches Ihnen etwas über die Wahrscheinlichkeit einer erfolgreichen Grabung sagt. Sie müssen dann entscheiden, ob sich die Ausbeutung der Mine lohnt.

Haben Sie sich für ein Gelände entschieden, finden dort aber kein Gold, so können Sie an einer anderen Stelle erneut Ihr Glück versuchen.

Haben Sie Gold gefunden, so können Sie das abgebaute Gold entweder im Tresor verwahren oder sofort an der Börse verkaufen. Bargeld hat Vorteile; allerdings gibt es Goldräuber, die Ihren Schatz auf listige Weise rauben könnten.

Spectrum

```
5 BORDER 6: PAPER 6: INK Ø: CLS
1Ø PRINT AT 9,2;"How many players? (1 or
   2)": LET a$=INKEY$: IF a$="" THEN
   GOTO 1Ø
2Ø IF a$<"1" OR a$>"2" THEN GOTO 1Ø
3Ø LET p=VAL a$: LET nop=p
4Ø DIM a(2,6): DIM c(2,5): DIM a$(p,8): DIM
   r(2): LET er=1ØØØØ
5Ø LET r(1)=Ø: LET r(2)=Ø: LET
   a(1,1)=2ØØØØØØ: LET a(1,2)=2ØØØØØØ:
   LET a(2,1)=2ØØØØØØ: LET
   a(2,2)=2ØØØØØØ: LET a(1,3)=Ø: LET
   a(2,3)=Ø: LET a(1,4)=1ØØØØØ: LET
   a(2,4)=1ØØØØØ: LET a(1,5)=Ø: LET
   a(2,5)=Ø: LET a(1,6)=Ø: LET a(2,6)=Ø:
   PRINT
7Ø FOR n=1 TO p: INPUT "Name of
   player□";(n);"?", LINE a$(n): NEXT n
2ØØ FOR n=1 TO 3Ø: FOR m=1 TO nop
2Ø2 BORDER 7: PAPER 7: INK Ø: CLS
21Ø PRINT PAPER 6;n: PRINT PAPER
   1; INK 6;AT Ø,6;"□□G□O
   □L□D□M□I□N□E□□"
22Ø PRINT 'TAB 16;a$(1);: IF nop=2 THEN
   PRINT TAB 24;a$(2);
23Ø PRINT '"TOTAL ASSETS□$";TAB
   15;a(1,1);: IF nop=2 THEN PRINT TAB
   24;a(2,1);
24Ø PRINT '"CASH ASSETS□□$"; TAB
   15;a(1,2);: IF nop=2 THEN PRINT TAB
   24;a(2,2);
25Ø PRINT '"GOLD ASSETS□kg";TAB
   15;a(1,3);: IF nop=2 THEN PRINT TAB
   24;a(2,3);
26Ø PRINT '"COST TO MINE□$";TAB
   15;a(1,4);: IF nop=2 THEN PRINT TAB
   24;a(2,4);
27Ø PRINT '"NO. OF MINES";TAB 15;a(1,5);:
   IF nop=2 THEN PRINT TAB 24;a(2,5);
28Ø PRINT '"MINE DEPTH□□□m";TAB
   15;a(1,6);: IF nop=2 THEN PRINT TAB
   24;a(2,6);
3ØØ PRINT '' PAPER 4; INK Ø;"Current
   Exchange Rate:—": PRINT "$";er;"□per
   kg of gold"
4ØØ PRINT ' PAPER 5;">—";a$(m)
5ØØ PRINT PAPER 2; INK 7;"1";: PRINT "—
   Research and Development"
51Ø PRINT PAPER 2; INK 7;"2";: PRINT "—
   Exploration and Report"
52Ø PRINT PAPER 2; INK 7;"3";: PRINT "—
   Increase mine depth by 2ØØm"
53Ø PRINT PAPER 2; INK 7;"4";: PRINT "—
   Exchange gold for dollars"
54Ø PRINT PAPER 2; INK 7;"5";: PRINT "—
   Pass"
55Ø PRINT : PRINT FLASH 1; PAPER 1; INK
   6;"Enter your instruction"
6ØØ LET i$=INKEY$: IF i$="" THEN GOTO
   6ØØ
61Ø IF i$<"1" OR i$>"5" THEN GOTO
   6ØØ
62Ø GOSUB VAL i$*1ØØØ
7ØØ IF a(m,2)<Ø THEN GOTO 7ØØØ
71Ø LET er=er+INT (RND*1ØØØ)-2ØØ
72Ø IF INT (RND*16ØØ)-a(m,3)<Ø THEN
   GOSUB 9ØØ
74Ø LET a(m,1)=a(m,2)+a(m,3)*er
75Ø PAPER 7: INK Ø: BORDER 7: CLS
79Ø NEXT m
8ØØ NEXT n
81Ø PAPER 5: BORDER 5: INK Ø: CLS
82Ø PRINT FLASH 1; INK 7; PAPER 2;AT
   6,1Ø;"□GAME OVER□"
83Ø PRINT 'TAB 5;"Total Assets of□";a$(1):
   PRINT TAB 11;"$";a(1,1)
84Ø IF nop=2 THEN PRINT 'TAB 5;"Total
   Assets of □";a$(2): PRINT TAB
   11;"$";a(2,1)
85Ø PRINT '' PAPER 2; INK 6; FLASH 1;TAB
   2;"Press any key to play again"
86Ø IF INKEY$<>"" THEN GOTO 86Ø
87Ø IF INKEY$="" THEN GOTO 87Ø
88Ø RUN
9ØØ PAPER 2: INK 6: BORDER 2: CLS
9Ø5 LET jk=INT (RND*1ØØ)+5Ø: IF
   jk>a(m,3) THEN LET jk=a(m,3)
91Ø PRINT PAPER 6; INK 1; FLASH
   1;AT 9,8;"□R□O□B□
   B□E□R□Y□"
92Ø PRINT : PRINT INK 7;"□□□□□ You
   have had□";jk;"kg of": PRINT
   "□□□□your gold assets stolen": LET
   a(m,3)=a(m,3)-jk: LET
   a(m,1)=a(m,1)-(jk*er)
93Ø FOR x=1 TO 35: BEEP .Ø5,4Ø: BEEP
   .Ø5,2Ø: NEXT x
94Ø BORDER 7: PAPER 7: INK Ø: CLS :
   RETURN
```

Commodore 64

```
1 POKE52,48:POKE56,48:CLR:PRINT
   "♥" TAB(3)"DEFINING
   GRAPHICS,PLEASE WAIT . . ."
2 GOSUB6ØØØØ
3 POKE53272,28
5 POKE5328Ø,7:POKE53281,7:PRINT
   "♥■"
1Ø PRINT"■■■■■■■■■
   ■■■";TAB(5);"HOW MANY
   PLAYERS ? (1 OR 2)":GETA$:
   IFA$=""THEN1Ø
2Ø IFA$<"1"ORA$>"2"THEN1Ø
3Ø P=VAL(A$):NO=P
4Ø DIMA(2,6),C(2,5):ER=1ØØØØ
5Ø R(1)=Ø:R(2)=Ø:A(1,1)=2ØØØØØØ:
   A(1,2)=2ØØØØØØ:A(2,1)=2ØØØØØØ:
   A(2,2)=2ØØØØØØ
52 A(1,3)=Ø:A(2,3)=Ø:A(1,4)=
```

```
    100000:A(2,4)=100000:A(1,5)=0:
    A(2,5)=0:A(1,6)=0
54 A(2,6)=0:PRINT "♥"
70 FORN=1TOP:PRINT"NAME OF
    PLAYER"N;:INPUTA$(N):A$(N)=
    LEFT$(A$(N),10):NEXT
200 FORN=1TO30:FORM=1TONO
202 POKE53280,1:POKE53281,1:
    PRINT"■♥";
205 FORF=54272TO54296:POKEF,0:
    NEXT
210 PRINT"■■"N;TAB(10)"■□G□
    O□L□D□M□I□N□E□
    ■■"
220 PRINTTAB(16);A$(1);:IFNO=2
    THENPRINTTAB(28);A$(2);
230 PRINT:PRINT"■■TOTAL ASSETS";
    TAB(15);A(1,1);:IFNO=2THENPRINT
    TAB(27);A(2,1);
240 PRINT:PRINT"CASH ASSETS";
    TAB(15):A(1,2);:IFNO=2THEN
    PRINTTAB(27);A(2,2);
250 PRINT:PRINT"GOLD ASSETS KG ";
    TAB(15);A(1,3);:IFNO=2THEN
    PRINTTAB(27);A(2,3);
260 PRINT:PRINT"COST TO MINE $";
    TAB(15);A(1,4);:IFNO=2 THEN
    PRINTTAB(27);A(2,4);
270 PRINT:PRINT"NO. OF MINES";
    TAB(15);A(1,5);:IFNO=2THEN
    PRINTTAB(27);A(2,5);
280 PRINT:PRINT"MINE DEPTH M";
    TAB(15);A(1,6);:IFNO=2THEN
    PRINTTAB(27);A(2,6);
300 PRINT:PRINT"■■■CURRENT
    EXCHANGE RATE:—":PRINT"£"ER;
    "□PER KG OF GOLD"
400 PRINT "■■>■■■";A$(M)
500 PRINT"■■■1■□—□
    RESEARCH AND DEVELOPMENT"
510 PRINT"■2■□—□EXPLORATION
    AND REPORT"
520 PRINT"■3■□—□INCREASE
    MINE DEPTH BY 200M"
530 PRINT"■4■□—□EXCHANGE
    GOLD FOR DOLLARS
540 PRINT"■5■□—□PASS"
550 PRINT"■■■ENTER YOUR
    INSTRUCTION":POKE 198,0
600 GETI$:IFI$=""THEN600
610 IFI$<"1"ORI$>"5"THEN600
620 ONVAL(I$)GOSUB1000,2000,
    3000,4000
700 IFA(M,2)<0THEN7000
710 ER=ER+INT(RND(1)*1000)-200
718 J=0:K=0
720 IFINT(RND(1)*1600)-A(M,3)
    <0THENGOSUB900
740 A(M,1)=A(M,2)+A(M,3)*ER
750 POKE53280,1:POKE53281,1:
```

```
   PRINT"■ ♥"
790 NEXTM,N
810 POKE53280,3:POKE53281,3:
   PRINT"■ ♥"
820 PRINT"■■■■■■";TAB
   (15);"  GAME OVER■■■■"
830 PRINT"TOTAL ASSETS OF□";
   A$(1);TAB(18);"□ARE $"A(1,1)
840 IFNO=2THENPRINT"TOTAL
   ASSETS OF□";A$(2);TAB(18);
   "□ARE $"A(1,2)
850 PRINT:PRINT:PRINT"  ■■■■
   ■■■■  PRESS ANY KEY TO PLAY
   AGAIN"
860 POKE 198,0
870 IFPEEK(197)=64THEN870
880 RUN 3
900 POKE53280,2:POKE53281,2:
   PRINT"♥ π"
905 JK=INT(RND(1)*100)+50:IF
   JK>A(M,3)THENJK=A(M,3)
910 PRINT"■■■■■■■■";
   TAB(12)"  □R□O□B□
   B□E□R□Y□"
920 PRINTTAB(9)"■  YOU HAVE HAD
   "JK"KG OF":PRINTTAB(7)"YOUR GOLD
   ASSETS STOLEN"
925 A(M,3)=A(M,3)-JK:A(M,1)=
   A(M,1)-(JK*ER)
930 FORF=0TO24:POKE54272+F,0:
   NEXT
940 POKE54286,5:POKE54290,16:
   POKE54275,1:POKE54296,143:
   POKE54278,240
950 POKE54276,65:FR=5389:FOR
   T=1TO150
960 FQ=FR+PEEK(54299)*9.5:
   HF=INT(FQ/256):LF=FQ-HF*256:
   POKE54272,LF
965 POKE54273,HF:NEXT:POKE
   54296,0
970 POKE53280,1:POKE53281,1:
   PRINT"♥■":RETURN
```

Commodore VC 20

```
5 POKE36879,25:PRINT"♥  ":POKE
   36878,15
10 PRINT"  HOW MANY PLAYERS ?":
   PRINT"    (1 OR 2)":GETA$:
   IFA$=""THEN10
20 IFA$<"1"ORA$>"2"THEN10
30 P=VAL(A$):NO=P
40 DIMA(2,6),C(2,5):ER=10000
50 R(1)=0:R(2)=0:A(1,1)=2000000:
   A(1,2)=2000000:A(2,1)=2000000:
   A(2,2)=2000000
52 A(1,3)=0:A(2,3)=0:A(1,4)=
   100000:A(2,4)=100000:A(1,5)=0:
   A(2,5)=0:A(1,6)=0
54 A(2,6)=0:PRINT "♥"
70 FOR N=1TOP:PRINT"■■  NAME
   OF PLAYER ";N:INPUTA$(N):
   A$(N)=LEFT$(A$(N),9):NEXT
200 FORN=1TO30:FORM=1TONO
210 PRINT"♥■ "N;TAB(4)"■ □G□
   O□L□D□M□I□N□E□□ "
220 PRINT"  "A$(1);:IFNO=2THEN
   PRINTTAB(11);A$(2);
230 PRINT:PRINT"  T"A(1,1);:IF
   NO=2THENPRINTTAB(11);A(2,1)
240 PRINT:PRINT"C"A(1,2);:IFNO=
   2THENPRINTTAB(11);A(2,2)
250 PRINT:PRINT"G"A(1,3);:IFNO=
   2THENPRINTTAB(11);A(2,3);
260 PRINT:PRINT"M"A(1,4);:IFNO=
   2THENPRINTTAB(11);A(2,4);
270 PRINT:PRINT"N"A(1,5);:IFNO=
   2THEN PRINTTAB(11);A(2,5);
280 PRINT:PRINT"D"A(1,6);:IFNO=
   2THENPRINTTAB(11);A(2,6);
300 PRINT:PRINT"  CURRENT EXCHANGE
   RATE:$"ER;"PER KG"
400 PRINT"  "A$(M)
500 PRINT"■ 1 ■RESEARCH,
   DEVELOPMENT"
510 PRINT"■ 2 ■EXPLORATION,
   REPORT"
520 PRINT"■ 3 ■INCREASE MINE
   DEPTH□□BY 200M"
530 PRINT"■ 4 ■EXCHANGE GOLD
   FOR□□□□DOLLARS"
540 PRINT"■ 5 ■PASS"
550 PRINT"  ENTER YOUR
   INSTRUCTION":POKE 198,0
600 GETI$:IFI$=""THEN610
610 IFI$<"1"ORI$>"5"THEN600
620 ONVAL(I$)GOSUB1000,2000,
   3000,4000
700 IFA(M,2)<0THEN7000
710 ER=ER+INT(RND(1)*1000)-200
718 J=0:K=0
720 IFINT(RND(1)*1600)-A(M,3)
   <0THENGOSUB900
740 A(M,1)=A(M,2)+A(M,3)*ER
750 PRINT"■♥"
790 NEXTM,N
810 PRINT"■♥"
820 PRINT"■■■■■■■■
   ■■■  GAME OVER■■■■"
830 PRINT"TOTAL ASSETS OF":
   PRINTA$(1)"ARE $"A(1,1)
840 IFNO=2THENPRINT"TOTAL ASSETS OF
   ":PRINTA$(2)"□ARE $"A(1,2)
850 PRINT"■■  PRESS ANY KEY TO
   PLAY"
860 POKE198,0
870 IFPEEK(197)=64THEN870
880 RUN
900 PRINT"♥  "
905 JK=INT(RND(1)*100)+50:IFJK
   >A(M,3)THENJK=A(M,3)
910 PRINT"■■■■■■■■■
   ■■  □R□O□B□B□E□R□Y□"
915 FORDE=1TO100:POKE36875,200
   +SIN(DE)*10:NEXTDE:POKE36875,0
920 PRINT"■■  YOU HAVE HAD"
   JK"KG":PRINT"OF YOUR GOLD
   ASSETS■■■STOLEN"
923 FORDE=1TO3000:NEXT
925 A(M,3)=A(M,3)-JK:A(M,1)=
   A(M,1)-(JK*ER)
970 PRINT"♥■":RETURN
```

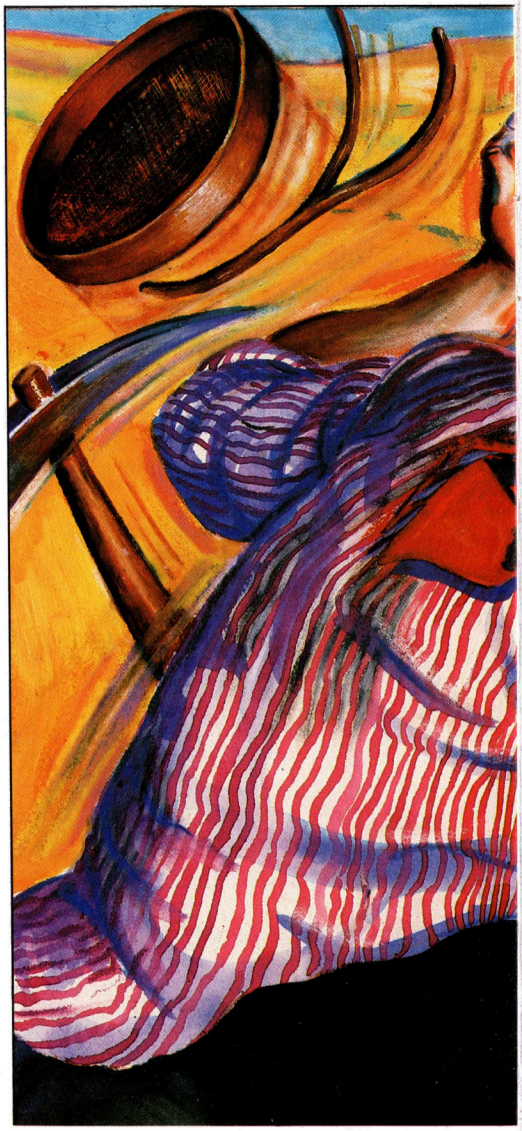
Acorn B

```
1 MODE1
3 *FX11,0
4 VDU 23;8202;0;0;0;
5 FOR T=224 TO 238:VDU23,T:FOR P=1
   TO 8:READ A:VDU A:NEXT:NEXT
10 VDU 19,2,4,0,0,0,19,0,2,0,0,0:
   PRINTTAB(7,10)"HOW MANY PLAYERS (1
   OR 2)":A$=GET$
20 IF A$<>"1" AND A$<>"2" THEN 10
30 P=VAL(A$):NOP=P
40 DIM A(2,6),C(2,5),A$(P),R(2):
   ER=10000
50 R(1)=0:R(2)=0:A(1,1)=2000000:
   A(1,2)=2000000:A(2,1)=2000000:
   A(2,2)=2000000:A(1,3)=0:A(2,3)=
   0:A(1,4)=100000:A(2,4)=100000:
   A(1,5)=0:A(2,5)=0:A(1,6)=0:
   A(2,6)=0:PRINT
70 FOR N=1 TO P:PRINT"NAME OF PLAYER
   ";N;:INPUTA$(N):NEXT
200 FOR N=1 TO 30:FOR M=1 TO NOP
202 COLOUR131:COLOUR0:CLS
210 COLOUR130:PRINT;N;TAB(10,0)
   "□G□O□L□D□M□I□N□E□":
   COLOUR131
220 PRINTTAB(20,3)A$(1);:IF NOP=2 THEN
   PRINTTAB(30,3)A$(2);
230 PRINT'"TOTAL ASSETS□$"TAB(19)
   A(1,1);:IF NOP=2 THEN PRINTTAB
   (29)A(2,1);
240 PRINT'"CASH ASSETS□□$"TAB(19)
   A(1,2);:IF NOP=2 THEN PRINTTAB
   (29)A(2,2);
250 PRINT'"GOLD ASSETS□□$"TAB(19)
   A(1,3);:IF NOP=2 THEN PRINTTAB
   (29)A(2,3);
260 PRINT'"COST TO MINE□$"TAB(19)
   A(1,4);:IF NOP=2 THEN PRINTTAB
   (29)A(2,4);
270 PRINT'"NO. OF MINES□□"TAB(19
   A(1,5);:IF NOP=2 THEN PRINTTAB
   (29)A(2,5);
280 PRINT'"MINE DEPTH□□□m"TAB
   (19)A(1,6);:IF NOP=2 THEN PRINT
   TAB(29)A(2,6);
300 COLOUR1:PRINT'"CURRENT EXCHANGE
```

```
    RATE;:—S";ER"□PER kg OF GOLD"
400 PRINT:COLOUR131:COLOUR2:PRINT
    ">—";A$(M)
500 PRINT""1□—RESEARCH AND
    DEVELOPMENT"
510 PRINT"2□—EXPLORATION AND
    REPORT"
520 PRINT"3□—INCREASE MINE DEPTH BY
    200m"
530 PRINT"4□—EXCHANGE GOLD FOR
    DOLLARS"
540 PRINT "5□—PASS"
550PRINT""ENTER YOUR INSTRUCTION"
600 I$=GET$
610 IF I$<"1" OR I$>"5" THEN 600
620 GOSUB VAL I$*1000
700 IF A(M,2)<0 THEN 7000
710 ER=ER+RND(1000)-200
720 IF RND(1600)-A(M,3)<0 THEN
    GOSUB 900
740 A(M,1)=A(M,2)+A(M,3)*ER
750 COLOUR131:COLOUR0:CLS
790 NEXT
800 NEXT
810 CLS
820 PRINTTAB(15,12)"GAME OVER"
830 PRINT'TAB(5)"TOTAL ASSETS
    OF□"A$(1)"□ARE□$";A(1,1)
840 IF NOP=2 THEN PRINT'TAB(5)
      "TOTAL ASSETS OF□"A$(2)"□ARE
    □$";A(2,1)
850 PRINTTAB(0,30)"PRESS ANY KEY TO
    PLAY AGAIN"
860 G=GET
880 RUN
900 COLOUR129:COLOUR3:CLS
905 JK=RND(100)+50:IF JK>A(M,3)
    THEN JK=A(M,3)
910 PRINTTAB(12,13)"□R□O□B□
    B□E□R□Y□"
920 PRINT"""""□□□YOU HAVE
    LOST□";JK;"□kg OF YOUR
    GOLD":A(M,3)=A(M,3)-JK:A(M,1)
    =A(M,1)-JK*ER
930 FOR X=1 TO 35:SOUND1,-15,40,1:
    SOUND1,-15,100,1:NEXT
940 COLOUR131:COLOUR0:CLS:RETURN
```

Tandy/Dragon

```
10 PMODE 3,1:CLS
20 DIM H(23),T(0),D(1),B(1),A(1,5),
    C(1,4),A$(1),R(1)
40 FOR K=0 TO 9:READ NU$(K):NEXT
50 DATA NR2D4R2U4BR2,BFEND4BR2,
    R2D2L2D2R2BU4BR2,NR2BD2NR2BD2R2
    U4BR2,D2R2D2U4BR2,NR2D2R2D2L2BE4,
    D4R2U2L2BE2BR2,R2ND4BR2,NR2D4R2U2
    NL2U2BR2,NR2D2R2D2U4BR2
60 PCLS3
70 DRAW"BM36,23C2L35U6E3R3NU4R5U
    10E2RE3R3F4D3F4DF2DF2DF3D2":
    PAINT(18,16),2
80 DRAW"BM24,9C3G2D6F5R3E2UH2UHU
    H2UH2BM20,1NLDL2GR5D5BM14,6RBR3
    RBD4DBL4UBD3LBR4RBD2LBL3LBD2RBR
    3RBD2LBL3LBD2RBR3RBD2LBL3L":
    PAINT(26,15),3
90 DRAW"BM2,21C4UBR4ND3BR4D":
    GET(0,0)-(37,23),H,G
100 PCLS:DRAW"BM7,0C4L6BD2ERFRE
    RBD2L7DR7DL5GNR3DNR3FNR4DNR
    4GNR3DNR3FNR4DR2GL3FNR6FR3FL
    4GNRDR5DL3"
110 GET(0,0)-(7,2),T,G:GET(0,3)
    -(7,10),D,G:GET(0,11)-(7,20), B,G
120 PRINT@129,"HOW MANY PLAYERS (1
    OR 2)□?";
130 A$=INKEY$:IF A$<"1" OR A$>
    "2" THEN 130
140 P=VAL(A$):NO=P:ER=10000:
    A(0,0)=2000000:A(0,1)=2000000:
    A(1,0)=2000000:A(1,1)=2000000:
    A(0,3)=100000:A(1,3)=100000
150 FORN=1TOP:PRINT:PRINT:PRINT
    "□NAME OF PLAYER";N;:LINE
    INPUTA$(N-1):IF LEN(A$(N-1))
    >8 THEN A$(N-1)=LEFT$(A$
    (N-1),8)
160 NEXT
200 FORN=0TO29:FORM=0TONO-1
202 CLS
210 PRINT@3,"goldmine";
220 PRINTTAB(15);A$(0);:IF NO=2
    THEN PRINTTAB(24);A$(1)
230 PRINT@32,"TOTAL ASSETS";TAB
    (14);A(0,0):IF NO=2 THEN PRINT
    @55,A(1,0)
240 PRINT@64,"CASH ASSETS";TAB
    (14);A(0,1):IF NO=2 THEN PRINT
    @87,A(1,1)
250 PRINT@96,"GOLD ASSETS kg";
    TAB(14);A(0,2):IF NO=2 THEN
    PRINT@119,A(1,2)
260 PRINT@128,"COST TO MINE";
    TAB(14);A(0,3):IF NO=2 THEN
    PRINT@151,A(1,3)
270 PRINT@160,"NO. OF MINES";
    TAB(14);A(0,4):IF NO=2 THEN
    PRINT@183,A(1,4)
280 PRINT@192,"MINE DEPTH m";
    TAB(14);A(0,5):IF NO=2 THEN
    PRINT@215,A(1,5)
300 PRINT@224,"CURRENT EXCHANGE
    RATE:—":PRINTER;"PER KG OF GOLD"
400 PRINT@330,A$(M)
500 PRINT"1—RESEARCH AND
    DEVELOPMENT"
510 PRINT"2—EXPLORATION AND REPORT"
520 PRINT"3—INCREASE MINE DEPTH BY
    200m"
```



```
530 PRINT"4—EXCHANGE GOLD FOR
    DOLLARS"
540 PRINT"5—PASS";
600 A$=INKEY$:IF A$<"1" OR A$>"5"
    THEN 600
620 ON VAL(A$) GOSUB 1000,2000,
    3000,4000,5000
700 IF A(M,1)<0 THEN 7000
710 ER=ER+RND(1000)-200
720 IF RND(1600)-A(M,2)<0 GOSUB 900
740 A(M,0)=A(M,1)+A(M,2)*ER
750 CLS
790 NEXTM,N
810 CLS
820 PRINT@138,"GAME OVER"
830 PRINT@197, "TOTAL ASSETS OF ";
    A$(0):PRINTTAB(11);A(0,0)
840 IF NO=2 THEN PRINTTAB(5);"TOTAL
    ASSETS OF ";A$(1):PRINTTAB(11);A(1,0)
850 PRINT@449,"PRESS ANY KEY TO PLAY
    AGAIN"
```


```
860 IF INKEY$="" THEN 860 ELSE RUN
900 CLS
905 JK=RND(100)+49:IF JK>A(M,2)
    THEN JK=A(M,2)
910 PRINT@9,"R□O□B□B□E□R□Y"
920 PRINT:PRINT"□□□□□YOU HAVE
    HAD"; JK;"KG OF": PRINT
    "□□□□□YOUR GOLD ASSETS
    STOLEN":A(M,2)=A(M,2)-JK:
    A(M,0)=A(M,0)-JK*ER
930 PLAY"T401CDEFBAGFED"
940 CLS:RETURN
```

Spectrum

Zeile 10 legt die Mitspielerzahl fest, Zeile 20 sorgt dafür, daß die Eingabe im vorgegebenen Rahmen liegt. Zeile 30 setzt entsprechend der gewählten Spielerzahl die Variablen p und nop fest.

In Zeile 40 wird eine Reihe von Arrays DIMensioniert und der Goldpreis festgelegt. Array a speichert die Aktiva der einzelnen Mitspieler und ihrer Bergwerke, Array c enthält Informationen über die Minen. Im Array a$ stehen die Namen der Spieler, und Array r gibt an, ob in der vom Spieler gewählten Mine bereits gearbeitet wird. Zeile 50 initialisiert die Aktiva und den Status der Bergwerke beider Mitspieler. r(1) und r(2) sind Null, wenn das Bergwerk noch nicht ausgebeutet wird und die erste Mine noch auf Ergiebigkeit untersucht wird. Die anderen Werte entsprechen folgenden Faktoren: a(1,1) und a(2,1) enthalten den Wert des gesamten Besitzes der Spieler; a(1,2) und a(2,2) geben den jeweiligen Kassenstand an; a(1,3) und a(2,3) speichern den Goldbesitz. In a(1,4) und a(2,4) stehen die Betriebskosten. Die Anzahl der Minen speichern a(1,5) und a(2,5), die Tiefe der Schächte wird in a(1,6) und a(2,6) festgehalten. Zeile 70 ermöglicht die Eingabe der Spielernamen.

Commodore

Beim Commodore-64-Programm erzeugen die Zeilen 1 bis 5 die Grafik, wobei ein Unterprogramm ab Zeile 60000 die Grabungsarbeiten etc. darstellt. Beim VC 20 braucht man dafür eine 3K-RAM-Erweiterung. Die hochauflösende Grafik sollte durch die Blockgrafik des Gerätes ersetzt werden. In beiden Programmen legt Zeile 5 die Bildschirmfarbe fest und löscht den Schirm. Zeile 10 fragt die Spielerzahl ab, Zeile 20 prüft, ob dabei 1 oder 2 eingegeben wurde. In

Zeile 30 werden je nach eingegebenem Wert P und NO festgesetzt.

Zeile 40 dimensioniert zwei Arrays und legt den Goldkurs fest. Die Zeilen 50 bis 54 initialisieren die Arrays und löschen den Bildschirm. Die Werte des Arrays R geben an, ob die Arbeiten schon begonnen haben; wenn die Feldelemente 0 sind, herrscht noch Ruhe in der Mine. Das erste Elementepaar in A enthält die Aktivposten der beiden Spieler, das zweite Paar speichert den Kassenbestand, das dritte den Besitz an Gold, das vierte die Betriebskosten, das fünfte die Anzahl der Gruben und das sechste die Tiefe der jeweiligen Mine. Zeile 70 fragt nach dem Namen des (der) Spieler(s) und speichert die Antwort in Array A$.

Acorn B

Das Acorn-Programm ruft in Zeile 1 den MODE 1 auf. Zeile 3 und 4 schalten die Autorepeat-Funktion der Tastatur und des Cursors ab. Zeile 5 erzeugt durch Lesen der DATA-Zeilen am Programmende die Grafik des Spiels. Mit einer Reihe von VDU-Befehlen erzeugt Zeile 10 die Bildschirmfarben, bevor nach den Spielernamen gefragt wird. Zeile 20 prüft, ob sich die Anzahl im zulässigen Bereich befindet. Nach der Eingabe werden entsprechend P und NOP festgesetzt. In Zeile 40 werden die Arrays DIMensioniert, und der Goldpreis ER wird festgelegt. Die Elemente der Arrays bestimmt Zeile 50: R(1) und R(2) bleiben Null, solange noch nicht in den Minen gearbeitet wird. Das erste Elementepaar von Array A speichert die gesamten Aktiva der beiden Spieler, das zweite Paar den Barbestand, das dritte den Goldbesitz, das vierte die Betriebskosten, das fünfte die Zahl der Gruben, und Elementepaar 6 enthält die Schacht-Tiefe. Zeile 70 fragt nach dem (den) Spielernamen.

Tandy/Dragon

In der Programmversion für den Tandy und den Dragon schaltet Zeile 10 in den PMODE3 und löscht den Bildschirm. Zeile 20 DIMensioniert einige Arrays.

Da einige Teile des Spiels auf dem hochauflösenden Grafikbildschirm stattfinden, muß der Text gelegentlich über DRAW erzeugt werden. Die Zeilen 40 bis 110 enthalten ein Unterprogramm für das Erzeugen von Zahlen innerhalb der hochauflösenden Grafik.

Die Zeilen 120 und 130 erfragen die Anzahl der Spieler und überprüfen die jeweilige Eingabe. Entsprechend setzt Zeile 140 P und NO fest. Als nächstes werden einige der Elemente von Array A definiert: A(0,0) und A(1,0) beinhalten die gesamten Aktiva der Spieler, das nächste Elementepaar enthält die jeweiligen Barbestände, danach kommen der Goldbesitz und die Betriebskosten. Die Zeilen 150 und 160 fragen nach dem (den) Spielernamen.
Im weiteren sind die Programme untereinander weitgehend gleich: Alle enthalten zwei FOR . . . NEXT-Schleifen, die in Zeile 200 anfangen und in den Zeilen 790 und 800 abgeschlossen werden. Diese Schleifen erzeugen das Menü der Wahlmöglichkeiten und die Darstellung der aktuellen Situation der Gesellschaften (Bestände, Kosten etc.).

Bilanzen

Die Variable N (n beim Spectrum) zählt die Anzahl der Runden, die bereits durchlaufen worden sind. Die Variable nop (Spectrum), NO (Commodore, Dragon und Tandy) bzw. NOP (Acorn) gewährleistet, daß jeder Spieler 30mal an die Reihe kommt. Weiter hinten im Programm sorgen die gleichen Variablen für die Anzeige der jeweiligen Bilanzen der Spieler.

Beim Spectrum, Commodore und Acorn sorgt Zeile 202 für die Einstellung der Bildschirmfarben, während beim Dragon und Tandy lediglich der Bildschirm gelöscht wird. Die Zeile 205 des Commodore-Programms leert die Sound-Register, die für die Soundeffekte benötigt werden. In allen Programmen erzeugt Zeile 210 den Titelausdruck „GOLDMINE". Zeile 220 stellt den Namen des Spielers auf dem Bildschirm dar. Bei zwei Spielern wird nur der Name des zweiten angezeigt.

Die Zeilen 230 bis 300 zeigen den Gesamtbesitz (TOTAL ASSETS), Barbestand (CASH ASSETS), Goldbesitz (GOLD ASSETS), Betriebskosten (COST TO MINE), Anzahl der Gruben (NO. OF MINES), Schachttiefe (MINE DEPTH) und den Börsenkurs (EXCHANGE RATE) an. Bei zwei Spielern werden jeweils beide Werte angezeigt.

Zeile 400 zeigt den Namen des aktiven Spielers an. Die Zeilen 500 bis 540 ermöglichen eine Auswahl zwischen Forschung und Entwicklung (Research and Development), Erkundung und Gutachten (Exploration and Report), Erhöhung der Schachttiefe um 200 Meter (Increase mine depth by 200 metres), Wechseln von Gold in Dollars (Exchange gold for dollars) oder Weiter (Pass). In den Programmen für den Spectrum, Commodore oder Acorn wird der Spieler durch Zeile 550 nach seinen Anweisungen gefragt. Beim Dragon/Tandy-Programm geschieht dies nicht, weil der Bildschirm bereits voll belegt ist.

In den Zeilen 600 bis 620 werden INKEY$- oder GET$-Anweisungen verwendet, um die Eingaben des Spielers zu registrieren. Diese werden auf Gültigkeit geprüft und leiten weiter zu den Unterprogrammen der gewählten Option.

Zeile 700 prüft, ob die Bilanz des Unternehmens negativ ist, falls ja, wird das Spiel mit dem Abschluß-Unterprogramm beendet. Zeile 710 erzeugt zufällige Änderungen des Goldkurses.

In Zeile 720 wird eine Zufallszahl mit dem Goldbesitz verglichen. Dann wird entschieden, ob ein Goldraub stattfinden soll. Das Raub-Unterprogramm liegt zwischen Zeile 900 und 940. Die Zeile 905 legt fest, wieviel Gold gestohlen wird, Zeile 920 stellt die Menge am Bildschirm dar.

In Zeile 740 wird die Bilanz ermittelt. Zeile 350 erzeugt einen Reset der Bildschirmfarben und löscht den Bildschirm, bevor das Programm durch die NEXT-Anweisungen für eine weitere Runde nach Zeile 200 springt.

Die Zeilen 810 bis 840 bilden das Unterprogramm für das Spielende.

Ein Unterprogramm zwischen Zeile 850 und 880 fordert zu einer neuen Runde auf, in der Sie wieder Ihr Glück versuchen können.

Spectrum

```
1000 BORDER 6: PAPER 6: INK 0: CLS
1010 PRINT PAPER 1; INK 6;AT 3,4;
   "□RESEARCH & DEVELOPMENT□";
   AT 4,4;"(to lower mining costs) "
1020 PRINT AT 7,6;"How much would
   you";TAB 5;"like to invest ? ($)": INPUT rd
1050 LET a(m,4)=a(m,4)-INT (rd*.05)-1
1060 IF a(m,4)<0 THEN LET a(m,4)=0
1080 LET a(m,2)=a(m,2)-rd: LET
   a(m,1)=a(m,1)-rd
1100 PRINT AT 13,3;"Your mining costs will
   be";TAB 3;"reduced by $";INT
   (rd*.05)+1;"□per 200m"
1110 FOR z=1 TO 300: NEXT z
1120 RETURN
```

Commodore 64

```
1000 POKE53280,7:POKE53281,7:
   PRINT"■■"
1010 PRINT"■RESEARCH AND
   DEVELOPMENT":PRINT"(TO LOWER
   MINING COSTS)"
```

```
1020 PRINT "■■■■■HOW MUCH
   WOULD YOU LIKE TO INVEST
   ($)":INPUTRD
1030 R1=INT(RD*.5)-1
1050 A(M,4)=A(M,4)-R1
1060 IFA(M,4)<0THENA(M,4)=0
1080 A(M,2)=A(M,2)-RD:A(M,1)
   =A(M,1)-RD
1100 PRINT"■■■■■■■■■
   YOUR MINING COSTS WILL BE REDUCED
   BY:":PRINT"$";R1+1;
1110 PRINT "PER 200M":FORZ=1
   TO2300:NEXT
1120 RETURN
```

Commodore VC 20

```
1000 PRINT"■♥"
1010 PRINT"■RESEARCH,
   DEVELOPMENT":PRINT"■■
   TO LOWER MINING COSTS"
1020 RD=0:PRINT"■■■■■■
   HOW MUCH WOULD YOU□□□□LIKE
   TO INVEST ($)":INPUTRD
1030 R1=INT(RD*.5)-1
1050 A(M,4)=A(M,4)-R1
1060 IFA(M,4)<0THENA(M,4)=0
1080 A(M,2)=A(M,2)-RD:A(M,1)
   =A(M,1)-RD
1100 PRINT"■■■■■YOUR
   MINING COSTS WILL BE REDUCED
   BY:":PRINT"$";R1+1"PER 200M"
1110 FORZ=1TO2300:NEXT
1120 RETURN
```

Acorn B

```
1000 COLOUR129:COLOUR2:CLS
1010 PRINTTAB(8,3)"RESEARCH AND
   DEVELOPMENT"TAB(9,5)"(LOWERS
   MINING COSTS)"
1020 PRINT'"HOW MUCH WOULD YOU LIKE
   TO INVEST ($)":INPUTRD
1050 A(M,4)=A(M,4)-INT(RD*.05)
1060 IF A(M,4)<0 THEN A(M,4)=0
1080 A(M,2)=A(M,2)-RD:A(M,1)
   =A(M,1)-RD
1100 PRINTTAB(0,13)"YOUR MINING COSTS
   WILL BE REDUCED BY □□□□□$";
   INT(RD*.05);"□PER 200m"
1110 FOR Z=1 TO 4000:NEXT
1120 RETURN
```

Tandy/Dragon

```
1000 CLS
1010 PRINT@3,"research and
   development":PRINT@35,"(TO
   LOWER MINING COSTS)"
1020 PRINT:INPUT"HOW MUCH WOULD YOU
   LIKE TO□□□□□□□INVEST ($)";RD
1030 IF RD<0 THEN 1000
1050 A(M,3)=A(M,3)-INT(RD/20)-1
1060 IF A(M,3)<0 THEN A(M,3)=0
1080 A(M,1)=A(M,1)-RD:A(M,0)
   =A(M,0)-RD
1100 PRINT@257,"YOUR MINING
   COSTS WILL BE□□□□□□□□
   REDUCED BY $";INT(RD/20)+1;
   "PER 200m"
1110 FORZ=1TO2000:NEXT
1120 RETURN
```

Bei den Spectrum-, Commodore 64- und Acorn-Programmen erzeugt Zeile 1000 die Bildschirmfarben und löscht den Bildschirm. Beim VC 20 und der Dragon/Tandy-Version bleiben die Farben unverändert, der Bildschirm wird nur gelöscht. Zeile 1010 stellt die Überschrift dar, bevor mit Zeile 1020 gefragt wird, wie hoch die Investition in Forschung und Entwicklung sein soll. Die Variable RD (Spectrum rd) speichert diesen Zahlenwert.

Zeile 1050 senkt die Betriebskosten der Goldmine. Zeile 1060 sorgt dafür, daß die Betriebskosten nicht negativ werden. Die Anpassung der Bilanz und des Barbestandes nach der Investition erledigt Zeile 1080.

Das Ausmaß der Betriebskostenersparnis wird durch Zeile 1100 (beim Spectrum auch Zeile 1110) angezeigt. Zeile 1110 beinhaltet eine FOR . . . NEXT-Schleife, durch die das Verlassen des Unterprogramms ein wenig verzögert wird.

Spectrum

```
2000 PAPER 4: BORDER 4: INK 0: CLS
2030 LET r(m)=0: LET c(m,1)=INT
   (RND*90)+10: LET c(m,2)=INT
   ((RND*5)+2)*200: LET c(m,3)=INT
   (RND*200)+1: LET ll=INT (RND*3)-1
2050 LET c(m,4)=c(m,2)+ll*200
2070 LET c(m,5)=0: LET kk=INT (RND*
   100): IF kk<c(m,1) THEN LET c(m,5)=1
2080 PRINT PAPER 6; INK 0;AT 2,6;"□
   SCIENTIFIC REPORT□": PRINTAT 5,5;
   "Chance of gold=□";c(m,1);"%":
   PRINT AT 7,5;"Expected Depth=□";
   c(m,2);"m": PRINT AT 9,5;"Expected
   amount=□";c(m,3);"kg"
2100 LET z=INT (RND*150000): LET a(m,2)
   =a(m,2)-z: LET a(m,1)=a(m,1)-z
2110 PRINT FLASH 1;AT 12,0;"Would you like
   to mine? (y or n)"
2120 LET r$=INKEY$: IF r$="" THEN
   GOTO 2120
2130 IF r$="y" THEN LET a(m,6)=0: LET
   r(m)=1: GOTO 3000
2500 RETURN
```

Commodore 64

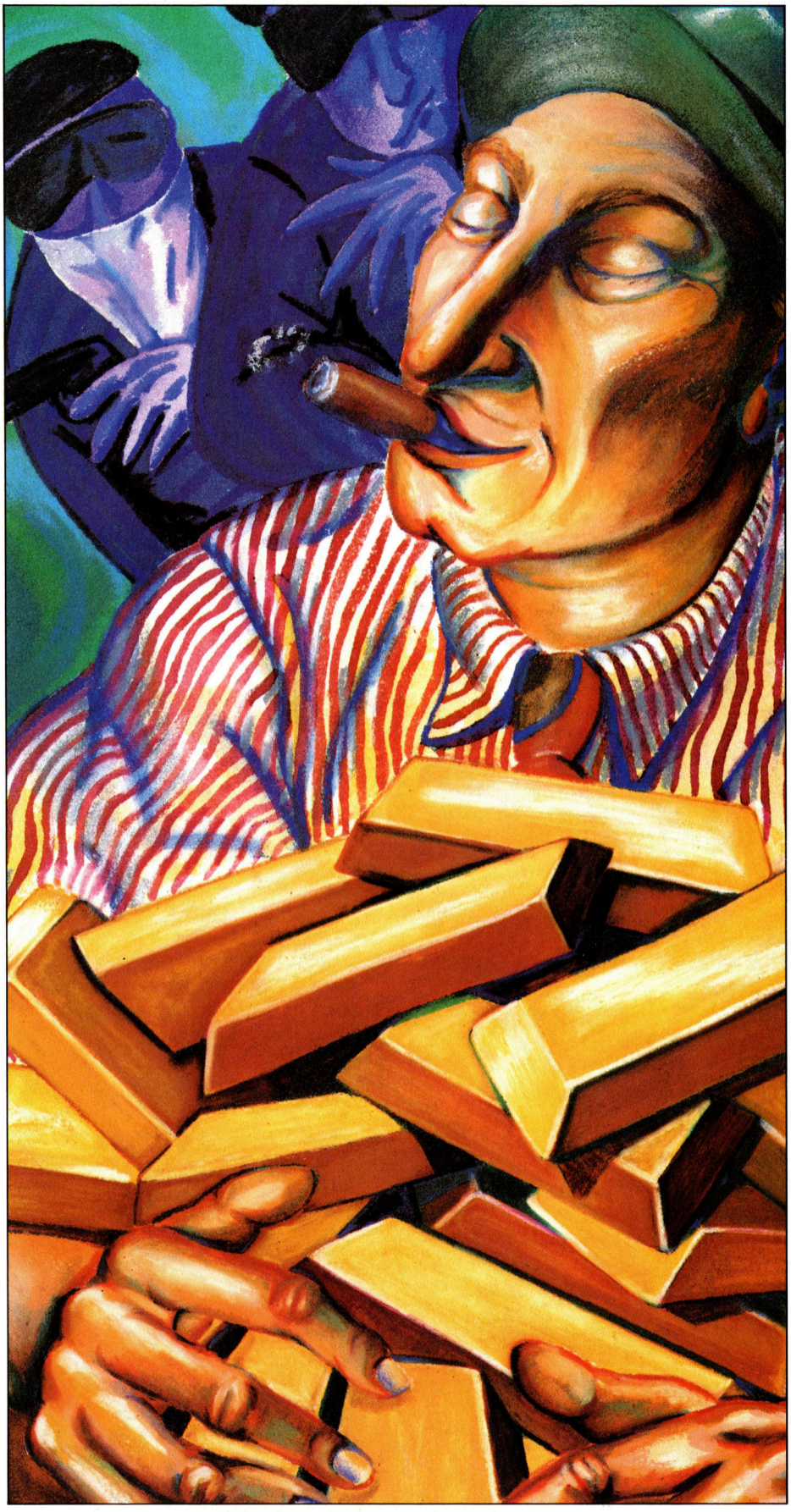
```
2000 POKE53280,5:POKE53281,5:
   PRINT"♥■";
2030 R(M)=0:C(M,1)=INT(RND(1)*90)+
   10:C(M,2)=INT((RND(1)*5)+2)*200
2031 C(M,3)=INT(RND(1)*200)+1:
   LL=INT(RND(1)*3)-1
2050 C(M,4)=C(M,2)+LL*200
2070 C(M,5)=0:KK=INT(RND(1)*
   100):IFKK<C(M,1)THENC(M,5)=1
2080 PRINT"■■■■■■■■■
   ■□SCIENTIFIC REPORT□"
2081 PRINT"■■■■■■■■
   CHANCE OF GOLD=";C(M,1)"%"
2082 PRINT"■■■■■■■■
   EXPECTED DEPTH=";C(M,2);"M"
2083 PRINT"■■■■■■■■
   EXPECTED AMOUNT";C(M,3) "KG"
2100 Z=INT(RND(1)*150000):A(M,2)
   =A(M,2)-Z:A(M,1)=A(M,1)-Z
2110 PRINT"■■■■WOULD YOU LIKE
   TO MINE (Y OR N)?"
2120 GETR$:IFR$<>"Y"ANDR$<>
   "N"THEN2120
2130 IFR$="Y"THENA(M,6)=0:R(M)
   =1:GOTO3000
2500 RETURN
```

Commodore VC 20

```
2000 PRINT"♥■"
2030 R(M)=0:C(M,1)=INT(RND(1)*90)
   +10:C(M,2)=INT((RND(1)*5)+2)*200
2031 C(M,3)=INT(RND(1)*200)+1:
   LL=INT(RND(1)*3)-1
2050 C(M,4)=C(M,2)+LL*200
2070 C(M,5)=0:KK=INT(RND(1)*
   100):IFKK<C(M,1)THENC(M,5)=1
2080 PRINT"■■■■SCIENTIFIC REPORT"
2081 PRINT"■■■CHANCE OF
   GOLD=";C(M,1)"%"
2082 PRINT"■■EXPECTED DEPTH
   =";C(M,2);"M"
2083 PRINT"■■EXPECTED AMOUNT";
   C(M,3)"KG"
2100 Z=INT(RND(1)*150000):A(M,2)
   =A(M,2)-Z:A(M,1)=A(M,1)-Z
2110 PRINT"■■■■WOULD YOU LIKE TO
   MINE■■■■■■■■(Y OR N)?"
2120 GETR$:IFR$<>"Y"ANDR$<>
   "N"THEN2120
2130 IFR$="Y"THENA(M,6)=0:R(M)
   =1:GOTO3000
2500 RETURN
```

Acorn B

```
2000 COLOUR129:COLOUR3:CLS
2030 R(M)=0:C(M,1)=RND(90)+9:
   C(M,2)=RND(5)*200+400:C(M,3)
```



```
    =RND(200):LL=RND(3)-2
2050 C(M,4)=C(M,2)+LL*200
2070 C(M,5)=0:KK=RND(100):IF
    KK<C(M,1) THEN C(M,5)=1
2080 PRINTTAB(10,3)"SCIENTIFIC
    REPORT"TAB(10,10)"CHANCE OF
    GOLD=□";C(M,1);"%"TAB(10,12)
    "EXPECTED DEPTH=□";C(M,2);
    "m"TAB(10,14)"EXPECTED
    AMOUNT=□";C(M,3);"KG"
2100 Z=RND(150000):A(M,2)=
    A(M,2)-Z:A(M,1)=A(M,1)-Z
2110 PRINTTAB(5,20)"WOULD YOU LIKE TO
    MINE (Y/N)?"
2120 R$=GET$
2130 IF R$="Y" THEN A(M,6)=0:
    R(M)=1:GOTO 3000
2500 RETURN
```

Tandy/Dragon

```
2000 CLS
2030 R(M)=0:C(M,0)=RND(90)+9:
    C(M,1)=(RND(5)+1)*200:C(M,2)
    =RND(200):LL=RND(3)-2
2050 C(M,3)=C(M,1)+LL*200
2070 C(M,4)=0:KK=RND(100)-1:
    IF KK<C(M,0) THEN C(M,4)=1
2080 PRINT@6,"scientific report":
    PRINT@129,"CHANCE OF GOLD=";
    C(M,0);"%":PRINT@193,
    "EXPECTED DEPTH =";C(M,1);
    "m":PRINT@257,"EXPECTED
    AMOUNT=";C(M,2);"kg"
2100 Z=RND(150000)-1:A(M,1)=
    A(M,1)-Z:A(M,0)=A(M,0)-Z
2110 PRINT@353,"WOULD YOU LIKE TO
    MINE (Y/N) ?"
2120 R$=INKEY$:IF R$<>"Y" AND
    R$<>"N" THEN 2120
2130 IF R$="Y" THEN A(M,5)=0:
    R(M)=1:GOTO3000
2500 RETURN
```

Bei allen Rechnern wird der Bildschirm durch Zeile 2000 gelöscht. Spectrum, Commodore und Acorn wechseln auch die Bildschirmfarbe. Zeile 2030 setzt R(M) (beim Spectrum r(m)) auf Null, um festzuhalten, daß noch nicht in der Mine gearbeitet wird. Diese Zeile erzeugt auch eine Vorentscheidung, ob, wieviel und in welcher Tiefe Gold gefunden wird. LL (oder ll) ist eine Zufallszahl zwischen -1 und 1, die in der nächsten Zeile für die Festsetzung der tatsächlichen Fundtiefe gebraucht wird; der Wert C(M,2) war ja nur eine vorläufige Tiefenangabe.

Zeile 2050 setzt C(M,4) gleich

C(M,2) plus/minus 200 Meter, also 200mal LL. Als nächstes entscheidet Zeile 2070, ob die Mine überhaupt Gold enthält. Bei C(M,5) gleich Null ist nichts zu finden. KK ist ein Zufallswert zwischen 0 und 99. KK wird mit der Wahrscheinlichkeit verglichen, Gold zu finden – ist KK kleiner, wird C(M,5) auf Eins gesetzt, um anzuzeigen, daß eine Goldader vorhanden ist.

Zeile 2080 konfrontiert den Spieler mit einem wissenschaftlichen Gutachten über die Mine; beim Commodore liegt dieser Teil in den Zeilen 2080 bis 2083. Obwohl dem Spieler gesagt wird, wo ein Fund wahrscheinlich ist, bleibt das endgültige Ergebnis dem Zufall überlassen. Der Spieler muß sich also entscheiden, ob er ein Risiko eingehen will oder nicht.

Jetzt aber zu den Kehrseiten: Das Gutachten kostet Geld. Der Preis ist noch unbestimmt; der Betrag kann zwischen 0 und 150 000 $ schwanken, also dem Wert von Z in Zeile 2100. Diese Kosten belasten natürlich die Bilanz erheblich.

Jetzt endlich kann der Grubenbesitzer mit den Grabungsarbeiten beginnen: Zeile 2110 fragt dazu „WOULD YOU LIKE TO MINE?" Bei einer positiven Antwort springt das Programm zum Unterprogramm für die Grabungsarbeiten in Zeile 3000.

Spectrum

```
3000 BORDER 6: PAPER 6: INK 1: CLS
3010 IF r(m)=Ø THEN PRINT FLASH 1;AT
  9,2;"You have not explored yet!": FOR z=1
  TO 1Ø: BEEP .3,-1Ø: NEXT z: RETURN
3020 BORDER 5: INK Ø: PAPER 4: CLS
3022 PRINT PAPER 5;TAB 14;CHR$ 147;CHR$
  148;CHR$ 149;TAB 14;CHR$ 150;CHR$
  151;CHR$ 152;CHR$ 153;TAB 13;CHR$
  154;CHR$ 155;CHR$ 156;CHR$ 157;CHR$
  158;TAB 31;CHR$ 32
3025 FOR z=1 TO 32: PRINT CHR$ 144;:
  NEXT z
3060 PRINT AT 4,Ø;: FOR z=1ØØ TO 14ØØ
  STEP 1ØØ: PRINT TAB 4-LEN STR$ z;z:
  NEXT z
3090 LET a(m,2)=a(m,2)-a(m,4): LET
  a(m,1)=a(m,1)-a(m,4): LET
  a(m,6)=a(m,6)+2ØØ: PAUSE 3Ø
3100 PRINT AT 3,15;CHR$ 146: FOR f=4 TO
  (a(m,6)/1ØØ)+3: PRINT AT f,15;CHR$
  145: FOR w=1 TO 1Ø: BEEP .Ø1,-2Ø:
  NEXT w: NEXT f
3120 IF a(m,6)=c(m,4) AND c(m,5)=1
  THEN GOTO 35ØØ
3130 PRINT FLASH 1; PAPER 5;AT 6,18;"No
  gold yet!": IF a(m,6)=c(m,2)+2ØØ THEN
  PRINT FLASH 1; PAPER 1; INK 6;AT
  18,Ø;"This mine doesn't contain any gold.
  Try starting another one.": FOR z=1 TO 1Ø:
  BEEP .5,-2Ø: NEXT z: LET a(m,6)=Ø:
  LET r(m)=Ø
3140 PAUSE 15Ø
3300 RETURN
3500 PRINT PAPER 6; INK 2; FLASH 1;AT
  f,12;"G O L D": FOR z=-2Ø TO 5Ø:
  BEEP .Ø17,z: NEXT z: PAUSE 75
3550 LET a(m,5)=a(m,5)+1: LET
  a(m,3)=a(m,3)+c(m,3): LET
  a(m,1)=a(m,1)+(a(m,3)*er):
  LET a(m,6)=Ø: LET r(m)=Ø: GOTO 33ØØ
```

Commodore 64

```
3000 POKE5328Ø,7:POKE53281,7
3010 IFR(M)<>ØTHEN3Ø2Ø
3011 PRINT"♥■■■■■■■■■
  ■■■■▣YOU HAVE NOT EXPLORED
  YET!":FORZ=1TO23ØØ:NEXT:RETURN
3020 POKE5328Ø,3:POKE53281,5:
  PRINT"♥■"
3022 PRINTTAB(14);"□□□":PRINT
  TAB(14);"□□□□":PRINTTAB
  (13);"□□□□□"
3025 FORZ=ØTO39:PRINT"♠";:NEXT
3060 PRINT"■■■■■■■";:
  FORZ=1ØØTO14ØØSTEP1ØØ:PRINT
  TAB(5-LEN(STR$(Z)));Z:NEXT
3090 A(M,2)=A(M,2)-A(M,4):A(M,1)=
  A(M,1)-A(M,4):A(M,6)=A(M,6)+2ØØ
3095 FORF=ØTO9Ø:NEXT
3100 PRINT"■■■■■■■■■■
  ■■■■■■■■■■■■■□"
3101 PRINT"○■■■■■■■■■
  ■■■■■■■■□"
3102 FORF=2TO A(M,6)/1ØØ:PRINT
  "■■■■■■■■■■■■■■
  ■■■■■□"
3104 POKE54272,33:POKE54273,33:
  POKE54277,15:POKE54296,15
3105 POKE54276,129:FORZ=1TO24Ø: NEXT
3110 NEXT:POKE54296,Ø
3120 IFA(M,6)=C(M,4)ANDC(M,5)
  =1THEN 35ØØ
3130 PRINT"■■■■■■■■
  ■■■■■■■■■■
  ■■■■■■■■■■■■
  NO GOLD YET!!!"
3131 IFA(M,6)<>C(M,2)+2ØØTHEN 314Ø
3132 PRINT"■■■THIS MINE DOESN'T
  CONTAIN ANY GOLD. TRY□□STARTING
  ANOTHER ONE."
3134 A(M,6)=Ø:R(M)=Ø
3140 FORF=1TO25ØØ:NEXT
3300 RETURN
3500 PRINT "■■■■":FORZ=1
  TOA(M,6)/1ØØ:PRINT:NEXT
3505 PRINT"■■■■■■■■■
  ■■■■□G□O□L□D□"
3510 FORF=54272TO54296:POKEF,Ø:NEXT
3520 POKE54284,15:POKE54283,17:
  POKE54296,14
3530 FORF=64TO124
3540 POKE5428Ø,F:FORG=1TO2Ø:
  NEXT:NEXT
3550 FORF=124TO64STEP-1:POKE
  5428Ø,F:FORG=1TO2Ø:NEXT:NEXT
3560 POKE54296,Ø
3570 A(M,5)=A(M,5)+1:A(M,3)=A(M,3)
  +C(M,3):A(M,1)=A(M,1)+A(M,3)*ER
3580 A(M,6)=Ø:R(M)=Ø:GOTO33ØØ
```

Commodore VC 20

```
3000 POKE 36879,25
3010 IFR(M)<>ØTHEN3Ø2Ø
3011 PRINT"♥■■■■■■■■■
  ▣YOU HAVE NOT EXPLORED YET!":
  FORZ=1TO23ØØ:NEXT:RETURN
3020 PRINT"♥■■"
3022 PRINTTAB(14);"■■■":
  PRINTTAB(14);"■■■□■■□"
3025 FORZZ=1TO15:FORZ=ØTO2Ø:
  PRINT"■□";:NEXT:PRINT:NEXT
3060 PRINT"■■■■■■";:FOR Z=
  1ØØTO14ØØSTEP1ØØ:PRINT"■"Z: NEXT
3090 A(M,2)=A(M,2)-A(M,4):A(M,1)
  =A(M,1)-A(M,4):A(M,6)=A(M,6)
  +2ØØ
3095 FORF=ØTO9Ø:NEXT
3100 PRINT"■■■■■"SPC(15)
  "■□"
3102 FORF=2TO A(M,6)/1ØØ:PRINT
  SPC(15)"■□":POKE 36877,128+F*3
3104 FORDE=5TO15STEP.3:POKE36878,
  DE:NEXT:POKE36877,Ø
3110 NEXT
3120 IFA(M,6)=C(M,4)ANDC(M,5)
  =1THEN 35ØØ
3130 PRINT"■■■■■■■
  ■■■■■■■■■■■■
  ■■■■■■■NO GOLD YET!!!"
3131 IFA(M,6)<>C(M,2)+2ØØTHEN314Ø
3132 PRINT"■■■THIS MINE
  DOESN'T□□□CONTAIN ANY GOLD.
  TRY□□STARTING ANOTHER ONE."
3134 A(M,6)=Ø:R(M)=Ø
3140 FORF=1TO3ØØØ:NEXT
3300 RETURN
3500 PRINT "■■■":FORZ=1TO
  A(M,6)/1ØØ:PRINT:NEXT
3505 PRINT"■■■■■■■■■
  ■■■■□G□O□L□D□"
3508 FORDE=25ØTO127STEP-1:POKE
```

```
   36876,DE:NEXT
3510 FORG=1TO2000:NEXT
3570 A(M,5)=A(M,5)+1:A(M,3)=A(M,3)
   +C(M,3):A(M,1)=A(M,1)+A(M,3)*ER
3580 A(M,6)=0:R(M)=0:GOTO3300
```

Acorn B

```
3000 COLOUR130:COLOUR0:CLS
3010 IF R(M)=0 THEN PRINTTAB(6,12)
   "YOU HAVE NOT EXPLORED YET!":FOR Z=
   1 TO 10:SOUND1,-15,100,1:SOUND1,0,
   1,1:NEXT:Z=INKEY(300):RETURN
3020 CLS
3022 VDU 31,16,3,224,225,226,31,16,4,227,
   228,229,230,31,15,5,231,232,233,234,235
3025 PRINT:FOR Z=1 TO 40:VDU236:NEXT
3060 PRINT:FOR Z=100 TO 1400 STEP
   100:PRINTTAB(4-LENSTR$Z);Z:NEXT
3090 A(M,2)=A(M,2)-A(M,4):A(M,1)
   =A(M,1)-A(M,4):A(M,6)=A(M,6)+
   200:Z=INKEY(60)
3100 VDU31,17,6,238:FOR F=7 TO
   (A(M,6)/100)+7:VDU31,17,F-1,237:
   FOR Z=1 TO 13: SOUND0,-15,6,1:
   SOUND0,0,0,1:NEXT:SOUND16,0,0,1:NEXT
3120 IF A(M,6)=C(M,4) AND C(M,5)
   =1 THEN 3500
3125 COLOUR1
3130 PRINTTAB(20,10)"NO GOLD YET!":IF
   A(M,6)=C(M,2)+200 THEN COLOUR3:
   PRINTTAB(0,29)"THIS MINE DOESN'T
   CONTAIN ANY GOLD. TRY STARTING
   ANOTHER ONE.":A(M,6)=0:R(M)=0
3140 FOR Z=1 TO 4000:NEXT
3300 RETURN
3500 COLOUR1:PRINTTAB(14,F)"G□O□L
   □D":FOR Z=0 TO 250 STEP 10:
   SOUND1,-15,Z,1:NEXT:Z=INKEY (150)
3550 A(M,5)=A(M,5)+1:A(M,3)=
   A(M,3)+C(M,3):A(M,1)=A(M,1)+(A(M,
   3)*ER):A(M,6)=0:R(M)=0:GOTO 3300
```

Tandy/Dragon

```
3000 CLS
3010 IF R(M)=0 THEN PRINT@66,"YOU
HAVE NOT EXPLORED YET !";:FORZ=1
   TO10:SOUND120,1:NEXT:RETURN
3015 IF A(M,5)>0 THEN LINE(140,40)
   -(157,191),PRESET,BF:GOTO3090
3020 PCLS:SCREEN1,0:COLOR3:LINE
   (0,0)-(255,31),PSET,BF
3022 PUT(131,8)-(168,31),H,PSET
3025 FORZ=0TO 31: PUT(Z*8,32)-
   (Z*8+7,34),T,PSET:NEXT
3060 FORZ=100 TO 1400 STEP 100:
   Z$=STR$(Z)+"-":DRAW"C4S8BM"
   +STR$(49-8*LEN(Z$))+","+STR$
   (32+10*Z/100):GOSUB9000:NEXT
3090 SCREEN1,0:A(M,1)=A(M,1)-
   A(M,3):A(M,0)=A(M,0)-A(M,4):
   A(M,5)=A(M,5)+200
3100 PUT(145,32)-(152,39),D,PSET:
   FORF=4TO(A(M,5)/100)+3:PUT(145,
   F*10)-(152,F*10+9),B,PSET:
   PLAY"T50O1BDBDEBDBDE":NEXT
3120 IF A(M,5)=C(M,3) AND C(M,4)
   =1 THEN 3500
3125 FORZ=1TO1000:NEXT
3130 PRINT@40," NO GOLD YET ! ";:
   IF A(M,5)=C(M,1)+200 THEN
   PRINT@128," THIS MINE DOESN'T
   CONTAIN ANY□□□GOLD.□TRY
   STARTING ANOTHER ONE.□";:PLAY
   "T50O3CDEFG":A(M,5)=0: R(M)=0
3140 FORZ=1TO2500:NEXT
3300 RETURN
3500 F=40+A(M,5)/10:COLOR2:LINE
   (140,F)-(157,F+5),PSET,BF
3510 FORZ=1TO10:PLAY"T5O2CA":PUT
   (140,F)-(157,F+5),H,NOT:NEXT
3520 FORZ=1TO2000:NEXT
3550 A(M,4)=A(M,4)+1:A(M,2)=A(M,2)
   +C(M,2):A(M,0)=A(M,0)+(A(M,2)
   *ER):A(M,5)=0:R(M)=0:GOTO3300
```

Dieses Unterprogramm kann von zwei verschiedenen Stellen im Ablauf angesprochen werden, entweder aus dem Unterprogramm Erkundung und Gutachten oder bei Entscheidung zugunsten einer weiteren Vertiefung des Minenschachtes um 200 Meter (Option 3).

Wie gewöhnlich löscht die erste Zeile zunächst den Bildschirm bzw. ändert die Farben. Zeile 3010 prüft, ob die Erkundungsphase abgeschlossen ist, bevor mit den Arbeiten begonnen wird. Bei den Commodore-Rechnern dienen die Zeilen 3010 und 3011 diesem Zweck und erzeugen auch entsprechende Meldungen am Bildschirm. Zeile 3020 bereitet den Bildschirm wieder für die Anzeige vor.

Zeile 3022 bis 3090 erzeugen die Grafik der Goldmine. Zeile 3100 illustriert die Grabungsarbeiten und sorgt für Soundeffekte. Bei den Commodore-Programmen werden hierfür Zeile 3100 bis 3110 genutzt.

Zeile 3120 prüft, ob der Schacht bereits auf dem Niveau des Goldes angekommen ist und ob überhaupt Gold vorhanden ist (es ist auch möglich, die richtige Tiefe zu erreichen, ohne daß tatsächlich Gold an dieser Stelle liegt.) Wenn Sie fündig geworden sind, verzweigt das Programm zu Zeile 3500, die den Erfolg meldet und ein Lied spielt; beim Commodore erstreckt sich dieser Programmteil wiederum über mehrere Zeilen. Zeile 3550 bringt die Bilanz des Spielers immer auf den neuesten Stand.

Falls nichts gefunden wurde, läuft das Programm mit Zeile 3130 weiter. Wenn bereits über den erwarteten Fundort hinaus gegraben wurde, erfährt der Spieler, daß dort nichts zu finden ist. Vor diesem Punkt wird nur angezeigt „NO GOLD YET!" – „Bis jetzt noch kein Gold!"

Spectrum

```
4000 PAPER 6: INK 1: BORDER 6: CLS
4020 PRINT INVERSE 1;AT 2,7;"□EXCHANGE
   AGENCY□":PRINT AT 6,0;"The current
   exchange rate is:—";AT 8,5;"1 kg of
   gold=□$";er;AT 12,2;"Enter no. of kg to
   exchange": INPUT nte
4070 IF nte>a(m,3) THEN PRINT FLASH 1;AT
   16,0;"You do not have that much gold!"
4080 LET nte=INT nte
4090 IF nte>a(m,3) OR nte<0 THEN GOTO
   4020
4095 PRINT AT 16,0;CHR$ 32;TAB 31;CHR$ 32
4100 LET a(m,3)=a(m,3)-nte: LET
   a(m,2)=a(m,2)+(nte*er): LET
   a(m,1)=a(m,1)+(nte*er)
4130 PRINT PAPER 5;AT 16,1;nte;"kg
   exchanged for□$";nte*er: PAUSE 170:
   RETURN
5000 RETURN
```

Commodore 64

```
4000 POKE53280,7:POKE53281,7:
   PRINT"♥■"
4020 PRINT"□□□□□□□■■■
   □EXCHANGE AGENCY□"
4030 PRINT"■■■■THE CURRENT
   EXCHANGE RATE IS:—"
4040 PRINT:PRINT"■■■1 KG OF
   GOLD=$"ER
4050 PRINT:PRINT:PRINT"ENTER NO. OF KG
   TO EXCHANGE":INPUT NT
4060 IFNT>A(M,3)THENPRINT"■YOU DO
   NOT HAVE THAT AMOUNT OF GOLD!!!!!"
4070 NT=INT(NT)
4090 IFNTE>A(M,3)ORNTE<0THEN 4020
4100 A(M,3)=A(M,3)-NT:A(M,2)=A(M,2)
   +(NT*ER):A(M,1)=A(M,1)+(NT*ER)
4130 PRINT"▒"NT"KG EXCHANGED FOR $
   "NT*ER:FORF=1TO2000:NEXT:RETURN
```

Commodore VC 20

```
4000 PRINT"♥■"
4020 PRINT"■■■□□□EXCHANGE
   AGENCY□□□□"
```

```
4030 PRINT"■■THE CURRENT
   EXCHANGE□□ RATE IS:—"
4040 PRINT:PRINT"1 KG=$"ER
4050 PRINT:PRINT"■ENTER NO.OF KG
   TO□□□□EXCHANGE":INPUTNT
4060 IFNT>A(M,3)THENPRINT"■YOU DO
   NOT HAVE THAT AMOUNT OF
   GOLD□!□!□!□□"
4070 NT=INT(NT)
4090 IFNTE>A(M,3)ORNTE<0THEN4020
4100 A(M,3)=A(M,3)-NT:A(M,2)=A(M,2)
   +(NT*ER):A(M,1)=A(M,1)+(NT*ER)
4130 PRINT"▓"NT"KG EXCHANGED FOR":
   PRINT"▌$"NT*ER:FORF=1TO2000:
   NEXT:RETURN
```

Acorn B

```
4000 CLS
4020 PRINTTAB(12,3)"EXCHANGE
   AGENCY"TAB(5,10) "THE CURRENT
   EXCHANGE RATE IS:—"TAB(5,12) "1 kg
   OF GOLD□=□$";ER;:INPUT ""ENTER
   NO. OF kg TO EXCHANGE",NTE
4070 NTE=INT(NTE)
4080 IF NTE>A(M,3) THEN PRINT""YOU
   DON'T HAVE THAT MUCH GOLD!"
4090 IF NTE>A(M,3)OR NTE<0 THEN
   PRINTTAB(28,14)SPC(10):GOTO 4020
4095 VDU11:PRINTSPC(39)
4100 A(M,3)=A(M,3)-NTE:A(M,2)=A(M,2)
   +(NTE*ER):A(M,1)=A(M,1)+(NTE*ER)
4130 PRINT";NTE"kg EXCHANGED□FOR□$";
   NTE*ER;SPC(20):Z=INKEY(340): RETURN
5000 RETURN
```

Tandy/Dragon

```
4000 CLS
4020 PRINT@7,"exchange agency":
   PRINT@128,"THE CURRENT EXCHANGE
   RATE IS:—":PRINT@197,"1KG OF
   GOLD=";ER:PRINT@288,"ENTER NO. OF
   KG TO EXCHANGE";:INPUT NT
4080 NT=INT(NT)
4090 IF NT>A(M,2) OR NT<0 THEN 4020
4100 A(M,2)=A(M,2)-NT:A(M,1)=A(M,1)
   +(NT*ER):A(M,0)=A(M,0)+(NT*ER)
4130 PRINT@448,NT;"KG EXCHANGED FOR";
   NT*ER:FORZ=1TO1000:NEXT:RETURN
5000 RETURN
```

Zeile 4000 initialisiert dieses Programm.

Zeile 4020 (Commodore 4020 bis 4050) erzeugt die Überschrift, den aktuellen Goldkurs und fragt danach, wieviele Kilo Gold verkauft werden sollen. Außer im Tandy/Dragon-Programm prüft Zeile 4070, ob genügend Gold für den Verkauf vorrätig ist. Zeile 4080 sorgt dafür, daß die zu verkaufende Goldmenge ein ganzzahliges Gewicht hat.

Mit Zeile 4090 gelangt der Ablauf zurück zu der Frage, ob das zu tauschende Gold überhaupt vorhanden ist. Zeile 4100 bringt die Bilanz des Spielers nach dem Verkauf wieder ins Lot.

Das Unterprogramm meldet dem Spieler durch Zeile 4130, wieviel Gold in Geld umgewechselt wurde.

Die Zeilen 7000 bis 7030 stellen die Aufforderung zu einer neuen Spielrunde dar.

Spectrum

```
1 FOR n=USR"a" TO USR "o"+7:
   READ a: POKE n,a: NEXT n
7000 PAPER 5: INK 0: BORDER 5: CLS
7010 PRINT AT 9,12;a$(m): PRINT AT
```

```
10,8;"has gone bust!": PRINT FLASH 1;AT
20,1; "□Press any key to play again□"
7030 PAUSE 0: RUN 5
8000 DATA 255,85,170,0,0,0,0,0,62,28,
   56,126,28,62,120,28
8010 DATA 255,255,62,126,127,60,124,
   126,0,0,0,0,1,1,1,1
8020 DATA 7,29,49,45,255,255,91,126,
   128,96,48,80,152,140,252,138
8030 DATA 1,1,1,49,49,49,49,255,122,
   187,62,95,153,255,153,126
8040 DATA 209,177,224,128,128,128,128,
   128,0,0,128,128,64,32,32,16
8050 DATA 1,3,7,7,4,4,7,7,255,255,255,
   255,149,149,159,159
8060 DATA 24,126,153,255,126,153,126,
   219,128,192,224,240,248,168,248,255
8070 DATA 16,8,8,4,14,31,31,255
```

Commodore 64

```
7000 POKE53280,3:POKE53281,3:
   PRINT"♥■"
7010 PRINT"■■■■■■■■
   ■■"TAB(16);A$(M):PRINTTAB
   (12)"■■HAS GONE BUST!!!"
7015 FORZ=1TO1000:NEXT
7020 PRINTTAB(6);"■■■PRESS ANY
   KEY TO PLAY AGAIN"
7030 POKE198,0:WAIT198,1:RUN5
60000 POKE56334,0:POKE1,35
60010 FORF=0TO2047:POKE12288+
   F,PEEK(53248+F):NEXT
60020 POKE1,39:POKE56334,1
60030 SP=12808
60040 READA:IFA=-1THEN60070
60050 POKESP,A
60060 SP=SP+1:GOTO60040
60070 RETURN
60080 DATA255,85,170,0,0,0,0,0,
   62,28,56,126,28,62,120,28
60090 DATA255,255,62,126,127,60,
   124,126,0,0,0,0,1,1,1,1
60100 DATA7,29,49,45,255,255,91,
   126,128,96,48,80,152,140,252,138
60110 DATA1,1,1,49,49,49,49,255,
   122,187,62,95,153,255,153,126
60120 DATA209,177,224,128,128,128,
   128,128,0,0,128,128,64,32,32,16
60130 DATA1,3,7,7,4,4,7,7,255,255,
   255,255,149,149,159,159
60140 DATA24,126,153,255,126,153,
   126,219,128,192,224,240,248,168,
   248,255
60150 DATA16,8,8,4,14,31,31,255,-1
```

Commodore VC 20

```
7000 PRINT"♥■"
7010 PRINT"■■■■■■■■
   ■■"TAB(6);A$(M):PRINT"■■
   ■■HAS GONE BUST!!!"
7015 FOR Z=1TO1000:NEXT
7020 PRINT"■■■■PRESS ANY KEY TO
   PLAY"
7030 POKE198,0:WAIT198,1:RUN
```

Acorn B

```
7000 CLS
7010 PRINTTAB(10,10)A$(M)"□HAS
   GONE BUST"TAB(0,29)"PRESS ANY
   KEY TO PLAY AGAIN"
7030 Z=GET:RUN
8000 DATA 0,0,0,0,1,1,1,1
8010 DATA 7,29,49,45,255,255,91,126,
   128,96,48,80,152,140,252,138
8020 DATA 1,1,1,49,49,49,49,255,122,
   187,62,95,153,255,153,126
8030 DATA 209,177,224,128,128,128,128,
   128,0,0,128,128,64,32,32,16
8040 DATA 1,3,7,7,4,4,7,7,255,255,255,
   255,149,149,159,159
8050 DATA 24,126,153,255,126,153,126,
   219,128,192,224,240,248,168,248,255
8060 DATA 16,8,8,4,14,31,31,255,255,85,
   170,0,0,0,0,0
8070 DATA 62,28,56,126,28,62,120,28,
   255,255,62,126,127,60,124,126
```

Tandy/Dragon

```
7000 CLS
7010 PRINT@76,A$(M):PRINT@168,"HAS
   GONE BUST !":PRINT@449,"PRESS ANY
   KEY TO PLAY AGAIN"
7020 IF INKEY$=""" THEN 7020 ELSE RUN
9000 FOR K=1 TO LEN(Z$)
9010 B$=MID$(Z$,K,1)
9020 IF B$>="0" AND B$<="9" THEN
   DRAW NU$(VAL(B$)):GOTO 9050
9030 IF B$="-" THEN DRAW"BF2R4"
9050 NEXT
9060 RETURN
```

Bei den Spectrum- und Acorn-Versionen enthalten die Zeilen 8000 bis 8070 die DATAs für die UDGs. Im Commodore-64-Programm werden die DATA-Informationen von den Zeilen 60000 bis 60150 gePOKEt. Die Acorn-Version hat die UDG-DATA in den Zeilen 8000 bis 8070. Die Zeilen 9000 bis 9060 dienen beim Tandy/Dragon-Programm dazu, die Schacht-Tiefe links auf dem Bildschirm darzustellen.

Also los – stellen Sie Ihr Glück auf die Probe!

Neue Formeln

Nachdem wir die Grundfunktionen von Daten und Formeln in das Arbeitsblatt eingegeben haben, zeigen wir Ihnen in dieser Folge ein nützliches Hilfsmittel für Finanzpläne.

Die in die Zellen eingegebenen Formeln für das Arbeitsblatt sind sich sehr ähnlich. Möchten Sie zum Beispiel alle Zellen der ersten Spalte mit den dazugehörigen Zellen der zweiten Spalte addieren und das Resultat in dem dritten Element der Reihe ablegen, so erfordern diese Berechnungen je eine Formel in der dritten Spalte aller Reihen. Die erste Zelle dieser Spalte, A3, müßte mit der Formel A1+A2 programmiert werden, die folgende, B3, mit B1+B2 und so weiter. Diese Formeln sind zwar ähnlich, doch nicht identisch. Um die Fleißarbeit des Eintippens zu sparen, ist eine Funktion hilfreich, die die Urformel entsprechend verändert und in die gewünschten Zellen einsetzt. Sie heißt REPLICATE-Funktion.

Die Funktionen CLEAR, STORE und RETRIEVE beeinflussen das gesamte Arbeitsblatt. Wie die Namen andeuten, ermöglichen sie das Löschen der Daten aller Zellen, das Zwischenspeichern des Arbeitsblattes im Speicher (anstelle auf Band oder Diskette) und das Zurückrufen dieser Informationen.

Eine weitere nützliche Funktion ist die TAB-Funktion, die den Cursor auf dem Arbeitsblatt bewegt. Mit den Cursor-Steuertasten können alle Zellen angesprochen werden, doch muß das Programm nach einer Bewegung den Bildschirm neu aufbauen, und das nimmt natürlich Zeit in Anspruch. Mit der TAB-Funktion spricht der Anwender eine bestimmte Zelle direkt an, einfach durch Eingabe des Namens.

Alles Routine

Die Unterprogramme für die REPLICATE-Funktion liegen in den Zeilen 5400 bis 5995. Zunächst ruft die Hauptroutine ein Subprogramm auf, das die Formel aus dem Zeichenfeld F$() nach C$ kopiert und in ihre einzelnen Elemente zerlegt. Diese Elemente (Operatoren, Zellnamen und Konstanten) werden im Feld E$() abgelegt. Die Routine arbeitet C$ Zeichen für Zeichen ab und überträgt diese in den Zwischenspeicher T$. Bei Auffinden eines Operators wird der Inhalt von T$ in das nächste freie Element von E$() gelegt. Der gefundene Operator wird dann ebenfalls in E$() gespeichert. Dieser Vorgang wiederholt sich, bis die ganze Formel in C$ verarbeitet ist.

In den Zeilen 5500 bis 5650 liegt die Eingabe- und Prüfroutine, die es dem Anwender ermöglicht, eine Formel über eine bestimmte Gruppe von Zellen zu reproduzieren, und Kommandos im Format A1(B1–F1) annimmt. Dieses Kommando bedeutet: Nimm den Ausdruck in A1 und reproduziere ihn in die Zellen der Spalte 1 von B1 bis F1. Zum Reproduzieren einer Formel über den Bereich einer Reihe müßte die Eingabe A1(A2–A10) lauten. Der erste Teil der Eingaberoutine (zwischen 5500 bis 5520) prüft die Eingabe auf ihre richtige Schreibweise. Anschließend wird die Eingabe in die drei zur Ausführung benötigten Zellnamen zerlegt und in R1$, R2$ und R3$ abgelegt.

Harte Arbeit

Die eigentliche Replicate-Routine beginnt bei Zeile 5700 und entscheidet zunächst anhand der eingegebenen Zellnamen, ob die Formel über eine Reihe oder eine Spalte vervielfältigt werden soll. Wenn die drei Zellnamen mit dem gleichen Buchstaben beginnen (wie bei A1(A2–A10)), soll die Formel über Zellen einer Reihe reproduziert werden, im Beispiel über die Reihe A. Falls die Nummern der Zellnamen gleich sind, erfolgt die Reproduktion über eine Spalte. Falsche Eingaben wie A1(B1–F2) werden erkannt und nicht ausgeführt.

In den Zeilen 5800 bis 5895 finden wir die Routine, die Formeln über eine Spalte vervielfältigt. Nach dem oben beschriebenen Zerlegen der Eingabe benutzt eine FOR . . . NEXT-Schleife die ASCII-Werte der ersten Zeichen in R2$ und R3$ als Kontrollvariablen. Der Vorgang der Konstruktion einer neuen Formel für alle Zellen beginnt bei 5830.

Es ist notwendig, die Beschränkungen der REPLICATE-Funktion zu besprechen. Wenn Sie eine Formel über eine Spalte reproduzieren, erstellt die Routine neue Formeln nur dann erfolgreich, wenn die Zelladressen in der Formel innerhalb der gleichen Spalte liegen. Also wird die Formel A1+A2*A3 fehlerfrei umgeschrieben, A1+A2*B2 jedoch nicht.

Diese kurzen Unterprogramme befinden sich alle im Bereich der Zeilen 5000 bis 5395. Das Arbeitsblatt löschen ist ein simpler Vorgang. Die Elemente der Feldvariablen M(,), die alle Zelldaten enthält, werden mit verschachtelten FOR . . . NEXT-Schleifen auf Null gesetzt und das Arbeitsblatt durch Aufruf des Unterprogramms bei 1700 auf dem Bildschirm neu ausgegeben.

Für das zwischenzeitliche Abspeichern der eingegebenen Werte dient eine zweite Feldvariable, N(,). Die STORE- und RETRIEVE-Rou-

tinen schreiben einfach den Inhalt von M(,) in N(,) oder umgekehrt.

Das TAB-Unterprogramm beginnt bei der Zeile 5200 und akzeptiert als Eingabe eine Zelladresse. Vor dem Darstellen des Arbeitsblattes auf dem Bildschirm sind die horizontalen und vertikalen Grenzen des Bildschirmfensters neu zu berechnen.

Auch müssen die Zeilen- und Spaltenadressen neu ausgegeben werden, falls die angewählte Zelle außerhalb des bisherigen Fensterausschnittes liegt.

In der nächsten und letzten Folge dieses Projektes beschreiben wir Methoden zum Sichern und Laden der Arbeitsblattdaten auf Diskette oder Cassette.

BASIC-Dialekte

Acorn B
Ändern Sie folgende Zeilen in dem Commodore-Programm. Die PRINT-Befehle in den Zeilen 5220 und 5502 enthalten Leerfelder, um eine ganze Bildschirmzeile zu löschen.

```
5210 PRINT TAB (0,22);
5220 PRINT "NEW CELL:      "
5501 PRINT TAB (0,22);
550 PRINT "REPLICATE:      "
```

Schneider CPC
Ändern Sie folgende Zeilen in dem Commodore-Programm. Die PRINT-Befehle in den Zeilen 5220 und 5502 müssen ausreichend Leerfelder enthalten.

```
5210 LOCATE 1,22
5220 PRINT "NEW CELL:      "
5501 LOCATE 1,22
5502 PRINT "REPLICATE:     "
```

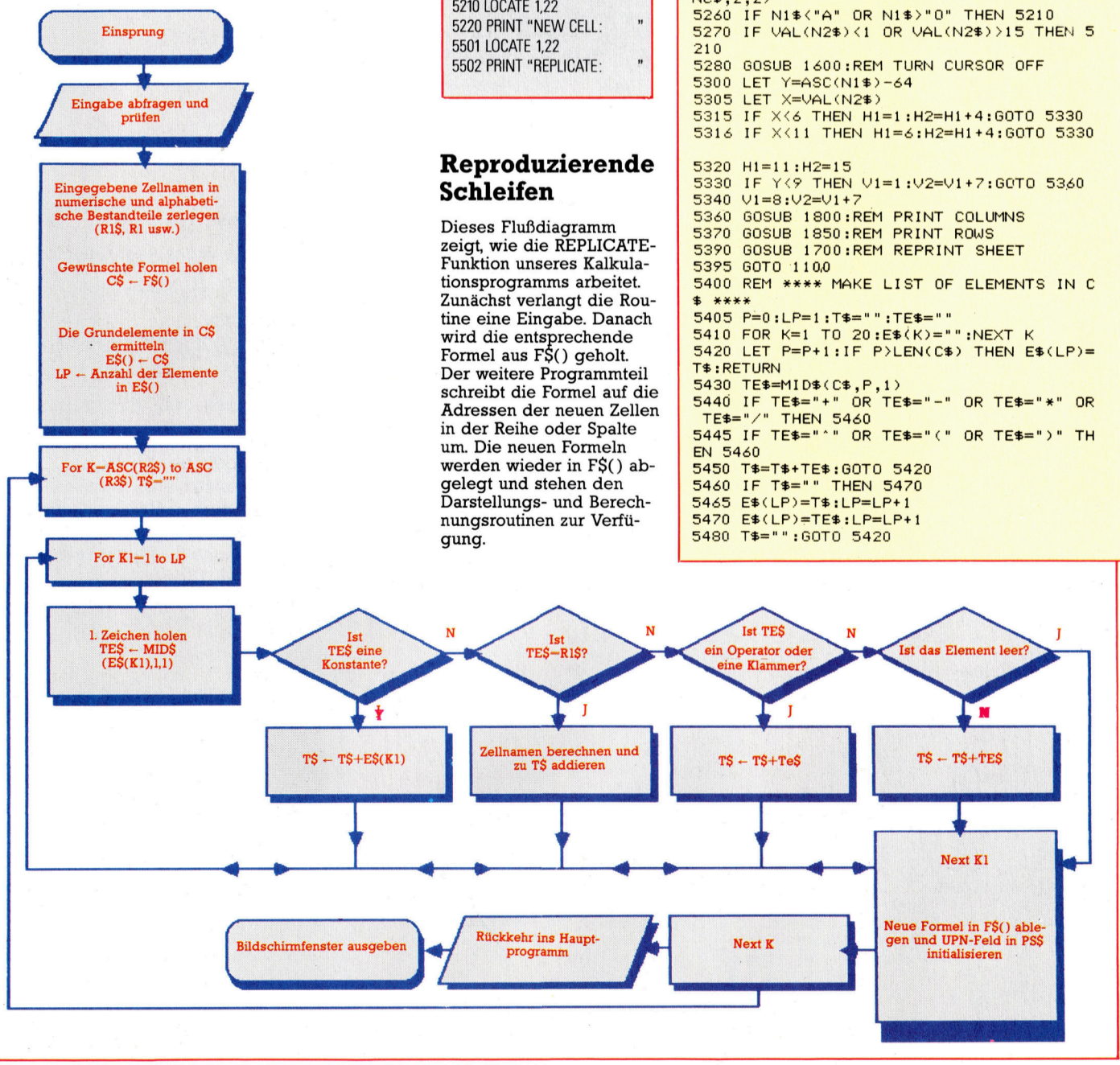
Zusätzliche Funktionen

Commodore 64

```
2320>LET P$=H$((J-1)*15+I,1TO ):
LET I$=F$((J-1)*15+I,1TO )
5000 REM **** CLEAR ARRAY ****
5010 FOR I=1 TO 15:FOR J=1 TO 15:M(I,J)=
0:NEXT J,I:GOSUB 1700:RETURN
5100 REM **** GET PREVIOUS SHEET ****
5110 FOR I=1 TO 15:FOR J=1 TO 15
5120 M(I,J)=N(I,J):NEXT J,I
5130 GOSUB 1700:RETURN:REM PRINT DATA
5150 REM ***** STORE CURRENT SHEET ****
5160 FOR I=1 TO 15:FOR J=1 TO 15
5170 N(I,J)=M(I,J):NEXT J,I
5180 GOSUB 1700:RETURN:REM PRINT DATA
5200 REM **** GOTO CELL ROUTINE ****
5210 GOSUB 1950:REM MOVE TO INPUT LINE
5220 PRINT "NEW CELL:
      ";CU$
5230 INPUT "NEW CELL:";NC$
5240 LET NC$=MID$(NC$,1,3)
5250 LET N1$=MID$(NC$,1,1):LET N2$=MID$(
NC$,2,2)
5260 IF N1$<"A" OR N1$>"O" THEN 5210
5270 IF VAL(N2$)<1 OR VAL(N2$)>15 THEN 5
210
5280 GOSUB 1600:REM TURN CURSOR OFF
5300 LET Y=ASC(N1$)-64
5305 LET X=VAL(N2$)
5315 IF X<6 THEN H1=1:H2=H1+4:GOTO 5330
5316 IF X<11 THEN H1=6:H2=H1+4:GOTO 5330

5320 H1=11:H2=15
5330 IF Y<9 THEN V1=1:V2=V1+7:GOTO 5360
5340 V1=8:V2=V1+7
5360 GOSUB 1800:REM PRINT COLUMNS
5370 GOSUB 1850:REM PRINT ROWS
5390 GOSUB 1700:REM REPRINT SHEET
5395 GOTO 1100
5400 REM **** MAKE LIST OF ELEMENTS IN C
$ ****
5405 P=0:LP=1:T$="":TE$=""
5410 FOR K=1 TO 20:E$(K)="":NEXT K
5420 LET P=P+1:IF P>LEN(C$) THEN E$(LP)=
T$:RETURN
5430 TE$=MID$(C$,P,1)
5440 IF TE$="+" OR TE$="-" OR TE$="*" OR
 TE$="/" THEN 5460
5445 IF TE$="^" OR TE$="(" OR TE$=")" TH
EN 5460
5450 T$=T$+TE$:GOTO 5420
5460 IF T$="" THEN 5470
5465 E$(LP)=T$:LP=LP+1
5470 E$(LP)=TE$:LP=LP+1
5480 T$="":GOTO 5420
```

Reproduzierende Schleifen

Dieses Flußdiagramm zeigt, wie die REPLICATE-Funktion unseres Kalkulationsprogramms arbeitet. Zunächst verlangt die Routine eine Eingabe. Danach wird die entsprechende Formel aus F$() geholt. Der weitere Programmteil schreibt die Formel auf die Adressen der neuen Zellen in der Reihe oder Spalte um. Die neuen Formeln werden wieder in F$() abgelegt und stehen den Darstellungs- und Berechnungsroutinen zur Verfügung.



```
5500 REM ***  REPLICATING FORMULA *****
5501 GOSUB 1950
5502 PRINT "REPLICATE:
       ";CU$
5503 INPUT "REPLICATE:";R$
5505 LET P=0:R1$="":R2$="":R3$="":T$=""
5510 IF MID$(R$,3,1)<>"(" AND MID$(R$,4,
1)<>"(" THEN 5500
5515 IF MID$(R$,6,1)<>"-" AND MID$(R$,7,
1)<>"-" AND MID$(R$,8,1)<>"-" THEN 5500
5517 IF R$="" THEN 5500
5520 LET P=P+1
5530 LET T$=MID$(R$,P,1)
5540 IF T$="(" THEN P=P+1:GOTO 5570
5550 LET R1$=R1$+T$
5560 GOTO 5520
5570 LET T$=MID$(R$,P,1)
5580 IF T$="-" THEN P=P+1:GOTO 5610
5590 LET R2$=R2$+T$
5600 LET P=P+1:GOTO 5570
5610 LET T$=MID$(R$,P,1)
5620 LET R3$=R3$+T$
5630 LET P=P+1
5640 IF P<=LEN(R$) THEN 5610
5650 RETURN
5700 REM **  DECIDE COLUMN OR ROW ***
5730 GOSUB 5500:REM TEST FOR VALID INPUT
5765 R1=VAL(MID$(R1$,2)):R1$=MID$(R1$,1,1)
5770 R2=VAL(MID$(R2$,2)):R2$=MID$(R2$,1,1)
5775 R3=VAL(MID$(R3$,2)):R3$=MID$(R3$,1,1)
5780 IF R1=R2 AND R1=R3 THEN 5800
5790 IF MID$(R1$,1,1)=MID$(R2$,1,1) AND
MID$(R1$,1,1)=MID$(R3$,1,1) THEN 5900
5795 GOTO 5700
5800 REM **** REPLICATE COLUMN ****
5805 LET C$=F$((ASC(R1$)-65)*15+R1)
5810 GOSUB 5400:REM MAKE LIST OF ELEMENTS
5820 FOR K=ASC(R2$) TO ASC(R3$):T$=""
5830 FOR K1=1 TO LP
5840 LET TE$=MID$(E$(K1),1,1)
5845 IF (TE$)="0" AND TE$(="9") OR TE$="
." THEN T$=T$+E$(K1):GOTO 5880
5850 IF TE$=R1$ THEN T$=T$+CHR$(K)+MID$(
E$(K1),2):GOTO 5880
5860 IF TE$="+" OR TE$="-" OR TE$="*" TH
EN T$=T$+TE$:GOTO 5880
5865 IF TE$="/" OR TE$="^" OR TE$="(" OR
 TE$=")" THEN T$=T$+TE$:GOTO 5880
5870 IF TE$<>""THEN T$=T$+TE$
5880 NEXT K1
5890 F$((K-65)*15+R1)=T$:PS$((K-65)*15+R
1)=""
5895 NEXT K:GOSUB 1900:RETURN
5900 REM **** REPLICATE ROW ****
5905 LET C$=F$((ASC(R1$)-65)*15+R1)
5910 GOSUB 5400:REM MAKE LIST OF ELEMENTS
5920 FOR K=R2 TO R3:T$=""
5930 FOR K1=1 TO LP
5940 LET TE$=MID$(E$(K1),1,1)
5945 IF (TE$)="0" AND TE$(="9") OR TE$="
." THEN T$=T$+E$(K1):GOTO 5980
5950 IF TE$)="A" AND TE$(="O" THEN T$=T$
+TE$+MID$(STR$(K),2):GOTO 5980
5960 IF TE$="+" OR TE$="-" OR TE$="*" TH
EN T$=T$+TE$:GOTO 5980
5965 IF TE$="/" OR TE$="^" OR TE$="(" OR
 TE$=")" THEN T$=T$+TE$:GOTO 5980
5970 IF TE$<>""THEN T$=T$+TE$
5980 NEXT K1
5990 F$((ASC(R1$)-65)*15+K)=T$:PS$((ASC(
R1$)-65)*15+K)=""
5995 NEXT K:GOSUB 1900:RETURN
```

Spectrum:

```
5000>REM *** CLEAR ARRAY ****
5010 DIM M(15,15): GO SUB 1700:
RETURN
5100 REM *** GET PREVIOUS SHEET****
5110 FOR I=1 TO 15: FOR J=1 TO 15
5120 LET M(I,J)=N(I,J)
5130 NEXT J: NEXT I
5140 GO SUB 1700: RETURN
5150 REM STORE CURRENT SHEET IN
MEMORY
5160 FOR I=1 TO 15: FOR J=1 TO 15
5170 LET N(I,J)=M(I,J): NEXT J:
NEXT I
5180 GO SUB 1700: RETURN
5200 REM **** GOTO CELL ROUTINE ****
5210 PRINT AT 18,0;"       ENTER
NEW CELL                "
5220 INPUT  LINE N$
5225 IF LEN (N$)<3 THEN  LET N$=
N$+" "
5230 LET N$=N$(1 TO 3): LET O$=N
$(2 TO 3)
5240 LET N$=N$(1)
5250 IF N$<"A" OR N$>"O" THEN  G
O TO 5210
5260 IF VAL (O$)<1 OR VAL (O$)>1
5 THEN  GO TO 5210
5280 GO SUB 1600: REM TURN CURSO
R OFF
5300 LET Y=CODE (N$)-64
5305 LET X=VAL (O$)
5310 LET VL=V2: LET HL=H2
5315 IF X<H1 THEN  GO TO 5330
5316 IF X<H2 THEN  GO TO 5335
5320 IF X>=12 THEN  LET H1=12: L
ET H2=15: GO TO 5340
5330 LET H1=X: LET H2=H1+3
5335 IF Y<V1 THEN  GO TO 5350
5336 IF Y<V2 THEN  GO TO 5360
5340 IF Y>=9 THEN  LET V1=9: LET
 V2=V1+6: GO TO 5360
5350 LET V1=Y: LET V2=Y+6
5360 GO SUB 1800: REM PRINT COLUMNS
5370 GO SUB 1850: REM PRINT ROWS
5380 IF V2=VL AND H2=HL THEN  GO
 SUB 1650: GO TO 1100
5390 GO SUB 1700
5395 GO TO 1100
5400 REM **** MAKE LIST OF ELEME
NTS IN I$ ***
5405 LET P=0: LET LP=1: LET T$="
": LET U$=""
5410 DIM E$(20,5)
5420 LET P=P+1: IF P>LEN (C$) TH
EN  LET E$(LP)=T$: RETURN
5430 LET U$=C$(P)
5440 IF U$="+" OR U$="-" OR U$="
*" OR U$="/" THEN  GO TO 5460
5445 IF U$="^" OR U$="(" OR U$="
)" THEN  GO TO 5460
5450 LET T$=T$+U$: GO TO 5420
5460 LET E$(LP)=T$: LET LP=LP+1
5470 LET E$(LP)=U$: LET LP=LP+1
5480 LET T$="": GO TO 5420
5500 REM *** REPLICATING ****
5505 LET P=0: LET X$="": LET Y$=
"": LET Z$="": LET T$=""
5510 IF R$(3)<>"(" AND R$(4)<>"(
" THEN  GO TO 5660
5515 IF R$(6)<>"-" AND R$(7)<>"-
" AND R$(8)<>"-" THEN  GO TO 5660
5520 LET P=P+1
5530 LET T$=R$(P)
5540 IF T$="(" THEN  LET P=P+1:
GO TO 5570
5550 LET X$=X$+T$
5560 GO TO 5520
5570 LET T$=R$(P)
5580 IF T$="-" THEN  LET P=P+1:
GO TO 5610
5590 LET Y$=Y$+T$
5600 LET P=P+1: GO TO 5570
5610 LET T$=R$(P)
5620 LET Z$=Z$+T$
5630 LET P=P+1
5640 IF P<=LEN (R$) THEN  GO TO 5610
5650 RETURN
5670 PRINT AT 18,0;"ERROR IN REP
LICATE STRING"
5680 RETURN
5700 REM **** DECIDE COLUMN OR R
OW ****
5710 PRINT AT 18,0;"REPLICATE:
                "
5720 INPUT  LINE R$
5730 GO SUB 5500
5765 LET R1=VAL (X$(2 TO )): LET
 X$=X$(1)
5770 LET R2=VAL (Y$(2 TO )): LET
 Y$=Y$(1)
5775 LET R3=VAL (Z$(2 TO )): LET
 Z$=Z$(1)
5780 IF R1=R2 AND R1=R3 THEN  GO
 TO 5800
5790 IF X$=Y$ AND X$=Z$ THEN  GO
 TO 5900
5795 GO TO 5700
5800 REM **** REPLICATE COLUMN *****
5805 LET C$=F$((CODE (X$)-65)*15+R1)
5810 GO SUB 5400: REM MAKE LIST
OF ELEMENTS
5820 FOR K=CODE (Y$) TO CODE (Z$
): LET T$=""
5830 FOR J=1 TO LP
5840 LET U$=E$(J)( TO 1)
5850 IF U$=X$ THEN  LET T$=T$+CH
R$ (K)+E$(J,2 TO ): GO TO 5880
5860 IF U$="+" OR U$="-" OR U$="
*" THEN  LET T$=T$+U$: GO TO 588
5865 IF U$="/" OR U$="^" OR U$="
(" OR U$=")" THEN  LET T$=T$+U$:
 GO TO 5880
5870 IF U$<>"" THEN  LET T$=T$+U$
5880 NEXT J
5882 LET U$="": FOR i=1 TO LEN (
T$): IF T$(i TO i)<>" " THEN  LE
T U$=U$+T$(i TO i)
5885 NEXT i
5890 LET F$((K-65)*15+R1)=U$: LE
T H$((K-65)*15+R1)=""
5895 NEXT K: GO SUB 1900: RETURN
5900 REM **** REPLICATE ROW ****
5905 LET C$=F$((CODE (X$)-65)*15
+R1)
5910 GO SUB 5400: REM MAKE LIST
OF ELEMENTS
5920 FOR K=R2 TO R3: LET T$=""
5930 FOR J=1 TO LP
5940 LET U$=E$(J,1 TO 1)
5950 IF U$)="A" AND U$(="O" THEN
  LET T$=T$+U$+STR$ (K): GO TO 5980
5960 IF U$="/" OR U$="^" OR U$="
(" OR U$=")" THEN  LET T$=T$+U$:
 GO TO 5980
5970 IF U$<>"" THEN  LET T$=T$+U$
5980 NEXT J
5990 LET F$((CODE (X$)-65)*15+K)
=T$: LET H$((CODE (X$)-65)*15+K)
=""
5995 NEXT K: GO SUB 1900: RETURN
```

Bit für Bit

Auf 68000-Systemen benutzt die Peripherie den gleichen Kommunikationsbus, an den auch Speicher und CPU angeschlossen sind. Wir untersuchen die Ein- und Ausgabe (E/A) und wie die CPU mit der Methode der „Memory Map" arbeitet.

Es gibt zwei Hauptmethoden, E/A-Geräte anzuwählen und mit ihnen Daten auszutauschen:

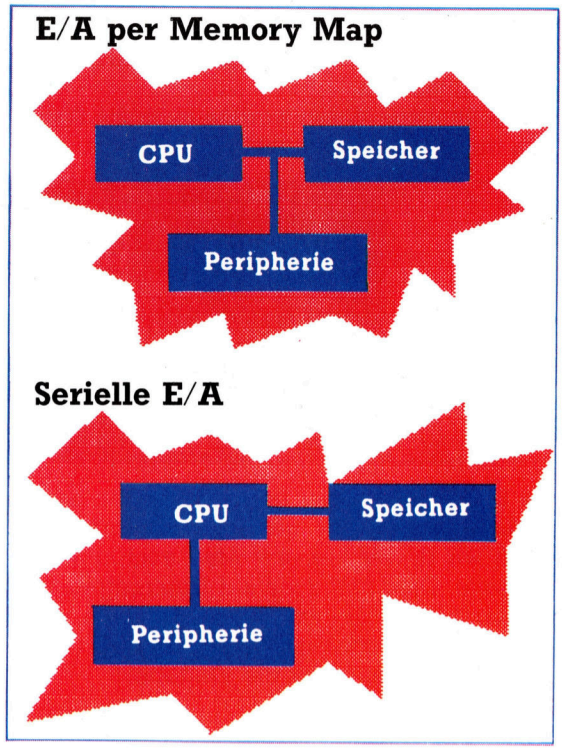
- Per „Memory Map": Hier werden die E/A-Geräte über entsprechende elektronische Schaltungen mit dem zentralen Datenbus (an den auch Speicher und CPU angeschlossen sind) verbunden. Der Prozessor spricht die Geräte dabei wie RAM-Speicher im „Memory Map"-Verfahren an. Wenn mehrere Peripheriegeräte den Bus gleichzeitig beanspruchen, verliert das System allerdings an Geschwindigkeit.
- Isolierte E/A: Diese Methode arbeitet mit einem separaten Bus für E/A-Geräte. Hier ist zwar die Übertragungsgeschwindigkeit höher, doch braucht jedes einzelne Peripheriegerät ein eigenes Befehlsmodul.

Es überrascht nicht, daß der 68000 mit der Memory Map arbeitet. In einer Reihe über die Struktur der Microcomputer hatten wir das Prinzip der E/A-Steuerung von Peripheriegeräten per Memory Map schon ausführlich beschrieben.

Hier die drei wichtigsten Informationen, die der Prozessor für das Memory-Map-System braucht:

- Status: Da Peripherievorgänge Zeit brauchen (beispielsweise zum Drucken eines Zeichens oder für die Tastaturabfrage), verhindern Statusinformationen, daß das Gerät vor Beendigung eines Befehls neue Anweisungen erhält. In diese Kategorie fallen auch Informationen, die nicht unmittelbar mit dem laufenden Vorgang zu tun haben, aber Auskunft über Ablaufart oder Fehler geben.
- Steuerung: Mit Steuerregistern werden Peripheriegeräte eingestellt (konfiguriert) oder aktualisiert (z. B. die Anweisung an ein Diskettenlaufwerk, einen Datenblock zu lesen, oder das Einschalten einer bestimmten Baudrate für einen Übertragungskanal).
- Daten: Schließlich werden Daten an ein Gerät gesandt, von dort empfangen und bis zum Abschluß des Vorgangs zwischengespeichert.

Auf der Ebene der Impulsübermittlung des Hauptdatenbusses können die Vorgänge in den Schnittstellenchips schon recht komplex



Das Bild zeigt die Unterschiede zwischen der seriellen E/A, die über einen eigenen Peripheriebus Daten übermittelt, und der E/A per „Memory Map", bei der die Peripheriegeräte ihre Daten aus dem gleichen Bus erhalten, mit dem auch der Speicher angesprochen werden kann.

werden. Diese komplizierte Steuerung nehmen dem Programmierer normalerweise jedoch spezielle Interfacechips ab. Motorola bietet zwei dieser Chips an – ACIA für die serielle und PIA für die parallele Steuerung.

Serielle Geräte brauchen nur vier Leitungen zum Chip – zwei zum Senden und zwei zum Empfangen von Informationen. Die Daten werden hier nacheinander und bitweise übermittelt, während sie bei der parallelen Kommunikation im Byteformat ankommen.

Die Schnittstellenchips bieten große Flexibilität beim Hardwareaufbau. Dabei kann aber auch die Bitzahl eines Steuerwortes recht groß werden. So werden beispielsweise die ACIA-Steuerbits folgendermaßen zugeordnet:

Bit 0 und 1	Takt-Teilungsfrequenz und Initialisierung (serielle Bitrate)
Bit 2 bis 4	serielles Zeichenformat (z. B. Parity Bit, Zahl der Stopbits)
Bit 5 und 6	Steuerbits für die Übertragung (z. B. Interrupt AN)
Bit 7	Steuerbit für den Empfang auf Interrupt AN gesetzt.

Diese Bits regeln im wesentlichen die Datenübertragung zwischen Computer und Peripherie. Ohne sie ist keine geregelte Kommunikation möglich.

Jede serielle Datenübertragung braucht zwei Befehle. Ein Befehl stellt die ACIA auf die Anforderungen des Peripheriegerätes ein. Hier ein Beispiel:

```
MOVE.B #$3,ACIACON    anfänglicher Hard-
                      warereset
```

```
MOVE.B #$15,ACIACON  Hardware konfigu-
                     rieren
```

In diesem Fall muß ACIACON natürlich zuvor definiert werden.

Auf ähnliche Weise werden auch die Statusbits für die Datenübermittlung gelesen:

Bit 0	Im Empfangsregister sind Daten (ein neues Zeichen steht bereit)
Bit 1	Übertragungsregister ist leer (ein neues Zeichen kann gesendet werden)
Bit 2 und 3	Signale zur Modemsteuerung (CTS und DCD)
Bit 4 bis 6	Fehleranzeige für Empfangsdaten (z. B. bei einem Parityfehler)
Bit 7	Interruptanforderung

Da Statusregister nur gelesen werden und in Steuerregistern nur gespeichert wird, brauchen wir in der Memory Map auch nur eine Adresse. Dies trifft auch auf Datenregister zu, da daraus nur die empfangenen Daten gelesen oder Sendedaten geschrieben werden.

Hier ein Beispiel für die Eingabe:

```
INCH   BTST     #0,ACIASTAT  steht ei
                             chen zu
                             fügung!
       BEQ      INCH         Falls nic
                             nochma
                             prüfen
       MOVE.B   ACIADATA,DO  Zeicher
                             Parame
                             DO lese
       RTS                   Rücksp
                             von der
                             routine
```

Das Programm wartet, bis Bit 0 des Steuerregisters auf 0 gesetzt ist und liest dann das Datenregister. ACIASTAT entspricht ACIACON.

Auf die gleiche Weise kann eine Subroutine auch Zeichen ausgeben:

```
OUTCH  BTST     #1,ACIASTAT  warten,
                             das Dat
                             gister le
       BEQ      OUTCH        Falls nic
                             nochma
                             verzwei
                             und prü
       MOVE.B   DO,ACIADATA  Zeichen
                             geben
       RTS
```

Zur Wiederholung muß danach nur folgende Subroutine aufgerufen werden:

```
ECHO   JSR      INCH         Zeichen
                             lesen
       JSR      OUTCH        Zeichen
                             geben
       BRA      ECHO         und Abl
                             wiederh
```

Da der Computer die Daten zeichenweise übermittelt, ist dieser E/A-Ablauf nicht besonders schnell, zeigt aber den Grundmechanismus der E/A-Programmierung.

Die Befehle des 68000

Bevor wir uns mit den Interrupts des 68000 beschäftigen, geben wir einen Überblick über alle bisher beschriebenen Befehle. Hier zunächst die Gruppe der Kopierbefehle, bei denen die MOVE-Anweisung eine wichtige Rolle spielt:

Befehl	Vorgang
MOVE	Transportiere von der Quelle zum Ziel
MOVEM	Transportiere mehrere Daten und Adreßregister (praktisch für die Parameterübergabe bei Subroutinen)
MOVEA	Transportiere den Inhalt einer Adresse in ein Adreßregister
LEA	Lade die Quelladresse in ein Adreßregister
MOVEQ	Schnellübertragung einer kleinen Konstanten in ein Datenregister
PEA	Schiebe effektive Adresse auf den Stapel
SWAP	Vertausche untere und obere Hälfte eines Datenregisters
EXG	Tausche die Daten zweier Register
LINK/UNLK	Wird zur Stacksteuerung für die Parameterübergabe bestimmter Subroutinen eingesetzt

Dieser Befehlssatz bietet zwar viele flexible Möglichkeiten, doch sollten Sie besonders bei weniger gebräuchlichen Anweisungen wie LINK/UNLK vorsichtig sein. Die Befehle für die Ganzzahlenarithmetik:

Befehl	Vorgang
ADD	Quelle auf Ziel addieren
ADDA	Quelle auf das Zieladreßregister addieren
ADDI	Unmittelbare Daten auf das Ziel (datenveränderndes Register) addieren
ADDQ	Schnelladdition für kleine Konstante (1 bis 8)
ADDX	Addieren mit Übertrag
SUB (A/I/Q/X)	Subtrahiere Quelle vom Ziel in einer der Adreßvarianten, wie bei ADD gezeigt
CMP(A/I)	Vergleiche Quelle mit Ziel und setze die Bedingungscodes
CMPM	Vergleiche Speicherstellen unter Nach-Inkrementierung des Pointers
NEG(X)	Negiere das Operanden-(Daten-) Register
EXT	Vorzeichenerweiterung des Operanden-(Daten-)Registers
MULU	Multipliziere Daten ohne Vorzeichen in der angegebenen effektiven Adresse, und speichere das Ergebnis im Datenregister
MULS	Wie bei MULU, jedoch für Daten mit Vorzeichen
DIVU(S)	Dividiere Ziel-(Daten-)Register

	durch Quelloperanden. Quotient und Rest werden im Zielregister gespeichert
TST	Setze die Bedingungscodes je nach Operand
TAS	Teste und setze das höchstwertige Bit des Operanden
CLR	Setze Operand auf 0

Bei diesen breitgefächerten Anweisungen müssen Sie darauf achten, daß Sie für den Operanden den richtigen Adreßmodus einsetzen. Es folgen die Befehle der BCD-Arithmetik:

Befehl	Vorgang
ABCD	Addiere BCD-Operanden und speichere die Summe im Ziel
SBCD	Subtrahiere die Operanden
NBCD	Negiere den BCD-Operanden

Beachten Sie, daß es keinen BCD-Befehl für Multiplikation gibt. Die folgenden Logikbefehle sind jedoch umfassend:

Befehl	Vorgang
AND	Logisches AND der Quelle mit dem Ziel. Mindestens ein Operand muß Datenregister sein
ANDI	Logisches AND mit unmittelbaren Daten als Quelle
OR	Logisches OR mit den gleichen Komponenten wie AND
ORI	Logisches OR mit unmittelbaren Daten
EOR	Exklusives OR
EORI	Exklusives OR mit unmittelbaren Daten
NOT	Einerkomplement bilden

Folgende Befehle steuern das bitweise Testen und Bearbeiten:

Bitverarbeitende Befehle

Befehl	Vorgang
BTST	Das im Quelloperanden festgelegte Bit testen
BSET	Zieloperand testen und Bit setzen
BCLR	Bit testen und löschen
BCHG	Bit testen und Operand ändern

Beachten Sie, daß sie mit dem getesteten Bit keine Abläufe ausführen müssen.

Befehle für Verschiebung und Rotation

Befehl	Vorgang
ASI	Arithmetische Linksverschiebung
ASR	Arithmetische Rechtsverschiebung
LSL	Logische Linksverschiebung
LSR	Logische Rechtsverschiebung
ROL	Operand nach links rotieren und Übertragsbit entsprechend setzen
ROR	Operand nach rechts verschieben und Übertragsbit entsprechend setzen

Schließlich noch die Befehle zur Steuerung von Programmen und Subroutinen:

Befehle zur Programmsteuerung

Befehl	Vorgang
BRA	Unbedingte Verzweigung
JMP	Unbedingter Sprung (sehr praktisch, wenn Sie die Sprungadresse zuvor berechnen wollen)
Bcc	Bedingte Verzweigung. Abhängig vom Testen des mit "cc" angegebenen Bedingungscodes. Hier die Codes im einzelnen:

Für Operanden mit Vorzeichen:

GT	größer als
LT	kleiner als
GE	größer oder gleich
LE	kleiner oder gleich
VS	Überlaufbit gesetzt
VC	Überlaufbit nicht gesetzt

Für Operanden ohne Vorzeichen:

EQ	gleich
NE	ungleich
MI	Minus
PI	Plus
HI	größer als
LS	kleiner oder gleich
CS	Übertragsbit gesetzt
CC	Übertragsbit nicht gesetzt
DBcc	Zähler dekrementieren und verzweigen Bedingung 'cc' (wie oben)

Befehle zur Subroutinensteuerung

Befehl	Vorgang
JSR	Auf Subroutine springen
BSR	Verzweige auf Subroutine
RTS	Rücksprung von der Subroutine

Diese Zusammenfassung enthält nicht alle Befehle des 68000. So gibt es für Betriebssystemdesigner weitere Anweisungen zur Systemsteuerung, die außerhalb der Thematik dieses Kurses liegen. Diese Befehle drehen sich hauptsächlich um das Statusregister, die Stackpointer und die (‚Trap' genannten) ‚Softwareinterrupts'. Sie sind im Anwenderhandbuch des 68000 ausführlich beschrieben.

Heiße Räder

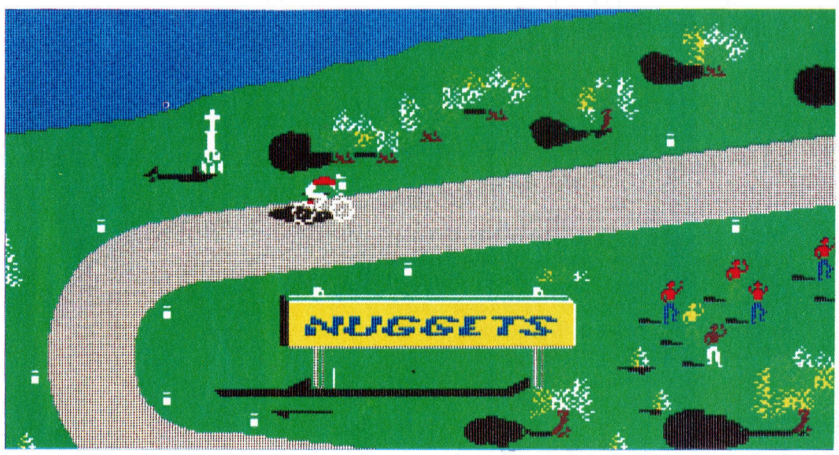
Zwei Spiele von Activision sollen die Herzen der Rennfreunde höher schlagen lassen: das Great American Cross-Country Race für den Commodore 64 und die Atari-Computer sowie Tour de France für den C64.

Jeder Wagen, der mehr als 220 mph schafft, sollte ohne größere Schwierigkeiten einem Polizeiwagen entkommen. Nicht aber in dem Great American Cross-Country Race, und dies ist nur eine der vielen verwunderlichen Details in diesem Programm. Als Fahrspiel bietet es noch weitere Unmöglichkeiten. Ausgestattet mit einem Viergang-Getriebe können sie die Maschine überdrehen, wenn Sie nur einige Sekunden lang versuchen, mit 30 mph im ersten Gang zu fahren, wobei man wiederum ganz gemütlich mit 120 mph im dritten fahren kann. Auf alle Fälle können sie ihren Wagen jederzeit zur nächsten Tankstelle schieben, auftanken und wieder am Rennen teilnehmen, obwohl der Motor explodierte.

Das Rennen läuft in den USA, und es geht darum, schneller als alle anderen Mitbewerber, entweder ihre Mitspieler oder die vom Computer gesteuerten Gegner, das Ziel zu erreichen. Verschiedene Routen stehen zur Auswahl, von einer kurzen Piste bis hin zur großen New York-Los Angeles Querfeldein-Strecke.

Es gibt drei Widrigkeiten in diesem Spiel – den Tank leerfahren, die Maschine zum Explodieren bringen und der mit Radar ausgestatteten Polizei in die Hände zu fallen. Alles läßt sich ganz einfach durch Auftanken, Schalten bzw. Abbremsen vermeiden. Meistens befahren sie einen Highway, und somit ist es schade, daß der Sound der Maschine nicht viel mehr als ein dumpfes Heulen hergibt.

Beide Spiele sind, trotz unterschiedlicher Aufmachung, Fahrspiele gegen die Zeit und andere Mitspieler. Diese Spiele werden sicherlich viele Liebhaber finden.



Tour de Force

Es ist nicht klar, wen das Spiel ansprechen soll, da es weder besonders reizvoll ist noch große Geschicklichkeit erfordert. Wie dem auch sei, es basiert auf großen Filmhits wie z. B. „Cannonball" oder „Auf dem Highway ist die Hölle los". So besteht die Möglichkeit, daß das Great American Cross-Country Race für diejenigen ein Hit ist, die diese Streifen genossen haben.

Tour de France bietet eine angenehm scrollende Szenerie, Schatten, Anzeigetafeln mit wechselnden Mitteilungen, jubelnde Fans und realistische Bewegungen. Die Graphik macht dieses Programm zur Tour de Force. Jede Etappe des Rennens soll schnellstmöglich absolviert werden, um sich so das begehrte gelbe Trikot des Spitzenreiters dieses weltbekannten Radsportereignisses zu sichern.

Beim Laden wird die Titelgraphik von einer triumphalen Marseillaise begleitet. Doch während der Etappen, was sollten sie auch mit pastoraler Musik entlang der idyllischen Straßen, klingt es recht schauerlich. Dies ist allerdings bei einem sonst so gelungenen Programm ein erträglicher Nachteil. Zur Technik: Pedalbewegungen werden durch Rechts- und Linksbewegen des Joysticks erreicht, nach vorne drücken wechselt die Gänge und zurückziehen bremst das Fahrrad. Gelenkt wird, indem man den Joystick nach einer Seite bewegt und gleichzeitig den Feuerknopf drückt. Obwohl es so simpel ist, verlangt das Spiel ein wohlüberlegtes Handeln, und durch geschickte Bedienung lassen sich durchaus ein paar Minuten in jeder Etappe einsparen. So kann zum Beispiel das Durchfahren der Kurven an der Innenseite die Geschwindigkeit optimieren. Beide Spiele bei Activision.

Fachwörter von A bis Z

UART = UART
Ein „UART" oder „Universal Asynchronous Receiver/Transmitter" ist eine asynchrone Universalschnittstelle mit einem Parallel/Seriell-Wandler und einem umgekehrt arbeitenden Umsetzer, beide in einem Chip vereinigt. UARTs dienen als Interface zwischen dem parallelen Rechnerbus und einem angeschlossenen seriellen Peripheriegerät oder Kommunikationsnetz.

Unconditional Branch = Unbedingter Sprung
Ein Sprungbefehl bewirkt allgemein, daß der Rechner aus der aktuellen Kommandofolge aussteigt und das Programm an einer anderen Stelle fortsetzt. Bei einem „unbedingten" Sprung ist die Ausführung nicht an das Erfülltsein einer oder mehrerer Bedingungen gebunden, sondern sie findet in jedem Fall statt.

Der Unterschied zwischen einem „bedingten" und einem unbedingten Sprung läßt sich z. B. an den entsprechenden 6502-Assemblerkommandos verdeutlichen: Die Anweisung BEQ = Branch (if) EQual (zero) ist ein bedingter Sprungbefehl – der Sprung erfolgt nur, wenn das Zero-Flag im Statusregister des Prozessors gesetzt ist. Das Kommando JMP = Jump veranlaßt dagegen einen unbedingten Sprung, d. h. die Fortsetzung der Programmbearbeitung bei der angegebenen Adresse ohne Abfrage einer Bedingung.

Unix = Unix
Das Betriebssystem Unix wurde in den Bell Laboratories zunächst für einen PDP-7-Minicomputer von DEC entwickelt und dann auf die PDP-11-Serie übertragen. Als Mehrbenutzersystem gestattet es den gleichzeitigen Zugriff mehrerer Teilnehmer auf dieselbe Anlage; es hat seit seiner Vorstellung im Jahr 1971 bei Minicomputern und 16-Bit-Micros große Verbreitung gefunden.

Unix war das erste Betriebssystem mit einer leistungsfähigen flexiblen Benutzerschnittstelle, und viel von seinem erfolgreichen Konzept – u. a. die hierarchische Dateistruktur – ist bei anderen Systemen (z. B. bei MS-DOS) übernommen worden.

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.

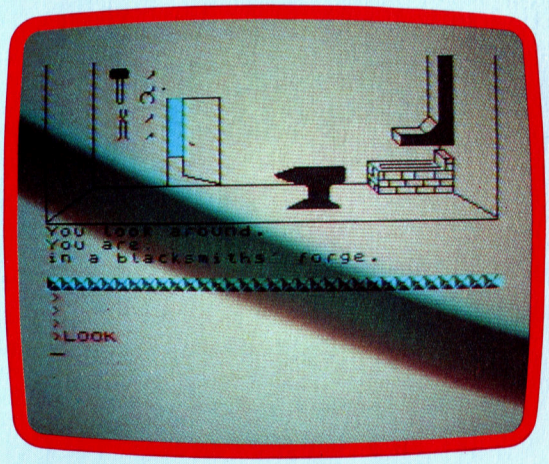
Validity Check = Gültigkeitskontrolle
Beim Validity Check wird softwaremäßig geprüft, ob die Eingangsdaten für ein Programm innerhalb des zulässigen Wertebereiches liegen. Vor allem bei der Tastatureingabe ist das sehr zweckmäßig. Die BASIC-Anweisung INPUT X z. B. erwartet das Eintippen von Ziffern und nicht von Buchstaben. Der erfahrene Programmierer baut in seine Routinen weiterreichende Gültigkeitskontrollen ein, um zu vermeiden, daß offensichtlich fehlerhafte Daten in die eigentliche Verarbeitung geraten und dann vielleicht einen Programmabsturz auslösen.

Der Validity Check erfolgt im Rahmen der Datenübergabe an ein Programm. Durch ein paar zusätzliche Anweisungszeilen läßt sich leicht feststellen, ob die Eingangsgrößen akzeptabel sind; wenn nicht, ergeht an das Eingabegerät die Aufforderung, neue Werte bereitzustellen.

Variable = Variable
Einzelne Zeichen oder Zeichenfolgen, die in höheren Programmiersprachen als Platzhalter für numerische oder alphanumerische Größen verwendet werden, bezeichnet man als „Variable"; im Laufe der Programmausführung lassen sich ihnen wechselnde Werte zuweisen. Variablen stellen wesentliche Elemente der Programmierung dar, weil sie neben der Einführung sinnvoller Namen auch den Aufbau von Datenstrukturen erlauben.

VDU = Datensichtgerät
VDU bedeutet „Video Display Unit", und dieser Begriff wird oft nicht nur für das Sichtgerät selbst gebraucht, sondern auch für die Elektronik-Baugruppe zur Erzeugung der Steuersignale im Rechner. Das Datensichtgerät arbeitet mit einer Kathodenstrahlröhre, auf deren Schirm das Bild zeilenweise durch einen Raster-Scan aufgebaut wird. Im Prinzip genügt daher als Sichtgerät auch schon ein Heimempfänger, obwohl spezielle Monitore mit RGB-Eingang bessere Bilder liefern. Bei tragbaren Computern setzen sich jetzt zunehmend Flachbildschirme mit reduziertem Leistungsbedarf durch, entweder als LCD-Anzeigen oder als Plasma- bzw. Luminiszenz-Displays.

Die Anzeige muß bei Sichtgeräten kontinuierlich aufgefrischt werden, was zwar einigen elektronischen Aufwand erfordert, aber andererseits die rasche Veränderung der Bilder auf dem Schirm zuläßt.



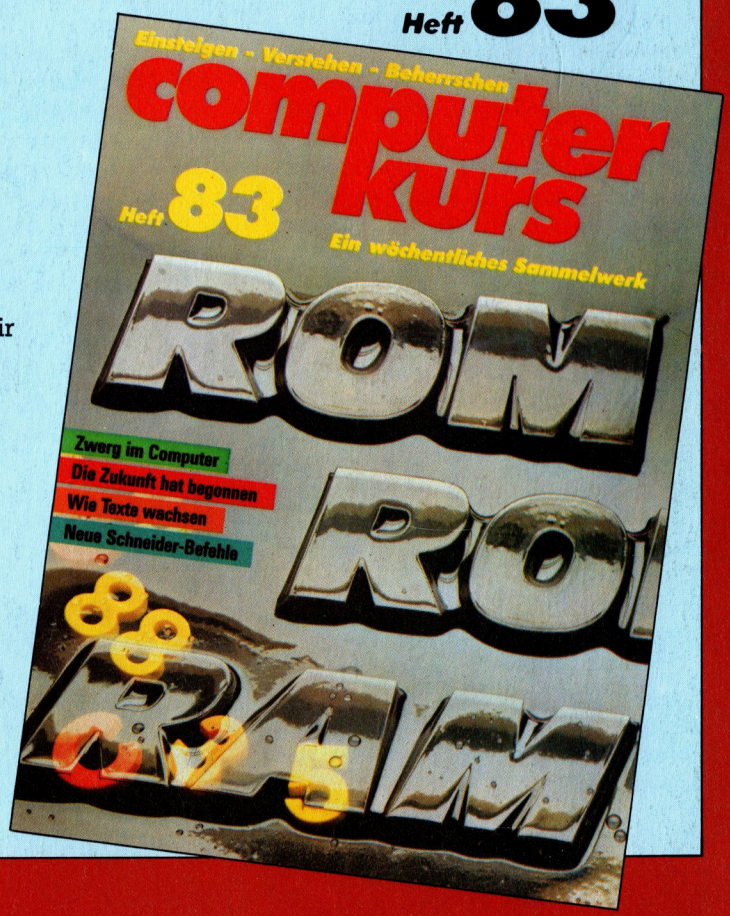
Bei den Datensichtgeräten mit Bildröhre wird der Schirm fünfzigmal pro Sekunde durch einen Elektronenstrahl abgetastet. Der Eindruck einer gleichmäßigen Bildhelligkeit entsteht wesentlich durch das Nachleuchten des Phosphors.

Bildnachweise
2270: Kevin Jones
2273, 2291, 2293: Caroline Clayton
2276–2289: Johanne Ryder

Der CD-Rom Laserplattenspieler gilt heute als unerläßliches Peripheriegerät für jeden Micro-Besitzer. Das Foto zeigt eins der ersten CD-ROM Laufwerke, das Hitachi Mitte der achtziger Jahre mit einer Kapazität von 552 MByte und einer Übertragungsrate von 176 KByte/s herausbrachte.

computer kurs

Heft 83



Die Zukunft hat begonnen

Die rapide Entwicklung der Microcomputertechnik macht eine verbindliche Vorhersage unmöglich.

Ihren Einsatz bitte

Wetten ist ein fester Bestandteil von 17+4. Wir zeigen Ihnen, wie Sie Ihren Einsatz behalten können.

Text soll wieder wachsen

Wir beschließen unsere Serie über Textkompression mit einem Treiberprogramm.

Commodore-Zwerge

Wir stellen Ihnen ein neues Spiel vor, „Little Computer People“ von Activision für den Commodore 64. In diesem Spiel wird nicht gekämpft, sondern für das Wohlbefinden der Zwerge gesorgt.